МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 003.26 + 004.056 + 001.99 DOI 10.17223/20710410/59/2

КРАТКИЕ НЕИНТЕРАКТИВНЫЕ АРГУМЕНТЫ С НУЛЕВЫМ РАЗГЛАШЕНИЕМ НА ОСНОВЕ НАБОРОВ ПОЛИНОМОВ

И. В. Мартыненков

АО «КВАНТ-ТЕЛЕКОМ», г. Москва, Россия

E-mail: mivpost@yandex.ru

Рассматриваются принципы построения и основные виды кратких неинтерактивных аргументов с нулевым разглашением для выполнимости булевых функций с использованием различных полиномиальных наборов, в том числе и для аутентифицированных данных. Приведены различные алгоритмы формирования публичных параметров, доказательств достоверности вычислений и их верификации. Представлены необходимые криптографические преобразования и некоторые примеры многосторонних верифицируемых вычислений.

Ключевые слова: *доказательство знания, достоверность вычислений, нулевое разглашение, краткие неинтерактивные аргументы.*

ZERO-KNOWLEDGE SUCCINCT NON-INTERACTIVE ARGUMENTS OF KNOWLEDGE BASED ON SETS OF POLYNOMIALS

I. V. Martynenkov

JSC "KVANT-TELECOM", Moscow, Russia

The paper discusses the basic principles of construction and the main types of zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) which is used in the model of a three-way insecure computing environment and based on sets of polynomials. A number of zk-SNARK cryptographic protocols with different algorithms for generating public parameters (Trusted Setup) are given, constructing succinct proofs of reliability calculations (Prover) and public/designated verification of proofs (Verifier). The cases of satisfying the feasibility of discrete functions (arithmetic/ Boolean circuits) using different polynomial sets are presented in quadratic arithmetic programs (QAP), square arithmetic programs (SAP), quadratic span programs (QSP), square span programs (SSP), quadratic polynomial programs (QPP), etc., also the use of authenticated data are described. The cryptographic transformations needed to build zk-SNARKs based on symmetric and asymmetric hash functions, exponential knowledge problems, digital signatures, homomorphic encryption, bilinear pairings based on elliptic curves, etc. are presented. Examples of multilateral verifiable calculations based on zk-SNARK are given.

Keywords: *proof of knowledge, reliability of calculations, zero knowledge, succinct non-interactive arguments.*

Введение

Системы доказательств для задач класса NP позволяют доказывающему, в том числе ненадёжному, убедить проверяющего в том, что объект w обладает свойством L или, что эквивалентно, $w \in L$, где L — NP-полный язык. В общем случае проверяется выполнимость дискретной функции $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}^l$, представляющей бинарное отношение $\mathcal{R}_C = \{(x,w) \in \{0,1\}^n \times \{0,1\}^h : C(x,w) = 0^l\}$ с языком $\mathcal{L}_C = \{x \in \{0,1\}^n : \exists w \in \{0,1\}^h \, (C(x,w) = 0^l)\}$. Если $l = 1$, то функция $C(x,w) = \{0,1\}$ — булева схема, если $l > 1$, то арифметическая. Преобразование в неинтерактивный режим выполняется за счёт доверенного формирования публичных значений на основе секретных параметров защиты. Например, в криптовалюте Zcash этот этап называется «церемония» [1].

Как правило, краткие неинтерактивные аргументы с нулевым разглашением включают тройку алгоритмов (G, P, V): формирователь ключей (Key Generator), доказывающего (Prover) и верификатора (Verifier). Алгоритм G принимает $\lambda \in \mathbb{N}$, дискретную функцию $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}^l$ и проводит однократную дорогостоящую настройку общедоступных параметров выводом ключей доказательства σ (Proving Key) и верификации τ (Verification Key). На основе σ, публичного входа схемы x и секрета w, знание которого доказывается, алгоритм P вычисляет доказательство π, подтверждающее знание w. Для согласования с англоязычными источниками секрет w, знание которого доказывается, будем обозначать как секретное свидетельство w. Верификатор V проверяет, что π является допустимым доказательством для $x \in \mathcal{L}_C$ и $(x,w) \in \mathcal{R}_C$:

$$(\sigma, \tau) \leftarrow G(1^\lambda, C),$$
$$\pi \leftarrow P(\sigma, x, w),$$
$$0/1 \leftarrow V(\tau, x, \pi).$$

Для параметра защиты $\lambda \in \mathbb{N}$, дискретной функции $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}^l$ и пары ключей $(\sigma, \tau) \leftarrow G(1^\lambda, C)$ если выводится пара (x, π), такая, что не существует $(x,w) \in \mathcal{R}_C$, то $V(\tau, x, \pi)$ отклоняет доказательство. Протоколы с нулевым разглашением должны обладать свойствами полноты, корректности и нулевого разглашения [2].

В дальнейшем для краткости вместо наименования «краткие неинтерактивные аргументы с нулевым разглашением» будем использовать стандартное обозначение «протоколы zk-SNARK» (Zero-Knowledge Succinct Non-interactive Argument of Knowledge). Конструирование и применение протоколов zk-SNARK включает [3] описание функции $F(\cdot)$ на языке программирования, компиляцию исходного кода в арифметическую/булеву схему, задание канонического представления схемы в R1CS-форме, переход к полиномиальному представлению (QAP, SAP, QSP, SSP, QPP и др.) для сведения доказательства к выборочной проверке полиномов в секретной точке, внесение значений полиномов в степень порождающих элементов групп и выполнение билинейных спариваний. Подробное описание основополагающих этапов построения и принципов работы протоколов zk-SNARK представлено в [4].

Протоколы zk-SNARK являются строительными блоками многих криптографических приложений, в число которых входят защищённые многосторонние распределённые вычисления [5–8], групповые подписи [9], гибкие системы проверки [10], анонимные учётные данные [11], делегируемые учётные данные [12], электронное голосование [13, 14], финансовые технологии распределённых реестров на основе криптовалют [1, 15–21], схемы делегирования вычислений функций по заданным входам с дока-

зательством корректности результатов [22], конфиденциальные подтверждения пересечений множеств (Private Set Intersection, PSI) без разглашения сведений о них (мощность, состав и др.) [23, 24], конфиденциальное суммирование элементов по приватным атрибутам (Private Intersection-Sum, PIS) [25–27], обращения в базы данных недоверенного провайдера без разглашения запрашиваемых индексов (Private Information Retrieval, PIR) [28] на основе дискреционных схем доступа [29, 30] с использованием «шумового» подхода, по ответам которых невозможно выделение факта присутствия/отсутствия конкретного объекта, и др.

Исчерпывающее описание всех разновидностей протоколов zk-SNARK не соответствует формату статьи, поэтому в ней рассматривается ряд исторически базовых схем на основе полиномиальных наборов.

1. Протокол Й. Грота на основе перестановки элементов векторов

Протокол DV zk-SNARK Й. Грота [31] использует предположение о вычислительной надёжности задачи Диффи — Хеллмана (q-Computational Power Diffie — Hellman Assumption, q-CPDH). В [31] приводится пример использования публичной перестановки ρ, которая удовлетворяет равенствам $b_i = a_{\rho(i)}$ для $i \in \{1, \ldots, n\}$. Для краткости обозначим $\{1, \ldots, n\} = [n]$. Более полно идеи применения перестановок раскрыты в протоколе Х. Липмаа [32]. Рассматривается матрица n^2 значений $a_{11}, \ldots, a_{nn}, b_{11}, \ldots, b_{nn}$, соответствующих левым/правым входам вентилей булевой схемы. Значения в столбцах фиксируются обязательствами $c_1, \ldots, c_n$, $c_j = g^{r_j} \prod\limits_{i \in [n]} g_i^{a_{ij}}$, значения в строках — обязательствами $d_1, \ldots, d_n$, $d_i = g^{s_i} \prod\limits_{i \in [n]} g_{j(n+1)}^{a_{ij}}$. В результате зафиксированные n значений в c_j распределяются между различными n значениями обязательств d_i. Необходимо построить доказательство, подтверждающее, что (c_j, d_i) содержат столбцы и строки одной и той же мартицы.

Алгоритм формирования ключей $(\sigma = (ck, \hat{\sigma}, \bar{\sigma}, \dot{\sigma})) \leftarrow G(1^k)$

1. Формируются параметры группы $gk = (p, \mathbb{G}, \mathbb{G}_T, e)$, порождающий элемент $g \in \mathbb{G} \setminus \{1\}$, где p — порядок группы $\mathbb{G}$; e — билинейное спаривание.
2. Фиксируется четвёрка ограничений множеств для $q = n^2 + 3n - 2$ значений, где n — количество проводов (переменных) схемы (дискретной функции):

$$\begin{gathered} \tilde{S} = \{1, \ldots, n\},\ \bar{S} = \{(n+1), \ldots, n(n+1)\}, \\ \dot{S} = \{l \in [q] : l \neq 0 \pmod{n+2}\},\ S \subset [q]. \end{gathered} \tag{1}$$

   В (1) подмножество $S \subset [q]$, причём q имеет такой вид, что индексы ненулевых значений проводов схемы (переменных дискретной функции) ограничены S.
3. Выбираются случайные значения $x, \alpha \in \mathbb{Z}_p^*$.
4. Формируется ключ доказательства знания для $q = n^2 + 3n - 2$ значений:

$$ck = (gk, g, \ldots, g_q, \hat{g}, \ldots, \hat{g}_q) = (gk, g, \ldots, g^{x^{n^2+3n-2}}, g^{\alpha}, \ldots, g^{\alpha x^{n^2+3n-2}}).$$

5. С использованием подмножеств (1) строится главная ссылочная строка (Common Reference String, CRS) $\sigma = (h, \{h_i\}_{i \in S}) = (g^{\alpha}, \{g_i^{\alpha}\}_{i \in S})$, которая открыто публикуется для всех участников протокола и необходима для формирования и верификации доказательств:

$$\tilde{\sigma}, \bar{\sigma}, \dot{\sigma} \leftarrow G(ck, \tilde{S}), G(ck, \bar{S}), G(ck, \dot{S}).$$

6. Для схемы доказательства формируется секретная «лазейка» $tk = x$, необходимая верификатору для имитирования корректных доказательств.
7. На выход подаётся CRS в виде $\sigma = (ck, \hat{\sigma}, \bar{\sigma}, \dot{\sigma})$.

Алгоритм доказывающего $\pi \leftarrow P(\sigma, r_1, \ldots, r_n, a_{11}, \ldots, a_{nn}, s_1, \ldots, s_n, b_{11}, \ldots, b_{nn})$

Открытый вход имеет вид зафиксированных обязательств $\mathbf{c} = (c_1, \ldots, c_n)$, $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{G}^n$, а секретное свидетельство доказывающего включает показатели экспонент $r_1, \ldots, r_n, s_1, \ldots, s_n \in \mathbb{Z}_p$ и значения входов $a_{11}, \ldots, a_{nn}, b_{11}, \ldots, b_{nn}$, на основе которых строятся обязательства:

$$\forall i, j \in [n] \left(c_j = g^{r_j} \prod_{i \in [n]} g_i^{a_{ij}}, \quad d_i = g^{s_i} \prod_{j \in [n]} g_{j(n+1)}^{b_{ij}}, \quad a_{ij} = b_{ij} \right). \tag{2}$$

Например, на основе $(c_j, d_i) \in \mathbb{G}^2$ можно проверить корректность обязательств за счёт выполнения уравнения $e(g, d_i) = e(c_j, \hat{g})$. Выполняются следующие шаги:

1. Выбирается случайное $t \in \mathbb{Z}_p$, с помощью которого фиксируется «лазейка».
2. Вычисляется набор компонентов доказательства:

$$\begin{aligned}
\pi_L &= g^t \prod_{j \in [n]} g_{j(n+1)}^{-r_j}, \quad \pi_R = g^t \prod_{i \in [n]} g_i^{-s_i}, \\
\hat{\pi}_L &= \hat{g}^t \prod_{j \in [n]} \hat{g}_{j(n+1)}^{-r_j}, \quad \hat{\pi}_R = \hat{g}^t \prod_{i \in [n]} \hat{g}_i^{-s_i}, \\
\bar{\pi}_L &= \bar{h}^t \prod_{j \in [n]} \bar{h}_{j(n+1)}^{-r_j}, \quad \tilde{\pi}_R = \tilde{h}^t \prod_{i \in [n]} \tilde{h}_i^{-s_i}.
\end{aligned}$$

3. На выход подаётся доказательство $\pi = (\pi_L, \pi_R, \hat{\pi}_L, \hat{\pi}_R, \bar{\pi}_L, \tilde{\pi}_R)$.

Алгоритм верификатора $0/1 \leftarrow V(\sigma, \mathbf{c} = (c_1, \ldots, c_n), \mathbf{d} = (d_1, \ldots, d_n), \pi)$

Верификация подтверждается, если с использованием обязательств (2) выполняются все равенства:

$$\begin{gathered}
e(g, \hat{\pi}_R) = e(\pi_R, \hat{g}), \quad e(g, \tilde{\pi}_R) = e(\pi_R, \tilde{h}), \quad e(g, \hat{\pi}_L) = e(\pi_L, \hat{g}), \\
e(g, \bar{\pi}_L) = e(\pi_L, \bar{h}), \quad e(g, \pi_L) \prod_{j \in [n]} e(c_j, g_{j(n+1)}) = e(g, \pi_R) \prod_{i \in [n]} e(g_i, d_i).
\end{gathered}$$

2. Протокол Р. Дженнаро, С. Джентри и Б. Парно

Верифицируемые вычисления тесно связаны с протоколами zk-SNARK. В VC (Verifiable Computation) Р. Дженнаро, С. Джентри и Б. Парно [34] используется протокол Яо [35, 36], предназначенный для двухстороннего вычисления функции F схемы C по частным входным данным a, b. Формируется замаскированное значение схемы $G(C)$, входа $G(a)$ и отправляется второй стороне. Вторая сторона формирует и направляет первой стороне $G(b)$ и вычисляет $G(F(a, b))$, а первая снимает шифрование и получает искомое значение.

Детальнее, для каждого провода w выбирается два случайных значения $k_w^0, k_w^1 \in \{0,1\}^\lambda$, представляющих значения 0 и 1, где λ — параметр защиты. Строятся замаскированные значения вентилей g с входными проводами w_a, w_b и выходным проводом w_z. Для $i, j \in \{0,1\}$ и симметричной схемы шифрования E замаскированная версия $G(g)$ представляет собой четыре шифртекста

$$\gamma_{ij} = E_{k_a^i}(E_{k_b^j}(k_z^{g(i,j)})). \tag{3}$$

Схема шифрования E должна поддерживать свойство «проверяемого диапазона»: существует машина M, для которой $M(k,\gamma)=1$ при $\gamma \in \mathsf{Range}_\lambda(k)$, что позволяет распознавать связанные с каждым вентилем шифртексты. Для скрытия структуры схемы порядок зашифрованных текстов случайно изменяется, т. е. первый зашифрованный текст не обязательно кодирует выход для (0,0). Первая сторона фиксирует соответствия между истинными входами $0/1$ и соответствующими строками k_w^0/k_w^1 так, что вторая сторона не узнает истинные входные биты.

Стороны обмениваются сообщениями, чтобы второй участник получил значения проводов, соответствующие его входным данным (например, k_b^0 или k_b^1). Вторая сторона узнаёт по одному значению на провод, а первая сторона ничего не узнаёт о его вводе. Вторая сторона использует замаскированные значения проводов и выводит результирующее значение вентиля. Первая сторона сопоставляет данные со значениями $0/1$ и получает результат вычисления F.

Алгоритм формирования ключей схемы $(PK, SK) \leftarrow \mathsf{KeyGen}(F, \lambda)$

Используется функция F схемы C с соответствующей QAP [37] размера m и степени d. Полиномиальные задачи QAP также рассмотрены в [38]. Для каждого провода w_i выбираются случайные метки проводов $w_i^0, w_i^1 \leftarrow \{0,1\}^\lambda$. Для всех вентилей g, согласно (3), вычисляются по 4 шифртекста $(\gamma_{00}^g, \gamma_{01}^g, \gamma_{10}^g, \gamma_{11}^g)$. В схеме [37] используется открытый ключ PK для целой функции, а не для каждого отдельного входа [39]. Открытым ключом PK является набор шифртекстов, а секретным ключом SK — набор случайных меток проводов:

$$\begin{gathered} PK \leftarrow \bigcup_g (\gamma_{00}^g, \gamma_{01}^g, \gamma_{10}^g, \gamma_{11}^g)), \\ SK \leftarrow \bigcup_i (w_i^0, w_i^1)). \end{gathered} \tag{4}$$

Алгоритм формирования ключей схемы гомоморфного шифрования $\boldsymbol{\sigma}_x \leftarrow \mathsf{ProbGen}_{SK}(x)$

Формируются ключи $(PK_E, SK_E) \leftarrow \mathsf{KeyGen}_E(\lambda)$ схемы полностью гомоморфного шифрования. Пусть $w_i \subset SK$ являются значениями проводов, представляющими двоичное выражение x. Тогда пара открытый/секретный ключ принимает следующий вид:

$$\begin{gathered} \boldsymbol{\sigma}_x \leftarrow (PK_E, \mathsf{Enc}_E(PK_E, w_i)), \\ \boldsymbol{\tau}_x \leftarrow SK_E. \end{gathered} \tag{5}$$

Алгоритм вычисления функции $\boldsymbol{\sigma}_y \leftarrow \mathsf{Comp}_{PK}(\boldsymbol{\sigma}_x)$

Вычисляется $\mathsf{Enc}_E(PK_E, \gamma_i)$. Строится схема Δ, которая на входе w, w', γ выводит $D_w(D_{w'}(\gamma))$, где D — алгоритм расшифрования, соответствующий алгоритму шифрования E для искажений в протоколе Яо [35, 36]. Для расшифрования пути через шифртексты, аналогично оценке замаскированной схемы Яо [35, 36], многократно вычисляются $\mathsf{Eval}_E(\Delta, \mathsf{Enc}_E(PK_E, w_i)$ и $\mathsf{Enc}_E(PK_E, \gamma_i)$. Для значений проводов $\bar{w}_i$, представляющих $y = F(x)$ в двоичном формате, результатом вычисления функции выступает

$$\boldsymbol{\sigma}_y \leftarrow \mathsf{Enc}_E(PK_E, \bar{w}_i).$$

Алгоритм верификации $y/0 \leftarrow \mathsf{Ver}_{SK}(\boldsymbol{\sigma}_y)$

Используется секретный ключ SK_E (5) для расшифрования $\bar{w}_i \leftarrow \mathsf{Enc}_E(PK_E, \bar{w}_i)$, а SK (4) — для сопоставления значений проводов с выходом y. Если расшифрование и сопоставление успешны, то верификация выполняется.

Источник [40] представляет протокол zk-SNARK для любого NP-языка с небольшими CRS и включает соответствующие компактные доказательства того, что шифртекст получен из открытого текста 0 или 1, а также доказательства выполнимости схем.

3. Протокол П. Фаузи, Х. Липмаа, Б. Чжана

П. Фаузи, Х. Липмаа и Б. Чжан в [41] ввели новую (Λ, v)-схему обязательств с «лазейкой», в которой $n = \mathrm{poly}(k)$, $\Lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbb{Z}_p^n$ — множество без прогрессии, соответствующее [32], при $\lambda_i < \lambda_{i+1}$, $\lambda_i = \mathrm{poly}(k)$ и $v > \max\limits_i \lambda_i$. Фиксируются параметры группы gk, определяются секреты $(\sigma, \hat{\alpha}) \in \mathbb{Z}_p^2$, для $i \in [n]$ формируется набор $(g_{z,\lambda_i}, \hat{g}_{z,\lambda_i}) = (g_z, g_z^{\hat{\alpha}})^{\sigma^{\lambda_i}}$ и устанавливается $(h_z, \hat{h}_z) = (g_z, g_z^{\hat{\alpha}})^{\sigma^v}$. Публичный ключ, применяемый для формирования обязательств, определяется как $\mathsf{ck} = (\mathsf{gk}, (g_{z,\lambda_i}, \hat{g}_{z,\lambda_i})_{i\in[n]}, h_z, \hat{h}_z)$, «лазейка» $td = \sigma$. CRS обязательств ck и «лазейки» ck_{td} формируются аналогичными способами. В результате функции построения обязательств вектора $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, обязательств «лазейки» и раскрытия обязательств «лазейки» (соответствует выполнению равенства $\mathrm{Com}(\mathsf{ck}_{td}, \mathbf{0}, r) = \mathrm{Com}(\mathsf{ck}_{td}, \mathbf{a}, r_{td})$ используют рандомизатор $r \in \mathbb{Z}_p$ и имеют следующий вид:

$$\begin{gathered}\mathrm{Com}(\mathsf{ck}, \mathbf{a}, r) = (h_z, \hat{h}_z)^r \prod_{i=1}^{n} (g_{z,\lambda_i}, \hat{g}_{z,\lambda_i})^{a_i} = A,\\ \mathrm{Com}_{td}(\mathsf{ck}_{td}, r) = \mathrm{Com}(\mathsf{ck}_{td}, \mathbf{0}, r) = (h_z, \hat{h}_z)^r,\\ \mathrm{Open}_{td}(\mathsf{ck}_{td}, td, \mathbf{a}, r) = r - \sum_{i=1}^{n} a_i \sigma^{\lambda_i - v}.\end{gathered} \tag{6}$$

Протокол zk-SNARK на основе произведения Адамара [41] совпадает с вариантом [32], но вместо $(\Lambda, 0)$-схемы используется (Λ, v)-схема связывания. Как и прежде, доказывается, что при данных обязательствах A, B, C (6) выполняются равенства $c_i = a_i b_i$ для $i \in [n]$.

По сравнению с [32] вычисления доказывающего сокращаются до $O(r_3^{-1}(n) \log r_3^{-1}(n))$ в $\mathbb{Z}_p$ и $2O(r_3^{-1}(n))$ возведений в степень в $\mathbb{G}_2$, где $r_3(n)$ — мощность наибольшего множества без прогрессии $\Lambda \in [n]$. Верификация требует пять билинейных спариваний и одно произведение. CRS включает $O(r_3^{-1}(n))$ элементов группы.

В [31, 32] рассмотрены протоколы zk-SNARK на основе перестановок, которые требуют значительных вычислительных ресурсов. В протоколе zk-SNARK с использованием правого/левого сдвига на ξ ($\mathsf{rsft}_\xi([\![(A, \tilde{A})]\!]) = [\![(B, \tilde{B})]\!]$) [41] на основе обязательств A, B доказывается знание ξ для выполнения равенства $a_i = b_{i+\xi}$ при $i \in [n - \xi]$ и $a_{n-\xi+1} = \ldots = a_n = 0$ ($a_{n-\xi+1} = b_1, \ldots, a_n = b_\xi$). Таким образом, $(a_n, \ldots, a_1) = (0, \ldots, 0, b_n, \ldots, b_{\xi+1})$ $\big((a_n, \ldots, a_1) = (b_\xi, \ldots, b_1, b_n, \ldots, b_{\xi+1})\big)$. Данный протокол имеет следующий вид:

Алгоритм формирования ключей $\mathsf{crs}_{\mathsf{rsft}} \leftarrow \mathsf{G}_{\mathsf{rsft}}(1^k, n)$

1. Выводятся параметры группы gk, случайные значения $\sigma, \tilde{\alpha} \in \mathbb{Z}_p$ и $\tilde{g}_z = g_z^{\tilde{\alpha}}$ для $z \in \{0, 1\}$.
2. Для $l \in \{v\} \cup \Lambda$ вычисляются $(g_{1,l}, \tilde{g}_{1,l}) = (g_1, \tilde{g}_1)^{\sigma^l}$. Устанавливается $g_{2,\xi} = g_2^{\sigma^\xi}$.
3. Для $i \in \{\lambda_1, \ldots, \lambda_\xi, v, v + \xi\}$ вычисляются $(g_{2,i}, \tilde{g}_{2,i}) = (g_2, \tilde{g}_2)^{\sigma^i}$.
4. Для $i \in [1, n - \xi]$ вычисляются $(h_{2,i}, \tilde{h}_{2,i}) = (g_2, \tilde{g}_2)^{(\sigma^{\lambda_i + \xi}) - (\sigma^{\lambda_i + \xi})}$.
5. Устанавливается ключ, необходимый для формирования обязательства: $\tilde{\mathsf{ck}} = (\mathsf{gk}, (g_{1,l}, \tilde{g}_{1,l})_{l\in\{v\}\cup\Lambda})$, и выводится CRS:

$$\mathsf{crs} \leftarrow (\tilde{\mathsf{ck}}, g_1, \tilde{g}_1, g_2, g_{2,\xi}, (g_{2,i}, \tilde{g}_{2,i})_{i\in\{\lambda_1,\ldots,\lambda_\xi,v,v+\xi\}}, (h_{2,i}, \tilde{h}_{2,i})_{i\in[1,n-\xi]}).$$

Алгоритм доказывающего $\pi_{\mathsf{rsft}} \leftarrow \mathsf{P}_{\mathsf{rsft}}(\mathsf{crs}, u_{\mathsf{rsft}} = (A, \tilde{A}, B, \hat{B}, \tilde{B}), w_{\mathsf{rsft}} = (\mathbf{a}, r_a, \mathbf{b}, r_b))$

Вычисляется и подаётся на выход доказательство сдвига вида $\mathbb{G}_2^2$:

$$\pi_{\mathsf{rsft}} \leftarrow (\pi, \tilde{\pi}) = \prod_{i=1}^{n-\xi} (h_{2,i}, \tilde{h}_{2,i})^{b_{i+\xi}} \prod_{i=1}^{\xi} (g_{2,\lambda_i}, \tilde{g}_{2,\lambda_i})^{-b_i} (g_{2,v+\xi}, \tilde{g}_{2,v+\xi})^{r_a} (g_{2,v}, \tilde{g}_{2,v})^{-r_b}.$$

Алгоритм верификатора $0/1 \leftarrow \mathsf{V}_{\mathsf{rsft}}(\mathsf{crs}, u_{\mathsf{rsft}}, \pi_{\mathsf{rsft}} = (\pi, \tilde{\pi}))$

Верификация подтверждается, если выполняются равенства

$$e(A, g_{2,\xi})/e(B, g_2) = e(g_1, \pi), \quad e(g_1, \tilde{\pi}) = e(\tilde{g}_1, \pi).$$

Для доказывающего наиболее трудоёмки $2\,O(n)$ произведений в $\mathbb{Z}_p$, $(2 + O(1)) \times \times \log_2 \beta \cdot n / \log_2 n + O(n)$ билинейных произведений, где $\beta < p$. Верификация выполняется пятью билинейными спариваниями. CRS состоит из $O(n)$ элементов группы.

4. Протокол Р. Дженнаро, С. Джентри, Б. Парно, М. Райковой

Работы [37, 42] представляет собой развитие результатов [31, 43–45], при этом задачи QSP/QAP формируются из логических/арифметических схем $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$, являющихся отображением с $(n+h)$ значениями поля $\mathbb{F}$ и l выходами. Отображение C является арифметической схемой, если выходы определяются входами, проходящими по рёбрам (проводам) к вершинам графа (двоичным вентилям) в виде операторов «$+$» или «$\times$» (вентили неизменны, схема — ациклический ориентированный граф). Допустимым заданием схемы C является набор $(a_1, \ldots, a_N) \in \mathbb{F}^N$, где $N = (n + h) + l$ — число всех входов и выходов, при которых $C(a_1, \ldots, a_{n+h}) = (a_{n+h+1}, \ldots, a_N)$. При этом целевой полином $t(x) = \prod\limits_{g \in M} (x - r_i)$, где $r_i \in \mathbb{F}$ — корни; $g \in M$ — мультипликативные вентили схемы.

Недостатком QSP является то, что они описывают логические схемы (один бит на выходе), в то время как QAP более удобно описывают арифметические схемы (набор битов на выходе) из вентилей сложения и умножения в виде уравнений по модулю порядка группы p. По сравнению с QSP для QAP требуется три набора полиномов, что приводит к более длинным доказательствам постоянного размера. Однако вычисления QAP более производительны, так как доказывающий выполняет криптографические операции над целыми элементами в кольце $F[x]$ вместо операций для каждого логического вентиля в случае QSP. Далее рассматривается протокол DV zk-SNARK на основе QAP [37, 42], которая работает с элементами поля и поэтому является более производительной версией побитовых QSP.

Протокол zk-SNARK с алгоритмами $(\mathsf{Gen}, \mathsf{Regen}_f, \mathsf{P}, \mathsf{V})$ для QAP использует $R = = \{(\mathbf{u}, \mathbf{w})\}$ — множество отношений над $\mathbb{F}^n$, $\mathbf{u}$ — открытый вход, $\mathbf{w}$ — секретное свидетельство. Входные индексы $i \in \{1, \ldots, n'\}$ соответствуют состоянию $\mathbf{u}$, а позиции $i \in \{n' + 1, \ldots, n\}$ — секретному свидетельству $\mathbf{w}$. Отношение R может проверяться с помощью арифметической схемы над $\mathbb{F}$, что эквивалентно существованию арифметической функции $f(\mathbf{u}, \mathbf{w}) = 1$ при $(\mathbf{u}, \mathbf{w}) \in R$. В данном случае рассматривается язык L, в котором $(\mathbf{x}, \mathbf{w}) \in R$ при $f(\mathbf{x}) = \mathbf{w}$. Для этого требуется модификация f, чтобы построить арифметическую схему ϕ, принимающую $(\mathbf{u}, \mathbf{w})$ в качестве входных данных, которая выводит 1 при $(\mathbf{u}, \mathbf{w}) \in R$ и 0 в противном случае и запускает протокол zk-SNARK с использованием QAP для ϕ. В свою очередь, ϕ запускает f с входом $\mathbf{u}$ (путём задействования арифметической схемы для f) для получения выходных значений f вида $(w'_{n'+1}, \ldots, w'_n)$. Затем вычисляются $(b_{n'+1}, \ldots, b_n) = (w'_{n'+1} - w_{n'+1}, \ldots, w'_n - w_n)$,

которые должны иметь нулевые значения, если ввод схемы $\mathbf{u}$ является удовлетворительным.

Для отношения R многочлены QAP представлены гомоморфным зашифрованием их оценок в некоторой секретной точке s над полем $\mathbb{F}$, например $\{\mathsf{E}(v_k(s))\}$. Для общедоступной верификации может использоваться возведение в степень внутри билинейной группы [31, 32] в виде $\mathsf{E}(v_k(s)) = g^{v_k(s)}$ без расшифрования. Для фиксированного верификатора применяется схема аддитивного гомоморфного шифрования типа Пайе [46] или RSA [47] в виде $\mathsf{E}(v_k(s)) = \mathsf{E}_{pk}(v_k(s))$ с хранением секретного ключа расшифрования sk. В дальнейшем для краткости протоколы zk-SNARK с общедоступной верификацией будем обозначать PV (Public Verifiable) zk-SNARK, а с фиксированным верификатором — DV (Designated Verifiable) zk-SNARK.

Алгоритм Gen **формирования CRS**

На вход подаются параметр защиты λ и верхняя граница d степени QAP. Для функций f вычисляется пара асимметричных ключей (pk, sk), например RSA. В мультипликативной группе поля выбираются случайные независимые $\alpha, s \in \mathbb{F}^*$ и выводятся

$$\mathsf{priv} = sk,\ \ \mathsf{crs} = (pk, \{\mathsf{E}(s^i)\}_{i\in[0,d]}, \{\mathsf{E}(\alpha s^i)\}_{i\in[0,d]}).$$

Для протоколов PV zk-SNARK секретный ключ $\mathsf{priv} = sk$ также не используется.

Специфичное для функции формирование CRS алгоритмом Regen

На вход подаются crs, функция f со строгим QAP Q_f размера m степени не выше d:

$$Q_f = (V, W, Y, t(x), I_{\text{free}}, I_{\text{labeled}} = \bigcup_{i\in[n], j\in\{0,1\}} I_{i,j}). \tag{7}$$

Также подаётся $n' \in [1, n]$, $I_{\text{in}} = \bigcup\limits_{i\in[1,n'],j\in[0,1]} I_{ij}$, $I_{\text{mid}} = I \setminus I_{\text{in}} = \{1, \ldots, m\} \setminus I_{\text{in}}$. В мультипликативном поле выбираются случайные независимые $\beta_v, \beta_w, \beta_y, \gamma \in F^*$ и главная ссылочная строка для f и n' принимает следующий вид:

$$\begin{gathered}\mathsf{crs}_f = (\mathsf{crs}, Q_f, n', \{\mathsf{E}(v_k(s))\}_{k\in I_{\text{mid}}}, \{\mathsf{E}(w_k(s))\}_{k\in[m]}, \{\mathsf{E}(y_k(s))\}_{k\in[m]}, \{\mathsf{E}(s^i)\}_{i\in[0,d]},\\ \{\mathsf{E}(\alpha v_k(s))\}_{k\in I_{\text{mid}}}, \{\mathsf{E}(\alpha w_k(s))\}_{k\in[m]}, \{\mathsf{E}(\alpha y_k(s))\}_{k\in[m]}, \{\mathsf{E}(\alpha s^i)\}_{i\in[0,d]},\\ \{\mathsf{E}(\beta_v v_k(s))\}_{k\in I_{\text{mid}}}, \{\mathsf{E}(\beta_w w_k(s))\}_{k\in[m]}, \{\mathsf{E}(\beta_y y_k(s))\}_{k\in[m]}),\\ \mathsf{shortcrs}_f = (\mathsf{priv}, \mathsf{E}(1), \mathsf{E}(\alpha), \mathsf{E}(\gamma), \mathsf{E}(\beta_v\gamma), \mathsf{E}(\beta_w\gamma), \mathsf{E}(\beta_y\gamma), \{\mathsf{E}(v_k(s))\}_{k\in I_{\text{in}}}, \mathsf{E}(w_0(s)), \mathsf{E}(t(s))).\end{gathered} \tag{8}$$

По сравнению с QSP в QAP Q_f (7) добавлен набор многочленов Y с соответствующими включениями $\{\mathsf{E}(y_k(s))\}_{k\in[m]}, \{\mathsf{E}(\alpha y_k(s))\}_{k\in[m]}, \{\mathsf{E}(\beta_y y_k(s))\}_{k\in[m]}), \mathsf{E}(\beta_y\gamma)$ в CRS.

Алгоритм доказывающего P

На вход подаются crs_f, состояние $\mathbf{u} \in \{0,1\}^{n'}$ и секретное свидетельство $\mathbf{w} \in \in \{0,1\}^{n-n'}$, необходимые для проверки отношения $(\mathbf{u}, \mathbf{w}) \in R$, т. е. выполнимости $f(\mathbf{u}, \mathbf{w}) = 1$. P оценивает Q_f, чтобы получить $(a_1, \ldots, a_m)$ и многочлен $h(x)$ для проверки равенства

$$h(x)t(x) = (v_0(x) + \sum_{k=1}^{m} a_k v_k(x))(w_0(x) + \sum_{k=1}^{m} a_k w_k(x)) - (y_0(x) + \sum_{k=1}^{m} a_k y_k(x)).$$

Для $v_{\text{mid}}(x) = \sum\limits_{k\in I_{\text{mid}}} a_k\, v_k(x)$, $w(x) = \sum\limits_{k\in[m]} a_k\, w_k(x)$ и $y(x) = \sum\limits_{k\in[m]} a_k\, y_k(x)$ алгоритм P выводит доказательство знания

$$\begin{gathered}\pi = (\pi_{v_{\text{mid}}}, \pi_w, \pi_y, \pi_h, \pi_{v'_{\text{mid}}}, \pi_{w'}, \pi_{y'}, \pi_{h'}, \pi_z) = (\mathsf{E}(v_{\text{mid}}(s)), \mathsf{E}(w(s)), (\mathsf{E}(y(s)), \mathsf{E}(h(s)),\\ \mathsf{E}(\alpha v_{\text{mid}}(s)), \mathsf{E}(\alpha w(s)), \mathsf{E}(\alpha y(s)), \mathsf{E}(\alpha h(s)), \mathsf{E}(\beta_v v_{\text{mid}}(s) + \beta_w w(s) + \beta_y y(s))).\end{gathered} \tag{9}$$

Аналогично доказательству версии QSP [37], для заданных crs_f и открытого входа $\mathbf{u} \in \{0,1\}^n$ элементы V_{mid}, W и Y, однажды зафиксированные в доказательстве π, определяют все остальные элементы $H, V'_{\mathrm{mid}}, W', Y', H', Z$, которые также кодируются в π. Поэтому при формировании и использовании элементов V_{mid}, W и Y необходимо соблюдать повышенные требования к их защите.

По сравнению с доказательством для QSP [37] случай QAP (9) содержит девять элементов поля вместо семи. Добавлены $\mathsf{E}(y(s)), \mathsf{E}(\alpha y(s))$ и «смещение» $\beta_y y(s)$.

Алгоритм верификатора V

На вход подаются $\mathsf{shortcrs}_f$, sk, $\mathbf{u}$ и $\pi = (\pi_{v_{\mathrm{mid}}}, \pi_w, \pi_y, \pi_h, \pi_{v'_{\mathrm{mid}}}, \pi_{w'}, \pi_{y'}, \pi_{h'}, \pi_z)$. V подтверждает, что используются достоверные зашифрованные элементы.

Для соответствующих зашифрованных значений $V_{\mathrm{mid}}, W, Y, H, V'_{\mathrm{mid}}, W', Y', H', Z$ алгоритм V вычисляет $\mathsf{E}(v_{\mathrm{in}}(s))$ для $v_{\mathrm{in}}(s) = \sum\limits_{k \in I_{\mathrm{in}}} a_k\, v_k(s)$. Используя вычисление квадратных корней на зашифрованных переменных строгого QAP, верификация проводится следующими равенствами:

$$\begin{gathered}(v_0(s) + v_{\mathrm{in}}(s) + V_{\mathrm{mid}})(w_0(s) + W) - (y_0(s) + Y) = H \cdot t(s),\ V'_{\mathrm{mid}} = \alpha V_{\mathrm{mid}},\\ W' = \alpha W,\ Y' = \alpha Y,\ H' = \alpha H,\ \gamma Z = (\beta_v \gamma) V_{\mathrm{mid}} + (\beta_w \gamma) W + (\beta_y \gamma) Y.\end{gathered} \tag{10}$$

По сравнению с верификационными равенствами QSP [37] случай QAP (10) дополнительно учитывает значение $-(y_0(s)+Y)$ для условия делимости, контролирует $Y' = \alpha Y$ и проверяет корректность использования введённого многочлена y для нового фрагмента доказательства $\pi_z = Z$.

Нулевое разглашение

Свойство ZK обеспечивается добавлением в crs_f значений $\mathsf{E}(t(s)), \mathsf{E}(\alpha t(s)), \mathsf{E}(\beta_v t(s)), \mathsf{E}(\beta_w t(s)), \mathsf{E}(\beta_y t(s)), \mathsf{E}(v_0(s)), \mathsf{E}(\alpha v_0(s)), \mathsf{E}(w_0(s)), \mathsf{E}(\alpha w_0(s)), \mathsf{E}(y_0(s)), \mathsf{E}(\alpha y_0(s))$. Выполняется сдвиг значений $\mathsf{E}(v_{\mathrm{mid}}(s)), \mathsf{E}(w(s)), E(y(s))$ случайным образом, кратным $\mathsf{E}(t(s))$, другие значения изменяются соответственно. Таким образом, значения $\{ak\}_k \in I_{\mathrm{in}}$, $v_{\mathrm{in}}(x)$, $v_{\mathrm{mid}}(x)$, $v(x) = v_0(x) + v_{\mathrm{in}}(x) + v_{\mathrm{mid}}(x)$, $w(x)$, $y(x)$, $h(x)$ остаются прежними. Верификатор выбирает случайные $\delta_{v_{\mathrm{mid}}}, \delta_w, \delta_y \in F$, в результате чего P строит доказательство со свойством ZK:

$$\begin{gathered}\pi = (\pi'_{v_{\mathrm{mid}}}, \pi'_w, , \pi'_y, \pi'_h, \pi'_{v'_{\mathrm{mid}}}, \pi'_{w'}, \pi'_{y'}, \pi'_{h'}, \pi'_z) = (\mathsf{E}(v'_{\mathrm{mid}}(s)), \mathsf{E}(w'(s)), \mathsf{E}(y'(s)), \mathsf{E}(h'(s)),\\ \mathsf{E}(\alpha v'_{\mathrm{mid}}(s)), \mathsf{E}(\alpha w'(s)), \mathsf{E}(\alpha y'(s)), \mathsf{E}(\alpha h'(s)), \mathsf{E}(\beta_v v'_{\mathrm{mid}}(s) + \beta_w w'(s) + \beta_y y'_{\mathrm{mid}}(s))),\end{gathered}$$

где $v'_{\mathrm{mid}}(x) = v_{\mathrm{mid}}(x) + \delta_{v_{\mathrm{mid}}} t(x)$; $w'(x) = w(x) + \delta_w t(x)$; $y'(x) = y(x) + \delta_y t(x)$. Новый частный многочлен со значениями $v'(x) = v_0(x) + v_{\mathrm{in}}(x) + v'_{\mathrm{mid}}(x) = v''(x) + \delta_{v_{\mathrm{mid}}} t(x)$, $v''(x) = v_0(x) + v_{\mathrm{in}}(x) + v_{\mathrm{mid}}(x)$ и соответствующими $w'(x), w''(x), y'(x), y''(x)$ принимает следующий вид:

$$\begin{gathered}h'(x) = (v'(x)w'(x) - y'(x))/t(x) =\\ = ((v''(x) + \delta_{v_{\mathrm{mid}}} t(x))(w''(x) + \delta_w t(x)) - (y''(x) + \delta_y t(x)))/t(x) =\\ = h(x) + \delta_{v_{\mathrm{mid}}} w''(x) + \delta_w v''(x) + \delta_{v_{\mathrm{mid}}} \delta_w t(x) - \delta_y.\end{gathered} \tag{11}$$

По сравнению с частным многочленом QSP [37] случай QAP (11) отличается только составляющей $-\delta_y$.

Согласно [48], протокол zk-SNARK на основе QSP и QAP может повторно рандомизироваться не только исходным доказывающим, но и другими произвольными участниками протокола.

5. Протокол Б. Парно, Дж. Хауэлла, С. Джентри, М. Райковой

Протокол VC Б. Парно, Дж. Хауэлла, С. Джентри и М. Райковой [49] вносит изменения в вариант [34, 37]. Теперь сторонний работник проводит делегированные вычисления над публичным входом $\mathbf{u}$ в открытом виде. Исходный и более производительный протоколы представлены в [49]. По соображениям возможного практического применения ниже рассматривается только более производительная версия VC.

Алгоритм формирования ключей KeyGen

На вход подаются параметр защиты λ, функция F с N входными/выходными значениями из $\mathbb{F}$, для которой строится арифметическая схема C. Затем для C формируется QAP (7) размера m и степени d. Как и ранее, индексы $I_{\text{mid}} = \{N+1, \ldots, m\}$ рассматриваются как не связанные с входами/выходами. С использованием случайных секретов $r_v, r_w, s, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \in \mathbb{F}$ устанавливаются $r_y = r_w r_v, g_v = g^{r_v}, g_w = g^{r_w}, g_y = g^{r_y}$. На выходе формируются ключи публичной оценки EK_F и верификации VK_F:

$$(EK_F, VK_F) \leftarrow \mathsf{KeyGen}(F, 1^\lambda).$$

Ключи публичной оценки EK_F и верификации VK_F принимают следующий вид:

$$\begin{gathered} EK_F = \{g_v^{v_k(s)}\}_{k\in I_{\text{mid}}},\ \{g_w^{w_k(s)}\}_{k\in I_{\text{mid}}},\ \{g_y^{y_k(s)}\}_{k\in I_{\text{mid}}},\ \{g_v^{\alpha_v v_k(s)}\}_{k\in I_{\text{mid}}}, \\ \{g_w^{\alpha_w w_k(s)}\}_{k\in I_{\text{mid}}}, \{g_y^{\alpha_y y_k(s)}\}_{k\in I_{\text{mid}}},\ \{g^{s^i}\}_{i\in[d]},\ \{g_v^{\beta v_k(s)} g_w^{\beta w_k(s)} g_y^{\beta y_k(s)}\}_{k\in I_{\text{mid}}}; \\ VK_F = \big(g^1, g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma}, g^{\beta\gamma}, g_y^{t(s)}, \{g_v^{v_k(s)}, g_w^{w_k(s)}, g_y^{y_k(s)}\}_{k\in\{0\}\cup[N]}\big). \end{gathered} \tag{12}$$

По сравнению с EK_F исходного протокола [49], соответствующего $\mathbf{crs}_f$ (8), модифицированный вариант (12) исключает вычисление $g^{\alpha h(s)}$ и $g_{i\in[d]}^{\alpha s^i}$, сокращая нагрузку на исполнителя.

Алгоритм исполнителя Compute

Алгоритм оценивает схему C функции F на входе $\mathbf{u}$ для вычисления $y = F(\mathbf{u})$. Результатом оценки является знание работником значений $\{c_i\}_{i\in[m]}$ проводов схемы:

$$(y, \pi_y) \leftarrow \mathsf{Compute}(EK_F, \mathbf{u}).$$

Исполнитель выводит $h(x)$ для проверки $p(x) = h(x)t(x)$ и на основе $v_{\text{mid}}(s) = \sum_{k\in I_{\text{mid}}} c_k v_k(x)$, а также соответствующих $w_{\text{mid}}(s), y_{\text{mid}}(s)$, вычисляет доказательство π_y, отличное от (9):

$$\begin{gathered} \pi_y = (g_v^{v_{\text{mid}}(s)}, g_w^{w_{\text{mid}}(s)}, g_y^{y_{\text{mid}}(s)}, g^{h(s)}, g_v^{\alpha_v v_{\text{mid}}(s)}, g_w^{\alpha_w w_{\text{mid}}(s)}, g_y^{\alpha_y y_{\text{mid}}(s)}, \\ g_v^{\beta v_{\text{mid}}(s)} \cdot g_w^{\beta w_{\text{mid}}(s)} \cdot g_y^{\beta y_{\text{mid}}(s)}). \end{gathered} \tag{13}$$

Доказательство текущего протокола (13) по сравнению с исходным (9) сокращено с девяти до восьми элементов группы. Тем не менее работы по исчерпывающему обоснованию надёжности модифицированного протокола отсутствуют.

Алгоритм верификации вычислений Verify

Приведём формальную запись функции верификации:

$$0/1 \leftarrow \mathsf{Verify}(VK_F, \mathbf{u}, \mathbf{y}, \pi_y).$$

Проверка доказательства π_y с элементами $g^{V_{\text{mid}}}, g^{W_{\text{mid}}}, g^{Y_{\text{mid}}}, g^H, g^{V'_{\text{mid}}}, g^{W'_{\text{mid}}}, g^{Y'_{\text{mid}}}, g^Z$ осуществляется на основе ключа верификации VK_F и билинейного спаривания e.

Контроль делимости QAP использует VK_F для вычисления $g_v^{v_{io}(s)} = \prod\limits_{k\in[N]} (g_v^{v_k(s)})^{c_k}$, $g_w^{w_{io}(s)} = \prod\limits_{k\in[N]} (g_w^{w_k(s)})^{c_k}$ и $g_y^{y_{io}(s)} = \prod\limits_{k\in[N]} (g_y^{y_k(s)})^{c_k}$:

$$e(g_v^{v_0(s)} g_v^{v_{io}(s)} g_v^{V_{\text{mid}}}, g_w^{w_0(s)} g_w^{w_{io}(s)} g_w^{W_{\text{mid}}}) = e(g_y^{t(s)}, g^H) e(g_y^{y_0(s)} g_y^{y_{io}(s)} g_y^{Y_{\text{mid}}}, g).$$

Затем выполняется проверка нахождения полиномов множеств V, W, Y в соответствующих диапазонах:

$$e(g_v^{V'_{\text{mid}}}, g) = e(g_v^{V_{\text{mid}}}, g^{\alpha_v}),\ e(g_w^{W'_{\text{mid}}}, g) = e(g_w^{W_{\text{mid}}}, g^{\alpha_w}),\ e(g_y^{Y'_{\text{mid}}}, g) = e(g_y^{Y_{\text{mid}}}, g^{\alpha_y}). \quad (14)$$

В заключение проверяется, что в (14) использовались одинаковые коэффициенты в каждой из линейных комбинаций V, W, Y:

$$e(g^Z, g^\gamma) = e(g^{V_{\text{mid}}} g^{W_{\text{mid}}} g^{Y_{\text{mid}}}, g^{\beta\gamma}).$$

Нулевое разглашение Используя принципы [37], рабочий выбирает случайные $\delta_v, \delta_w, \delta_y \in \mathbb{F}$ и маскирует многочлены в виде $v_{\text{mid}}(x) + \delta_v t(x)$, $v(x) + \delta_v t(x)$, $w(x) + {}+\delta_w t(x)$, $y(x)+\delta_v t(x)$. Для упрощения рандомизации доказательства в ключ публичной оценки EK_F (12) включаются следующие величины:

$$g_v^{\alpha_v t(s)}, g_w^{\alpha_w t(s)}, g_y^{\alpha_y t(s)}, g_v^{\beta t(s)}, g_w^{\beta t(s)}, g_y^{\beta t(s)}.$$

6. Протокол Х. Липмаа «с обязательством и доказательством»

Как правило, протоколы zk-SNARK строят CRS для одного вида NP-языка, например для задачи выполнимости схем с использованием QAP, SAP, QSP, SSP [37, 50, 51] и др. Протокол zk-SNARK Х. Липмаа [52] за счёт новой схемы обязательств строит CRS, которая может использоваться повторно и не зависит от вида NP-языка.

Согласно модели CRS [53], в [52] строится схема обязательств с возможностью извлечения связываемых (скрываемых) данных (Trapdoor Commitment Scheme), не зависящая от бинарного отношения R, состоящая из алгоритма G_{com} (строит CRS ck и ключ-«лазейку») и C (по ck, сообщению m и рандомизатору r выдаёт обязательство $\mathsf{C}(\mathsf{ck}, m, r)$), что обуславливает их название — протоколы zk-SNARK с обязательством и доказательством (Commit-And-Prove zk-SNARK, CaP zk-SNARK). Связывание значения w_i вида $u_i = \mathsf{C}(\mathsf{ck}, w_i, r_i)$ доказывает, что набор $u = (u_{i_j}, w_{i_j}, r_{i_j})_{j=1}^{l_m(n)}$ для публично известных индексов i_j удовлетворяет тому, что u_{i_j} является обязательством для w_{i_j} с рандомизатором r_{i_j} и $(w_{i_j}) \in R$ (n — размер входа $\mathbb{Z}_p^n$, $l_m(n)$ — некоторый полином).

Вводится новая схема обязательств с «лазейкой» и возможностью извлечения данных, в которой n — степень 2; p — порядок группы; ω — n-й примитивный корень по модулю p. Определяются многочлены

$$\begin{aligned} f_0(X) = Z(X) &= \prod_{i=1}^{n} (X - \omega^{i-1}) = X^n - 1, \\ f_i(X) = l_i(X) &= \prod_{j\neq i} ((X - \omega^{j-1})/(\omega^{i-1} - \omega^{j-1})), \end{aligned} \quad (15)$$

где $Z(\omega^{i-1}) = 0$, а $l_i(X)$ — i-й базисный полином Лагранжа, т. е. $l_i(\omega^{i-1}) = 1$, $l_i(\omega^{j-1}) = 0$ при $i \neq j$. С использованием значений $(\mathsf{gk}, \chi, \gamma)$ и полиномов (15) вычисляется CRS

$$\mathsf{ck} = (\mathsf{gk}, (g_1, g_2^\gamma)^{f(\chi)})_{f\in\{Z, l_1, \ldots, l_n\}}$$

и определяется схема обязательств с «лазейкой»:

$$\begin{gathered}\mathrm{Com}(\mathsf{ck},(a_1,\ldots,a_k),r)=\prod_{i=1}^{k}((g_1,g_2^{\alpha})^{l_i(\sigma)})^{a_i}((g_1,g_2^{\alpha})^{Z(\sigma)})^{r}=\\ =((g_1,g_2^{\alpha})^{rZ(\sigma)+\sum\limits_{i=1}^{k}a_il_i(\sigma)}=(A_1,A_2^{\alpha})=(A,\hat{A})\in\mathbb{G}_1\times\mathbb{G}_1.\end{gathered}\tag{16}$$

Примечательно, что $\mathrm{Com}(\mathsf{ck},\mathbf{1}_n,0)=(g_1,g_2^{\gamma})$. Подлинность обязательства (16) контролируется равенством $e(A_1,g_2^{\gamma Z(\chi)})=e(g_1^{Z(\chi)},A_2^{\gamma})$, раскрытие обязательства требует знания $\mathbf{a},r$. С учётом использования схемы обязательств (16) рассматриваемый протокол zk-SNARK [52] совпадает с вариантом Й. Грот [31], где доказывающий подтверждает возможность раскрытия обязательств $(A,A^{\gamma}),(B,B^{\gamma}),(C,C^{\gamma})$ для векторов $\mathbf{a},\mathbf{b},\mathbf{c}$. Таким образом, используется бинарное отношение

$$R^{\times}_{\mathsf{ck},n}=\{(u_{\times},w_{\times},r_{\times}):u_{\times}=((A_1,A_2^{\gamma}),(B_1,B_2^{\gamma}),(C_1,C_2^{\gamma})),\ w_{\times}=(\mathbf{a},\mathbf{b},\mathbf{c}),\ r_{\times}=(r_a,r_b,r_c),$$
$$(A_1,A_2^{\gamma})=\mathrm{Com}(\mathsf{ck},\mathbf{a},r_a),\ (B_1,B_2^{\gamma})=\mathrm{Com}(\mathsf{ck},\mathbf{b},r_b),\ (C_1,C_2^{\gamma})=\mathrm{Com}(\mathsf{ck},\mathbf{c},r_c),\ \mathbf{a}\circ\mathbf{b}=\mathbf{c}\}.$$

Для формирования протокола zk-SNARK на основе произведения [52] рассматриваются полиномы $A(X),B(X),C(X)$ при $A(\omega^{i-1})=a_i$, $B(\omega^{i-1})=b_i$, $C(\omega^{i-1})=c_i$ для $i\in[n]$. Пусть $Q(X)=A(X)B(X)-C(X)$, $A(X),B(X),C(X)$ входят в $\{l_i(X)\}_{i=1}^{n}$ и $\pi(X)=Q(X)/Z(X)$. В таком случае выполняется требуемое равенство: $\mathbf{a}\circ\mathbf{b}=\mathbf{c}$. Используется следующая тройка алгоритмов.

Алгоритм формирования ключей $\mathsf{crs}_{\times}\leftarrow\mathsf{G}_{\times}(1^k,n)$

1. Выводятся параметры группы gk и случайные значения $(g_1,g_2,\chi,\gamma)\in\mathbb{G}_1^*\times\mathbb{G}_2^*\times\times\mathbb{Z}_p^2$ при $Z(\chi)\neq0$, $\gamma\neq0$.
2. В соответствии с (15) на основе базиса полиномов $F_C=(Z(X),(l_i(X))_{i=1}^{n})$ устанавливаются компоненты CRS:

$$\mathsf{crs}_P=\mathsf{ck}=(\mathsf{gk},(g_1,g_2^{\gamma})^{F_C(\chi)}),\quad \mathsf{crs}_V=(\mathsf{gk},g_2^{\gamma Z(\chi)}),\quad \mathsf{crs}_{\times}=(\mathsf{crs}_P,\mathsf{crs}_V).\tag{17}$$

Вход $u_{\times}=((A_1,A_2^{\gamma}),(B_1,B_2^{\gamma}),(C_1,C_2^{\gamma}))$ является общим и используется доказывающим и верификатором.

Алгоритм доказывающего $\pi_{\times}\leftarrow\mathsf{P}_{\times}(\mathsf{crs}_P,u_{\times},w_{\times}=(\mathbf{a},\mathbf{b},\mathbf{c}),r_{\times}=(r_a,r_b,r_c))$

Конфиденциальность обеспечивается случайными значениями $r_a,r_b,r_c\in\mathbb{Z}_p$. Определяется полином $Q_{wi}(X)=(L_{\mathbf{a}}(X)+r_aZ(X))L_{\mathbf{b}}(X)+r_bZ(X))-(L_{\mathbf{c}}(X)+r_cZ(X))$, где дополнения вида $rZ(X)$ гарантируют скрытие; $Q_{wi}(X)$ имеет степень $2n$ и делится на $Z(X)$, если $\mathbf{c}=\mathbf{a}\circ\mathbf{b}$:

1. Формируется компонент доказательства $\pi_{wi}(X)=Q_{wi}(X)/Z(X)=\sum\limits_{i=0}^{n}\pi_iX^i$ степени n.
2. Для сокращения сообщений проверяющий передаёт оценку $\pi_{wi}(X)$ в случайной секретной точке χ. Доказывающий вычисляет $\pi_{\times}=g_1^{\pi_{wi}(\chi)}$ с использованием значения $g_1^{\chi^i}$ из CRS (17) и коэффициентов π_i:

$$\pi_{\times}=g^{\pi_{wi}(\chi)}=\prod_{i=0}^{n}(g_1^{\chi^i})^{\pi_i}.$$

Алгоритм верификатора $0/1\leftarrow\mathsf{V}_{\times}(\mathsf{crs}_V,u_{\times},\pi_{\times})$

Верификация подтверждается, если выполняется равенство

$$e(A_1,B_2^{\gamma})=e(g_1,C_2^{\gamma})e(\pi_{\times},g_2^{\gamma Z(\chi)}).$$

Для доказывающего наиболее трудоёмка процедура $(n+1)$ возведений в степень в $\mathbb{G}_1$, три полиномиальных оценки, одно произведение и одно деление полиномов. Верификатор выполняет три билинейных спаривания. Исключая gk, CRS доказывающего и ck состоят из $2(n+1)$ элементов группы, CRS верификатора — из одного элемента группы. С использованием алгоритма [54] вычисление CRS составляет $O(n)$ операций.

7. Протокол Х. Липмаа на основе циклического сдвига векторов

Х. Липмаа [52] предлагает реконструкцию протокола zk-SNARK на основе сдвига [41] с использованием обязательств (16), где доказывающий подтверждает возможность раскрытия таких обязательств $(A, A^\gamma) = \mathrm{Com}(\mathsf{ck}, \mathbf{a}, r_a)$ и $(B, B^\gamma) = \mathrm{Com}(\mathsf{ck}, \mathbf{b}, r_b)$, что $\mathbf{a} = \mathbf{b} \gg z$, т. е. $a_i = b_{i+z}$ при $i \in \{1, \ldots, n-z\}$ и $a_i = 0$ при $i \in \{n-z+1, \ldots, n\}$. В этом случае бинарное отношение имеет вид

$$R^{\mathsf{rsft}}_{\mathsf{ck},n} = \{(u_\times, w_\times, r_\times) : u_\times = ((A_1, A_2^\gamma), (B_1, B_2^\gamma),\ w_\times = (\mathbf{a}, \mathbf{b}),\ r_\times = (r_a, r_b),$$
$$(A_1, A_2^\gamma) = \mathrm{Com}(\mathsf{ck}, \mathbf{a}, r_a),\ (B_1, B_2^\gamma) = \mathrm{Com}(\mathsf{ck}, \mathbf{b}, r_b),\ \mathbf{a} = \mathbf{b} \gg z.$$

Полученный после реконструкции протокол zk-SNARK с правым сдвигом представляет собой следующие алгоритмы.

Алгоритм формирования ключей $\mathsf{crs}_{\mathsf{rsft}} \leftarrow \mathsf{G}_{\mathsf{rsft}}(1^k, n)$

1. Фиксируется полином $Z^*(X) = Z(X)^2$, выводятся параметры группы gk и случайные значения $(g_1, g_2, \chi, \gamma, \delta) \in \mathbb{G}_1^* \times \mathbb{G}_2^* \times \mathbb{Z}_p^3$ при $Z(\chi) \neq 0$, $\gamma \neq 0$.
2. В соответствии с (15) на основе базиса полиномов $F_C = (Z(X), (l_i(X))_{i=1}^n)$ устанавливаются компоненты CRS:

$$\mathsf{ck} = (\mathsf{gk}, (g_1, g_2^\gamma)^{F_C(\chi)}),\ \mathsf{crs}_P = (\mathsf{gk}, (g_1, g_2^\delta)^{F_{z-\mathsf{rsft}}(\chi)}),\ \mathsf{crs}_V = (\mathsf{gk}, (g_1, g_2^\delta)^{Z(\chi)}), g_2^{\delta Z(\chi) Z^*(\chi)}).$$

3. На выход подаётся $\mathsf{crs}_{\mathsf{rsft}} = (\mathsf{ck}, \mathsf{crs}_P, \mathsf{crs}_V)$.

Вход $u_{\mathsf{rsft}} = ((A_1, A_2^\gamma), (B_1, B_2^\gamma))$ является общим и используется доказывающим и верификатором.

Алгоритм доказывающего $\pi_{\mathsf{rsft}} \leftarrow \mathsf{P}_{\mathsf{rsft}}(\mathsf{crs}_P, u_{\mathsf{rsft}}, w_{\mathsf{rsft}} = (\mathbf{a}, \mathbf{b}), r_{\mathsf{rsft}} = (r_a, r_b))$

1. Выводятся случайные секретные значения $\chi, \delta \in \mathbb{Z}_p$.
2. Вычисляется и подаётся на выход доказательство сдвига:

$$\pi_{\mathsf{rsft}} = (\pi_1, \pi_2^\delta) = (g_1, g_2^\delta)^{\pi(\chi)} = \prod_{i=z+1}^{n} ((g_1, g_2^\delta)^{l_{i-z}(\chi) Z^*(\chi) - l_i(\chi)})^{b_i} \times$$
$$\times \prod_{i=1}^{z} ((g_1, g_2^\delta)^{l_i(\chi)})^{-b_i} ((g_1, g_2^\delta)^{Z(\chi) Z^*(\chi)})^{r_a} ((g_1, g_2^\delta)^{Z(\chi)})^{-r_b}.$$

Алгоритм верификатора $0/1 \leftarrow \mathsf{V}_{\mathsf{rsft}}(\mathsf{crs}_V, u_{\mathsf{rsft}}, \pi_{\mathsf{rsft}} = (\pi_1, \pi_2^\delta))$

Верификация подтверждается, если выполняются равенства

$$e(\pi_1, g_2^{\delta Z(\chi)}) = e(g_1^{Z(\chi)}, \pi_2^\gamma),\quad e(B_1 \pi_1, g_2^{\delta Z(\chi)}) = e(A_1, g_2^{\delta Z(\chi) Z^*(\chi)}).$$

Для доказывающего наиболее трудоёмки две процедуры по $(n+2)$ возведений в степень (одна в $\mathbb{G}_1$ и одна в $\mathbb{G}_2$). Объём передачи данных составляет два элемента группы. Исключая gk, CRS доказывающего и ck состоят из $4n+6$ элементов группы, CRS верификатора — из двух элементов группы. Верификатор выполняет четыре билинейных спаривания.

8. Протокол А. Э. Косба, Д. Пападопулоса, С. Папаманту и др.

Система VC «TRUESET» А. Э. Косба, Д. Пападопулоса, С. Папаманту и др. [55] представляет собой расширение реализации Pinocchio [49] на языке C++ с использованием библиотеки NTL [56, 57], обеспечивающей производительную полиномиальную арифметику на основе БПФ, библиотеки GMP [58] и библиотеки Ate-спаривания над кривыми Баррето — Нерига [59]. Система VC «TRUESET» способна достоверно вычислять любую функцию полиномиального времени с полным набором логики, т. е. выраженную в виде схемы элементов множества объединения, пересечения, суммы и разности, а также гибридные схемы из булевых и арифметических вентилей. Обеспечивается зависимость времени работы доказывающего и формирования ключа, пропорциональная размеру ввода. Это достигается путём кодирования набора мощности c в виде многочлена степени c, аналогично [60, 61], а набора схем — в виде полиномиальной схемы, где провода являются полиномами, а вентили выполняют полиномиальное сложение/умножение.

Для этого вводятся QPP, в которых используется полиномиальное умножение/сложение и многочлены сводятся к простым значениям проводов арифметической схемы за счёт их оценки в секретной точке s. В этом случае полиномиальная схема $\mathcal{F}$ над полем $\mathbb{F}$ представляет схему с вентилями сложения и умножения полиномов, где d — количество вентилей умножения; N — количество входных и выходных проводов; $I_m = \{N+1, \ldots, m\}$ — индексы внутренних проводов; n_i — степень полинома провода i; n — наивысшая степень полиномов проводов.

Для множества $A = \{a_1, a_2, \ldots, a_n\} \in \mathbb{F}^n$ определяется характеристический полином $A(z) = (z + a_1) \ldots (z + a_n)$. Операции над множествами преобразуются в набор схем по следующим формулам. Множество пересечения: $I = A \cap B$, если существуют такие полиномы $\alpha(z), \beta(z), \gamma(z), \delta(z)$, что $\alpha(z)A(z) + \beta(z)B(z) = I(z)$, $\gamma(z)I(z) = A(z)$, $\delta(z)I(z) = B(z)$. Множество объединения: $U = A \cup B$, если также существует такой полином $i(z)$, что $\alpha(z)A(z) + \beta(z)B(z) = i(z)$, $\gamma(z)i(z) = A(z)$, $\delta(z)i(z) = B(z)$, $\delta(z)A(z) = U(z)$. Множество разности: $D = A - B$, если выполняются равенства $\alpha(z)A(z) + \beta(z)B(z) = i(z)$, $D(z)i(z) = A(z)$, $\delta(z)i(z) = B(z)$. В результате схема C с N входами, d_1 вентилями пересечения и d_2 вентилями объединения преобразуется в полиномиальную схему $\mathcal{F}$ с $4d_1 + 5d_2$ элементами умножения. Данный подход является преобразованием из QAP в QPP с возможностью выбора уровней абстракции в виде операций над множествами или арифметическими/битовыми операциями для разных частей схемы.

Алгоритм формирования ключей $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathcal{F}, 1^k)$

Используется полиномиальная схема $\mathcal{F}$ и соответствующая QPP $\mathcal{Q} = (V, W, Y, \tau(x))$, билинейное спаривание $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, порядок групп равен p, порождающий элемент группы g. Фиксируются случайные значения $s, t, r_v, r_w, \alpha_v, \alpha_w, \alpha_y, \beta, \gamma \in \mathbb{Z}_p$, устанавливаются $r_y = r_v r_w$, $g_v = g^{r_v}$, $g_w = g^{r_w}$, $g_y = g^{r_y}$. Ключи оценки $EK_{\mathcal{F}}$ и верификации $VK_{\mathcal{F}}$ являются ключом доказательства pk для протокола zk-SNARK. В данном случае протокол является PV zk-SNARK, так как $\mathsf{sk} = \mathsf{pk}$. Ключи принимают следующий вид:

$$EK_{\mathcal{F}} = \{g_v^{t^i v_k(s)}, g_w^{t^i w_k(s)}, g_y^{t^i y_k(s)}\}_{(i,k) \in [n] \times I_m}, \quad \{g_v^{t^i \alpha_v v_k(s)}, g_w^{t^i \alpha_w w_k(s)}, g_y^{t^i \alpha_y y_k(s)}\}_{(i,k) \in [n] \times I_m},$$
$$\{g_v^{t^i \beta v_k(s)}, g_w^{t^i \beta w_k(s)}, g_y^{t^i \beta y_k(s)}\}_{(i,k) \in [n] \times I_m}, \quad \{g^{t^i s^j}\}_{(i,j) \in [2n] \times [d]};$$
$$VK_{\mathcal{F}} = (g^1, g^{\alpha_v}, g^{\alpha_w}, g^{\alpha_y}, g^{\gamma}, g^{\beta\gamma}, g_y^{t(s)}, \{g_v^{t^i v_k(s)}, g_w^{t^i w_k(s)}, g_y^{t^i y_k(s)}\}_{(i,k) \in [n] \times [N]}).$$

Таким образом, KeyGen строит обязательство цепи $\mathcal{F}$ путём вывода элементов, относящихся к внутреннему набору проводов I_m для QPP $Q = (V, W, Y, \tau(x))$. Полиномы оцениваются в случайно выбранных точках t и s.

Сложность формирования ключей составляет $O(n|I_m| + nd + nN) = O(dn)$. Вычисления доказывающего составляют $O(T + dv\log(dv) + mdv)$, где T — время вычисления многочленов $c_i(z)$; v — максимальная степень полиномов проводов. Верификация имеет сложность $O\left(\sum\limits_{i\in[N]} n_i\right)$.

Алгоритм доказывающего $\pi \leftarrow \mathsf{P}(\mathsf{pk}, x, w)$

Вход x содержит входные u и выходные y полиномы, секретное свидетельство w включает назначения полиномов внутренних проводов схемы $\mathcal{F}$. Значения $c_k(z)$ являются полиномами проводов цепи, для которых выполняется равенство $y = \mathcal{F}(u, w)$, а $h(x, z)$ — полином-частное QPP для равенства $p(x, z) = h(x, z)\tau(x)$. Доказательство π включает вычисление набора переменных для свойства извлекаемости, набора переменных проверки согласованности цепи — непротиворечивости назначения проводов (могут вычисляться из публичного ключа $\{g_v^{t^i\beta v_k(s)}, g_w^{t^i\beta w_k(s)}, g_y^{t^i\beta y_k(s)}\}_{(i,k)\in[n]\times I_m}$), а также значения $g^{h(s,t)}$ для контроля свойства делимости:

$$\pi = \Big((g_v^{v_m(s,t)}, g_w^{w_m(s,t)}, g_y^{y_m(s,t)}, g_v^{\alpha_v v_m(s,t)}, g_w^{\alpha_w w_m(s,t)}, g_y^{\alpha_y y_m(s,t)}), (g_v^{\beta v_m(s,t)}, g_w^{\beta w_m(s,t)}, g_y^{\beta y_m(s,t)}),$$
$$\big(g^{h(s,t)} : v_m(x,z) = \sum_{k\in I_m} c_k(z)v_k(x),\ w_m(x,z) = \sum_{k\in I_m} c_k(z)w_k(x),$$
$$y_m(x,z) = \sum_{k\in I_m} c_k(z)y_k(x)\big)\Big). \tag{18}$$

Таким образом, чтобы доказать допустимость назначения входных/выходных проводов $c_1(z), \ldots, c_N(z)$, достаточно доказать существование допустимых полиномов $c_{N+1}(z), \ldots, c_m(z)$ для соответствующих внутренних проводов. При этом многочлен $p(x, z)$ должен иметь корни $r_1, r_2, \ldots, r_d$. Для этого доказывающий вычисляет корректные полиномы-назначения $c_1(z), c_2(z), \ldots, c_m(z)$, которые используются с ключом $EK_{\mathcal{F}}$ для вычисления компонентов π (18) на основе полиномов внутренних проводов цепи $v_m(x, z), w_m(x, z), y_m(x, z)$.

Алгоритм верификации $0/1 \leftarrow \mathsf{Verify}(\mathsf{pk}, x, \pi)$

Верификатор последовательно разделяет доказательство π (18) на компоненты $((\gamma_v, \gamma_w, \gamma_y, k_v, k_w, k_y), \Lambda, \gamma_h)$. Верификация подтверждается, если выполняются проверки на основе α-переменных, свойства полиномиальной делимости для $\lambda_v = g^{\sum\limits_{k\in[N]} c_k(t)v_k(s)}$, $\lambda_w = g^{\sum\limits_{k\in[N]} c_k(t)w_k(s)}$, $\lambda_y = g^{\sum\limits_{k\in[N]} c_k(t)y_k(s)}$, а также на основе β-переменных:

$$e(\gamma_v, g^{\alpha_v}) = e(k_v, g),\ e(\gamma_w, g^{\alpha_w}) = e(k_w, g),\ e(\gamma_y, g^{\alpha_y}) = e(k_y, g),$$
$$e(\lambda_v\gamma_v, \lambda_w\gamma_w)/e(\lambda_y\gamma_y, g) = e(\gamma_h, g^{\tau(s)}), e(\gamma_v\gamma_w\gamma_y, g^{\beta\gamma}) = e(\Lambda, g^{\gamma}).$$

По сравнению с VC для арифметических схем [49], время работы доказывающего «TRUESET» [55] сокращается в 30–150 раз для произвольных входов.

9. Протокол Г. Данезиса, К. Фурне, Й. Грота, М. Кольвейса

Протокол zk-SNARK Г. Данезиса, К. Фурне, Й. Грота, М. Кольвейса [51] строится для удовлетворения выполнимости булевых схем с l_u-разрядным публичным входом и l_w-разрядным секретным свидетельством. Используется бинарное отношение R, зависящее от параметра защиты λ, которое фиксирует пары $(u, w) \in \{0,1\}^{l_u(\lambda)} \times \{0,1\}^{l_w(\lambda)}$,

выводимые из схем с $m(\lambda)$ проводами и $n(\lambda)$ вентилями общего размера $d(\lambda) = m(\lambda) + n(\lambda)$. Более простая квадратичная форма SSP требует для проверки один полином вместо двух полиномов QSP и трёх полиномов QAP, что приводит к более простой и компактной предварительной настройке, меньшим размерам ключей и количеству операций для доказательства и верификации.

Алгоритм формирования ключей $(\sigma, \tau) \leftarrow \mathsf{Setup}(1^\lambda, R)$

На вход подаются публичные параметры gk и многочлены SSP вида $(\mathbf{v}, t)$ и выполняются следующие шаги:

1. Вычисляется публичный параметр $gk = (r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, формируемый на основе параметра защиты λ, где r — простой порядок групп; $\mathbb{G}_1, \mathbb{G}_2$ — мультипликативные циклические группы порядка r; $\mathbb{G}_T$ — группа порядка r; $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ — билинейное спаривание.
2. На основе R формируется булева схема $C_R : \{0,1\}^{l_u} \times \{0,1\}^{l_w} \to \{0,1\}$.
3. Строится SSP вида $Q = (v_0(x), \ldots, v_m(x), t(x))$, предназначенная для верификации C_R над $\mathbb{Z}_p$.
4. Выбираются первообразные элементы $g_1 \in \mathbb{G}_1$, $g_2, g_2' \in \mathbb{G}_2$. Образующий элемент g_2' может быть получен возведением g_2 в случайную степень $\alpha \in \mathbb{Z}_p$.
5. Выбираются случайные $\beta, s \in \mathbb{Z}_p^*$, такие, что $t(s) \neq 0$.
6. Вычисляются главная ссылочная строка σ и «лазейка» τ:

$$\sigma = (gk, g_1, g_2, \ldots, g_1^{s^d}, g_2^{s^d}, \{g_1^{\beta v_i(s)}\}_{i>l_u}, g_1^{\beta t(s)}, g_2', g_2'^{\beta}, Q),\ \tau = (\sigma, \beta, s). \tag{19}$$

7. На выход подаётся пара (σ, τ).

Алгоритм доказывающего $\pi \leftarrow \mathsf{Prove}(\sigma, u, w)$

На вход подаются ключ доказательства σ (19), публичный вход u, секретное свидетельство w и выполняются следующие шаги:

1. Открытый вход разбивается на биты: $u = (a_1, \ldots, a_{l_u}) \in \{0,1\}^{l_u}$.
2. На основе секретного свидетельства w выводятся такие $(a_{l_u+1}, \ldots, a_m)$, что целевой многочлен $t(x)$ делит $\left(v_0(z) + \sum\limits_{i=1}^{m} a_i v_i(x)\right)^2 - 1$.
3. Выбирается случайный элемент $\delta \in \mathbb{Z}_p$ и вычисляется полином-частное:

$$h(x) = \left(\left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x) + \delta t(x)\right)^2 - 1\right) / t(x).$$

4. С использованием ключа доказательства (19) вычисляются компоненты доказательства:

$$\begin{gathered} H = g_1^{h(s)},\ B_w = g_1^{\beta\left(\sum\limits_{i>l_u}^{m} a_i v_i(s) + \delta t(s)\right)}, \\ V_w = g_1^{\sum\limits_{i>l_u}^{m} a_i v_i(s) + \delta t(s)},\ V_2 = g_2^{v_0(s) + \sum\limits_{i=1}^{m} a_i v_i(s) + \delta t(s)}. \end{gathered} \tag{20}$$

5. На выход подается доказательство $\pi = (H, B_w, V_w, V_2) \in \mathbb{G}_1^3 \times \mathbb{G}_2$.

Доказательство состоит из четырёх элементов: трёх элементов группы $\mathbb{G}_1$ и одного элемента $\mathbb{G}_2$.

Алгоритм верификатора $0/1 \leftarrow \mathsf{Vfy}(\sigma, u, \pi)$

На вход подаются главная ссылочная строка σ (19), вход u и доказательство π (20). Затем выполняются следующие шаги:

1. Открытый вход разбивается на биты: $u = (a_1, \ldots, a_{l_u}) \in \{0,1\}^{l_u}$.
2. Доказательство π (20) разбивается на компоненты $(H, B_w, V_w, V_2) \in \mathbb{G}_1^3 \times \mathbb{G}_2$.
3. Вычисляется $V = g_1^{v_0(s)+\sum\limits_{i=1}^{l_u} a_i v_i(s)} V_w$.
4. В порядке описания проверяется достоверность обязательств, факт использования одинаковых коэффициентов и условие выполнения делимости SSP:

$$e(V, g_2) = e(g_1, V_2),\ e(V_w, g_2'^{\beta}) = e(B_w, g_2'),\ e(H, g_2^{t(s)}) = e(V, V_2)e(g_1, g_2)^{-1}. \quad (21)$$

5. Если проверки (21) выполняются, то верификация подтверждается.

Зафиксированное значение V_w (20) однозначно определяет компоненты B_w, V_2, H. Поэтому для любой пары $(u, w) \in R$ если действительные/имитированные доказательства выбираются случайным образом, то проверочные уравнения будут выполняться.

Верификация выполняется за счёт шести билинейных спариваний и одного умножения для каждого ненулевого входного бита, независимо от размера схемы.

Алгоритм имитирования доказательств $\pi \leftarrow \mathsf{Sim}(\tau, u)$

На вход подаётся «лазейка» τ (19), вход u и выполняются следующие шаги:

1. Открытый вход разбивается на биты: $u = (a_1, \ldots, a_{l_u}) \in \{0,1\}^{l_u}$.
2. Выводится случайный элемент $\delta_w \in \mathbb{Z}_p$.
3. Вычисляется оценка многочлена-частного в секретной точке s:

$$h = \left(\left(v_0(s) + \sum_{i=1}^{l_u} a_i v_i(s) + \delta_w \right)^2 - 1 \right) / \, t(s).$$

4. На выход подаётся доказательство $\pi = \left(g_1^h, g_1^{\delta_w}, g_1^{\beta\delta_w}, g_2^{v_0(s)+\sum\limits_{i=1}^{l_u} a_i v_i(s)+\delta_w} \right) \in \mathbb{G}_1^3 \times \mathbb{G}_2$.

10. Протокол С. Костелло, С. Фурне, Д. Хауэлла и др.

Протокол С. Костелло, С. Фурне, Д. Хауэлла и др. [62] использует протокол zk-SNARK Pinocchio [49] в конструкции доказательств с разбиением исходного QAP арифметической схемы на более простые мелкие составляющие, которые обмениваются состояниями посредством общей шины данных. В отличие от стандартного подхода [49], случай [62] компилирует отдельные элементы QAP только для динамических частей схемы, за счёт чего экономятся трудоёмкие криптографические операции при формировании общего доказательства. Подобная декомпозиция доказательств способствует масштабируемости. Доказывающий выполняет только криптографическую работу, пропорциональную пути, фактически выполненному программой.

По сравнению с [49] протокол [62] включает незначительные доработки. Например, поддерживаются индексы $j \in [l]$ для всех переменных, соответствующих отдельным QAP Q_i. Кроме того, ключ-«лазейка» τ разделяется на компоненты для имитирования и экспортирования доказательств:

$$(\tau_S, \tau_E) = (s, \{\alpha_{v,j}, \alpha_{w,j}, \alpha_{y,j}\}_{j\in[l]}, r_v, r_w), (r_v, r_w).$$

По сравнению с ключом оценки (12) [49] в текущий вариант EK_j [62] добавлены зашифрованные значения целевого полинома $t(s)$:

$$EK_j = (\ldots, g_v^{t(s)}, g_w^{t(s)}, g_y^{t(s)}, g_v^{\alpha_{v,j}t(s)}, g_w^{\alpha_{w,j}t(s)}, g_y^{\alpha_{y,j}t(s)}, g_v^{\beta_j t(s)}, g_w^{\beta_j t(s)}, g_y^{\beta_j t(s)}).$$

Дополнительно ключи верификации VK_j включают проверяемые верификатором дайджесты для связывания EK_j вида $C_j = (EK_j, u_j, o_j)$, где $o_j = (o_v, o_w, o_y)$ — случайные элементы поля. Дайджесты C_j используются в качестве открытого входа для функции верификации. Определяются коэффициенты $(c_k)_{k \in I_j} = u_j$ и для полиномов значений общей шины QAP $j \in S$ ($j \notin S$ — индексы несвязанных дайджестов) вычисляются следующие обязательства:

$$j \in S : g_y^{y^{(j)}(s)}, g_v^{\alpha_{y,j} y^{(j)}(s)}, g^{\beta_j (r_y y^{(j)}(s))},$$

$$j \notin S : g_v^{v^{(j)}(s)}, g_w^{w^{(j)}(s)}, g_y^{y^{(j)}(s)}, g_v^{\alpha_{v,j} v^{(j)}(s)}, g_w^{\alpha_{w,j} w^{(j)}(s)}, g_y^{\alpha_{y,j} y^{(j)}(s)}, g^{\beta_j (r_v v^{(j)}(s) + r_w w^{(j)}(s) + r_y y^{(j)}(s))},$$

где $v^{(j)}(s) = \sum\limits_{k \in I_j} c_k v_k(s) + o_v d(s)$ (для $w^{(j)}(s), y^{(j)}(s)$ используются o_w, o_y соответственно).

11. Протокол М. Бэкеса, М. Барбоза, Д. Фиоре, М. Рейщук

Протокол М. Бэкеса, М. Барбоза, Д. Фиоре, М. Рейщук [63] применяет QAP и представляет собой развитие идей [64, 49, 65], позволяющее формировать доказательства с использованием аутентифицированных данных (Authenticated Data, AD). Характерной особенностью схемы [63] является то, что верификатор получает информацию из надёжного источника и направляет её третьей стороне, которая в состоянии проверить действительность принимаемых данных, аутентифицированных исходным источником. В данном случае аутентификация основана на формировании MAC-кодов с использованием схем гомоморфного шифрования [66, 67]. В классическом протоколе zk-SNARK верификатор всегда знает открытое состояние, а в случае AD zk-SNARK аутентифицированные некоторым доверенным источником входные состояния верификатору не раскрываются. По сравнению с Pinocchio [49] протокол AD zk-SNARK [63] строит доказательства в 25 раз быстрее и сокращает требования к размеру памяти доказывающих в 20 раз.

Для произвольных арифметических схем может строиться как протокол PV, так и протокол DV AD zk-SNARK. В случае протокола DV AD zk-SNARK размер доказательств фиксируется константой, а для протокола PV AD zk-SNARK — растёт линейно количеству $N \leqslant n$ аутентифицированных утверждений. Алгоритм верификации работает линейно по N.

Для верификаторов, которым известен секретный ключ аутентификации, например в случае криптографических устройств с симметричным ключом в защищённой внутренней памяти, доказательства протокола AD zk-SNARK имеют постоянный размер и знание верификатором такого ключа не ставит под угрозу конфиденциальность схемы. Защищённость рассматриваемого протокола основана на допущениях q-DHE [68] и q-PKE [31] в билинейных группах. Конфиденциальность в виде свойства ZK также сохраняется против злоумышленников, которые знают/формируют ключи аутентификации.

Реализация протокола AD zk-SNARK [63] включает схему подписи на основе эллиптической кривой ed25519 [69]. На вход PRF, основанного на AES, подаются 128-битная метка и 256-битный ключ. Затем одно вычисление AES-128 отображает метку в 128-битное псевдослучайное начальное значение, применяемое во втором экземпляре AES-CTR-128 для расширения начального значения до 384 псевдослучайных бит, которые сокращаются по модулю до 254-битного элемента поля. Быстродействие криптографических функций может оцениваться на платформе Supercop [70].

Алгоритм начальной установки $\mathsf{Setup}(1^\lambda)$**.** На основе λ вычисляются публичные параметры билинейной группы $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2) \leftarrow_{\mathcal{R}} G(1^\lambda)$, группы $\mathbb{G}_i$

имеют простой порядок $p > 2^\lambda$, $P_1 \in \mathbb{G}_1$ и $P_2 \in \mathbb{G}_2$ — образующие элементы групп, e — производительное билинейное спаривание. Фиксируется конечное поле $\mathbb{F}$ вычетов по модулю p.

Алгоритм формирования ключей аутентификации $\mathsf{AuthKG}(\mathsf{pp})$

Формируется ключевая пара схемы подписи $(\mathsf{sk}', \mathsf{vk_a}') \leftarrow_{\mathcal{R}} \Sigma.\mathsf{KG}(1^\lambda)$. Для псевдослучайной функции $F_S : \{0,1\}^* \to \mathbb{F}$ (Pseudo-Random Functions, PRF) вычисляется начальное секретное заполнение S и публичный параметр prfpp в виде $(S, \mathsf{prfpp}) \leftarrow_{\mathcal{R}} \mathsf{F.KG}(1^\lambda)$. Выводится случайное значение $k \leftarrow_{\mathcal{R}} \mathbb{F}$ и вычисляется $K_1 = kP_1 \in \mathbb{G}_1$, $K_2 = kP_2 \in \mathbb{G}_2$. Формируются секретный ключ аутентификации $\mathsf{sk} = (\mathsf{sk}', S, k)$, публичный ключ верификации аутентификатора $\mathsf{vk_a} = (\mathsf{vk_a}', K_2)$ и публичные параметры аутентификации $\mathsf{pap} = (\mathsf{pp}, \mathsf{prfpp}, K_1)$.

В случае протокола DV AD zk-SNARK формируется дополнительный вывод $F_S : \{0,1\}^* \to \mathbb{G}_2$ и вычисляется $K = e(P_1, P_2)^k \in \mathbb{G}_T$. На выход подаются секретный ключ аутентификации $\mathsf{sk} = \mathsf{vk_a} = (S, k)$ и публичные параметры аутентификации $\mathsf{pap} = (\mathsf{pp}, \mathsf{prfpp}, K)$.

Алгоритм аутентификации $\mathsf{Auth}(\mathsf{sk}, \mathbf{L}, x)$

Для аутентификации значения $x \in \mathbb{F}$ с меткой $\mathbf{L}$ формируется $\phi = F_S(L)$ с использованием PRF, вычисляется $\mu = \phi + kx \in \mathbb{F}$ и $\Phi = \phi P_2 \in \mathbb{G}_2$. Затем вычисляется подпись $\sigma' \leftarrow_{\mathcal{R}} \Sigma.\mathsf{Sign}(\mathsf{sk}', \Phi \parallel \mathbf{L})$ и выводится значение-аутентификатор $\sigma = (\mu, \Phi, \sigma')$ (символ «$\parallel$» означает конкатенацию).

В случае протокола DV AD zk-SNARK формируется дополнительный вывод $\sigma = \Phi + xkP_2$.

Алгоритм верификации аутентификатора $\mathsf{AuthVer}(\mathsf{vk_a}, \sigma, \mathbf{L}, x)$

Используется ключ верификации аутентификатора $\mathsf{vk_a} = (\mathsf{vk_a}', K_2)$. Чтобы убедиться, что $\sigma = (\mu, \Phi, \sigma')$ является допустимым тегом аутентификации для $x \in \mathbb{F}$ относительно метки L, проверяется выполнение равенств $\mu P_2 = \Phi + xK_2$ в $\mathbb{G}_1$ и $\Sigma.\mathsf{Ver}(\mathsf{vk_a}', \Phi \parallel L, \sigma') = 1$. В случае секретной проверки $\mathsf{vk_a}$ заменяется на sk и аутентификатор σ проверяется равенством $\mu = F_S(L) + kx$.

В случае протокола DV AD zk-SNARK включается дополнительная проверка выполнения равенства $\sigma = F_S(\mathbf{L}) + xkP_2$.

Алгоритм формирования ключей $\mathsf{Gen}(\mathsf{pap}, C)$

Используется арифметическая схема $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$. Индексы проводов схемы $\{1, \ldots, m+3\}$ делятся на n значений публичного состояния входа/выхода $I_x = \{1, \ldots, n\}$ и h значений внутренних проводов секретного свидетельства $I_{\text{mid}} = \{n+1, \ldots, m+3\}$. Выполняются следующие шаги:

1. Для схемы C вычисляется QAP вида $Q_C = (\mathbf{a}, \mathbf{b}, \mathbf{c}, z) = \mathsf{QAPInst}(C)$ размера m и степени d. Значения $\mathbf{a}, \mathbf{b}, \mathbf{c}$ являются векторами по $m+1$ полиномов в $\mathbb{F}^{\leqslant d-1[X]}$, целевой полином $z \in \mathbb{F}[X]$ имеет степень d. Векторы расширяются тремя полиномами каждый:

$$a_{m+1}(X) = b_{m+2}(X) = c_{m+3}(X) = z(X),$$
$$a_{m+2}(X) = a_{m+3}(X) = b_{m+1}(X) = b_{m+3}(X) = c_{m+1}(X) = c_{m+2}(X) = 0.$$

2. Выбираются случайные значения $\rho_a, \rho_b, \tau, \alpha_a, \alpha_b, \alpha_c, \beta, \gamma \in \mathbb{F}$, $\rho_c = \rho_a \rho_b$ и для $k \in \{0, \ldots, m+3\}$ вычисляются компоненты ключей:

$$\begin{aligned} Z = z(\tau)\rho_c P_2, \quad & K_a = z(\tau)\rho_a K_1, \\ A_k = a_k(\tau)\rho_a P_1, \quad & A'_k = \alpha_a a_k(\tau)\rho_a P_1, \\ B_k = b_k(\tau)\rho_b P_2, \quad & B'_k = \alpha_b b_k(\tau)\rho_b P_1, \\ C_k = c_k(\tau)\rho_c P_1, \quad & C'_k = \alpha_c c_k(\tau)\rho_c P_1, \\ E_k = \beta(a_k(\tau)\rho_a + b_k(\tau)\rho_b + c_k(\tau)\rho_c)P_1. & \end{aligned} \tag{22}$$

Расширение протокола AD zk-SNARK для k различных ключей аутентификации/источников использует видоизменённый алгоритм $\mathsf{Gen}(\{\mathsf{pap}_j\}, C)$ с набором общедоступных параметров аутентификации $\mathsf{pap}_1, \ldots, \mathsf{pap}_k$. Тогда в EK_C (23) включается набор $\{K_{a,j} = z(\tau)\rho_a K_{1,j}\}_{j\in[k]}$.

В случае протокола DV AD zk-SNARK вычисляется новое значение $K_a = (K)^{z(\tau)} \in \mathbb{G}_T$ (22).

3. На выход подаются ключи оценки EK_C и верификации VK_C схемы C:

$$\begin{aligned} EK_C &= (Q_C, \mathbf{A}, \mathbf{A}', \mathbf{B}, \mathbf{B}', \mathbf{C}, \mathbf{C}', \mathbf{E}, \{\tau^i P_1\}_{i\in\{0,\ldots,d\}}, K_a), \\ VK_C &= (P_1, P_2, \alpha_a P_2, \alpha_b P_1, \alpha_c P_2, \gamma P_2, \beta\gamma P_1, \beta\gamma P_2, Z, \{A_k\}_{k=0}^n). \end{aligned} \tag{23}$$

Алгоритм доказательства $\mathsf{Prove}(EK_C, \mathbf{x}, \mathbf{w}, \boldsymbol{\sigma})$

Используются ключ оценки EK_C (23), пара «открытое состояние — секретное свидетельство» $(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^h$, набор тегов аутентификации для x вида $\sigma = (\sigma_1, \ldots, \sigma_n)$, где для всех $i \in [n]$ выполняется равенство $\sigma_i = (\mu_i, \Phi_i, \sigma'_i)$ или $\sigma_i = *$ (значение «*» означает пустой параметр). Определяются индексы $I_\sigma = \{i \in I_x : \sigma_i \neq *\} \subseteq I_x$, для которых существуют аутентифицированные данные, а также $I_* = I_x \backslash I_\sigma$ как индексы пустого дополнения. Примечательно, что σ не обязательно содержит теги аутентификации для всех позиций. Если значение в позиции i не аутентифицировано, то используется пустой тег $\sigma_i = *$.

Для k различных источников дополнительно каждый тег аутентификации $\sigma_i \in \sigma$ указывает на соответствующий ключ аутентификации ak_{j_i}, а набор I_σ разбивается на k подмножеств $I_{\sigma,j}$ — по одному множеству для каждого ключа аутентификации.

Для получения доказательства выполнимости $C(\mathbf{x}, \mathbf{w}) = 0^l$ выполняются следующие шаги:

1. Вычисляется $\mathbf{s} = \mathsf{QAPwit}(C, \mathbf{x}, \mathbf{w}) \in \mathbb{F}^m$ ($s_i = x_i$ для всех $i \in [n]$).
2. Выбираются случайные $\delta_a^\sigma, \delta_a^{\text{mid}}, \delta_b, \delta_c \in \mathbb{F}$, $\delta_a = \delta_a^\sigma + \delta_a^{\text{mid}}$ и фиксируется вектор $\mathbf{u} = (1, \mathbf{s}, \delta_a, \delta_b, \delta_c) \in \mathbb{F}^{m+4}$.

   Для k различных ключей аутентификации/источников дополнительно выводится набор случайных значений $(\delta_a^{(1)}, \ldots, \delta_a^{(k)})$, $\delta_a = \sum\limits_{j=1}^{k} \delta_a^{(j)} + \delta_a^{\text{mid}}$.
3. Решается задача QAP Q_C для вычисления коэффициентов $(h_0, \ldots, h_d) \in \mathbb{F}^{d+1}$ полинома $h \in \mathbb{F}[X]$, соответствующего выполнению равенства $h(X)z(X) = a(X)b(X) - c(X)$, где $a, b, c \in \mathbb{F}[X]$ имеют следующий вид:

$$\begin{aligned} a(X) &= a_0(X) + \sum_{k\in[m]} s_k a_k(X) + \delta_a z(x), \\ b(X) &= b_0(X) + \sum_{k\in[m]} s_k b_k(X) + \delta_b z(x), \\ c(X) &= c_0(X) + \sum_{k\in[m]} s_k c_k(X) + \delta_c z(x). \end{aligned}$$

Затем вычисляется $H = h(\tau)P_1$ с использованием $\tau^i P_1$ из ключа оценки EK_C (23). В итоге вычисляются значения $a(X) = \langle \mathbf{u}, \mathbf{a} \rangle$, $b(X) = \langle \mathbf{u}, \mathbf{b} \rangle$ и $c(X) = \langle \mathbf{u}, \mathbf{c} \rangle$.

4. Вычисляются компоненты доказательства:

$$\begin{gathered}
\pi_b = \langle \mathbf{u}, \mathbf{B} \rangle, \quad \pi'_b = \langle \mathbf{u}, \mathbf{B}' \rangle, \quad \pi_c = \langle \mathbf{u}, \mathbf{C} \rangle, \quad \pi'_c = \langle \mathbf{u}, \mathbf{C}' \rangle, \\
\pi_\sigma = \langle \mathbf{u}, \mathbf{A} \rangle_{I_\sigma} + \delta_a^\sigma A_{m+1}, \quad \pi'_\sigma = \langle \mathbf{u}, \mathbf{A}' \rangle_{I_\sigma} + \delta_a^\sigma A'_{m+1}, \\
\pi_{\text{mid}} = \langle \mathbf{u}, \mathbf{A} \rangle_{I_{\text{mid}}} - \delta_a^\sigma A_{m+1}, \quad \pi'_{\text{mid}} = \langle \mathbf{u}, \mathbf{A}' \rangle_{I_\sigma} - \delta_a^\sigma A'_{m+1}, \\
\pi_E = \langle \mathbf{u}, \mathbf{E} \rangle.
\end{gathered}$$

В случае k различных ключей аутентификации/источников для $j = 1, \ldots, k$ дополнительно вычисляются компоненты доказательств:

$$\begin{gathered}
\pi_{\sigma,j} = \langle \mathbf{u}, \mathbf{A} \rangle_{I_{\sigma,j}} + \delta_a^{(j)} A_{m+1}, \quad \pi'_{\sigma,j} = \langle \mathbf{u}, \mathbf{A}' \rangle_{I_{\sigma,j}} + \delta_a^{(j)} A'_{m+1}, \\
\pi_{\text{mid}} = \langle \mathbf{u}, \mathbf{A} \rangle_{I_{mid}} - \sum_{j=1}^{k} \delta_a^{(j)} A_{m+1}, \quad \pi'_{\text{mid}} = \langle \mathbf{u}, \mathbf{A}' \rangle_{I_\sigma} - \sum_{j=1}^{k} \delta_a^{(j)} A'_{m+1}.
\end{gathered} \tag{24}$$

5. Вычисляются аутентификационные данные для π_σ (24):

$$\pi_\mu = \langle \boldsymbol{\mu}, \mathbf{A} \rangle_{I_\sigma} + \delta_a^\sigma K_a. \tag{25}$$

В случае k различных ключей аутентификации/источников дополнительно вычисляются аутентификационные данные для $\pi_{\sigma,j}$ (24):

$$\pi_{\mu,j} = \langle \boldsymbol{\mu}, \mathbf{A} \rangle_{I_{\sigma,j}} + \delta_a^j K_{a,j}.$$

В случае протокола DV AD zk-SNARK вычисляется новое значение π_μ, отличное от (25):

$$\pi_\mu = \left(\prod_{k \in I_\sigma} e(A_k, \Phi_k) \right) (K_a)^{\delta_a^\sigma} \in \mathbb{G}_T.$$

6. На выход подаётся доказательство и, в случае протокола PV zk-SNARK, набор дополнительных значений:

$$\begin{gathered}
\pi = (\pi_\mu, \pi_\sigma, \pi'_\sigma, \pi_{\text{mid}}, \pi'_{\text{mid}}, \pi_b, \pi'_b, \pi_c, \pi'_c, \pi_E, H), \\
\{\Phi_k, \sigma'_k\}_{k \in I_\sigma}.
\end{gathered} \tag{26}$$

В случае k различных ключей аутентификации/источников дополнительно на выход подается набор компонент $\{\pi_{\mu,j}, \pi_{\sigma,j}, \pi'_{\sigma,j}\}_{j=1}^k$.

Алгоритм верификации $\mathsf{Ver}(\mathsf{vk_a}, VK_C, \mathbf{L}, \{x_i\}_{L_i=*}, \pi)$

Используется ключ верификации аутентификатора $\mathsf{vk_a}$, ключ верификации схемы VK_C (23), вектор меток $\mathbf{L} = (L_1, \ldots, L_n)$, неаутентифицированные компоненты состояния x_i и доказательство π (26). Определяются аналогичные доказывающему индексы I_σ, I_*. Вычисляется значение $A_* = A_0 + \langle \mathbf{x}, \mathbf{A} \rangle_{I_*}$ и выполняются следующие шаги:

1. Используется секретный ключ верификации $\mathsf{sk} = (S, k)$ для проверки подлинности π_σ по меткам $\mathbf{L}$ за счёт выполнения уравнения в $\mathbb{G}_1$:

$$\pi_\mu = \langle F_S(\mathbf{L}), \mathbf{A} \rangle_{I_\sigma} + k\pi_\sigma = \sum_{i \in I_\sigma} F_S(L_i) A_i + k\pi_\sigma. \tag{27}$$

В случае протокола DV zk-SNARK вместо (27) включается проверка подлинности π_σ за счёт выполнения нового равенства в $\mathbb{G}_T$:

$$\pi_\mu = \left(\prod_{k\in I_\sigma} e(A_k, F_S(L_k))\right) e(\pi_\sigma, kP_2).$$

2. В слючае протокола PV zk-SNARK используется публичный ключ верификации $\mathsf{vk_a} = (\mathsf{vk_a}', K_2)$. Сначала проверяется правильность всех Φ_k за счёт контроля $\Sigma.\mathsf{Ver}(\mathsf{vk_a}', \Phi_k \parallel L_k, \sigma'_k) = 1$ для всех $k \in I_\sigma$. Затем проверяется подлинность π_σ за счёт выполнения равенства в $\mathbb{G}_T$:

$$e(\pi_\mu, P_2) = \prod_{k\in I_\sigma} e(A_k, \Phi_k) e(\pi_\sigma, K_2). \tag{28}$$

3. Проверяется подлинность обязательств для аутентифицированных значений:

$$e(\pi'_\sigma, P_2) = e(\pi_\sigma, \alpha_a P_2). \tag{29}$$

4. В порядке описания проверяется выполнимость QAP, корректность обязательств и подтверждение использования одинаковых коэффициентов для всех линейных комбинаций QAP:

$$\begin{gathered} e(A_* + \pi_\sigma + \pi_{\text{mid}}, \pi_b) = e(H, Z) e(\pi_c, P_2), \\ (e(\pi'_{\text{mid}}, P_2) = e(\pi_{\text{mid}}, \alpha_a P_2), \quad e(\pi'_b, P_2) = e(\alpha_b P_1, \pi_b), \quad e(\pi'_c, P_2) = e(\pi_c, \alpha_c P_2)), \\ e(\pi_E, \gamma P_2) = e(A_* + \pi_\sigma + \pi_{\text{mid}} + \pi_c, \beta\gamma P_2) e(\beta\gamma P_1, \pi_b). \end{gathered} \tag{30}$$

5. Верификация успешна, если все уравнения (27), (28) и (30) выполняются.
В случае k различных ключей аутентификации/источников алгоритм верификации дополнительно проверяет уравнения (27), (28) и (29) для набора $\{\pi_{\mu,j}, \pi_{\sigma,j}, \pi'_{\sigma,j}\}_{j=1}^k$.

Алгоритм рерандомизации доказательств $\mathsf{ReRand}(EK_C, \mathbf{L}, \{x_i\}_{L_i=*}, \pi)$

Если π (26) подтверждают наборы меток $\mathbf{L}$ и неаутентифицированных значений $\{x_i\}_{L_i=*}$, то π возможно повторно рерандомизировать. Для этого выполняются следующие шаги:

1. Выбираются случайные значения $\tilde\delta_a^\sigma, \tilde\delta_a^{\text{mid}}, \tilde\delta_b, \tilde\delta_c \in \mathbb{F}$, $\tilde\delta_a = \tilde\delta_a^\sigma + \tilde\delta_a^{\text{mid}}$.
2. В компоненты доказательства вводятся новые случайные значения:

$$\begin{gathered} \tilde\pi_b = \pi_b + \tilde\delta_b B_{m+2}, \quad \tilde\pi'_b = \pi'_b + \tilde\delta_b B'_{m+2}, \\ \tilde\pi_c = \pi_c + \tilde\delta_c C_{m+3}, \quad \tilde\pi'_c = \pi'_c + \tilde\delta_c C'_{m+3}, \\ \tilde\pi_\sigma = \pi_\sigma + \tilde\delta_a^\sigma A_{m+1}, \quad \tilde\pi_\sigma{}' = \pi'_\sigma + \tilde\delta_a^\sigma A'_{m+1}, \\ \tilde\pi_{\text{mid}} = \pi_{\text{mid}} + \tilde\delta_a^{\text{mid}} A_{m+1}, \quad \tilde\pi'_{\text{mid}} = \pi'_{\text{mid}} + \tilde\delta_a^{\text{mid}} A'_{m+1}, \\ \tilde\pi_E = \pi_E + \tilde\delta_a E_{m+1} + \tilde\delta_b E_{m+2} + \tilde\delta_c E_{m+3}, \\ \tilde\pi_\mu = \pi_\mu + \tilde\delta_a^\sigma K_a, \\ \tilde H = H + \tilde\delta_a \pi_b + \tilde\delta_b \pi_a + \tilde\delta_a \tilde\delta_b z(\tau) P_1 - \tilde\delta_c P_1. \end{gathered} \tag{31}$$

В (31) значение $z(\tau)P_1$ может включаться в ключ оценки EK_C (23).

3. На выход подаётся обновлённое повторно рандомизированное значение доказательства $\tilde\pi = (\tilde\pi_\mu, \tilde\pi_\sigma, \tilde\pi'_\sigma, \tilde\pi_{\text{mid}}, \tilde\pi'_{\text{mid}}, \tilde\pi_b, \tilde\pi'_b, \tilde\pi_c, \tilde\pi'_c, \tilde\pi_E, \tilde H)$.

Доказательство полноты рассмотренного протокола AD zk-SNARK представлено в [63].

По сравнению с производительной версией [49], представленной в [65], алгоритм Gen протокола [63] имеет одно дополнительное возведение в степень в $\mathbb{G}_1$ для формирования $K_a = z(\tau)\rho_a K_1$. Размер нового EK_C (23) расширен на один элемент $K_a \in \mathbb{G}_1$. Доказательство протокола AD zk-SNARK расширено на три элемента $\pi_\sigma, \pi'_\sigma, \pi_\mu \in \mathbb{G}_1$, каждый с $N = |I_\sigma|$ возведениями в степень, а также набором подписей $\{\sigma_k\} \in I_\sigma$ для протокола PV zk-SNARK. Верификатор протокола DV AD zk-SNARK вычисляет $|I_*| = n - N$ значений A_*, что меньше случая [49], а уравнение (27) требует многократного возведения в степень с $N = |I_\sigma|$ значениями и дополнительных вычислений PRF, что взаимно компенсируется. В итоге верификатор протокола DV AD zk-SNARK [63] ипользует на два билинейных спаривания в (29) больше, чем [49]. В случае протоколов PV zk-SNARK добавлено уравнение (28) с двумя билинейными спариваниями и $|I_\sigma|$ произведениями.

Протоколы [63, 71] унаследовали проблемы защищённости [72]. А. Габизон [73] выявляет уязвимость протокола zk-SNARK [72], связанную с включением в CRS избыточных по сравнению с исходным вариантом Pinocchio [49] элементов ключа верификации. При наличии доказательства некоторого заданного общедоступного входа это позволяет создавать доказательства для любых других входов. Для случая исключения данных элементов и при удовлетворении QAP определённым условиям в [73] представлено доказательство защищённости [72] в общей групповой модели.

12. Протокол Й. Грота на основе асимметричного билинейного спаривания

Протокол zk-SNARK Й. Грота [74] строится на основе QAP, билинейного спаривания и устраняет возможность раскрытия информации из CRS. Рассматривается вариант несимметричного билинейного спаривания $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, где g — образующий элемент $\mathbb{G}_1$; h — образующий элемент $\mathbb{G}_2$; $e(g, h)$ — образующий элемент $\mathbb{G}_T$, однако симметричный случай при $\mathbb{G}_1 = \mathbb{G}_2$ и $g = h$ выполняется аналогично. Доказательства знания состоят из трёх элементов группы. При работе с билинейным спариванием типа III [75] верификация в степенях дискретных логарифмов требует указания групп для каждого элемента. Поэтому главная ссылочная строка и доказательство разделяются на две части, в связи с чем являются составными: $\sigma = (\sigma_1, \sigma_2), \pi = (\pi_1, \pi_2)$.

Таким образом, сначала строится два сообщения для QAP, которая выводит бинарное отношение вида

$$R = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, l, \{u_i(x), v_i(x), w_i(x)\}_{i=0}^m, t(x)). \tag{32}$$

Отношение (32) предусматривает зависимость $|p| = \lambda$, задаёт поле $\mathbb{Z}_p$, язык состояний $(a_1, \ldots, a_l) \in \mathbb{Z}_p^l$ (открытые значения проводов) и секретных свидетельств $(a_{l+1}, \ldots, a_m) \in \mathbb{Z}_p^{m-l}$ (секретные значения проводов), удовлетворяющих выполнимости арифметической схемы (согласованность проводов ввода/вывода). Арифметические схемы формируют соотношения, описываемые уравнениями над множеством переменных утверждений и секретных свидетелей, удовлетворяющих всем уравнениям. При этом для $a_0 = 1$ выполняется соответствующее равенство QAP с $\deg(h(x)) = n - 2$ и $\deg(t(x)) = n$:

$$\sum_{i=0}^m a_i u_i(x) \sum_{i=0}^m a_i v_i(x) = \sum_{i=0}^m a_i w_i(x) + h(x)t(x).$$

В уравнении верификации элементы доказательства $\pi = (A, B, C)$ используются только один раз, поэтому их легко разнести в разные части билинейного теста. Разде-

ление общей ссылочной строки на две части позволяет вычислять отдельные компоненты доказательства. Составная конструкция также устраняет возможность раскрытия информации из общей ссылочной строки и поэтому формирует протокол zk-SNARK в общей групповой модели. В результате строится протокол zk-SNARK с QAP-степенью d и значениями μ, m, n, k, η в виде констант или функций от параметра защиты λ.

Алгоритм формирования ключей $(\sigma, \tau) \leftarrow \mathsf{Setup}(R)$

На основе отношения R (32) вырабатываются ключи доказательства и верификации, где $\sigma = (\sigma_1, \sigma_2) \in \mathbb{F}^{m_1} \times \mathbb{F}^{m_2}, \tau \in \mathbb{F}^n$. Ключ доказательства: $\sigma = (g^{\sigma_1}, h^{\sigma_2}) = ([\sigma_1]_1, [\sigma_2]_2)$, где g, h — соответствующие порождающие элементы групп билинейного спаривания $\mathbb{G}_1$ и $\mathbb{G}_2$. Части σ_1 и σ_2 содержат единицу для исключения различий между аффинными и линейными функциями. Алгоритм Setup выполняет следующие шаги:

1. Выводятся случайные элементы $\alpha, \beta, \gamma, \delta, x \in \mathbb{Z}_p^*$.
2. Выводится ключ верификации для протокола PV/DV zk-SNARK $\tau = (\alpha, \beta, \gamma, \delta, x)$.
3. Вычисляется двухкомпонентный ключ доказательства $\sigma = ([\sigma_1]_1, [\sigma_2]_2)$, соответствующий CRS, содержащий многовариантные полиномы Лорана, оцениваемые в элементах из $\mathbb{Z}_p^*$, где
$$\sigma_1 = \begin{pmatrix} \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma\}_{i=0}^{l}, \\ \{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta\}_{i=l+1}^{m}, \{x^i t(x)/\delta\}_{i=0}^{n-2} \end{pmatrix}, \quad (33)$$
$$\sigma_2 = \left(\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}\right).$$
4. На выход подаётся пара ключей (σ, τ).

Строка CRS в виде σ (33) включает определение бинарного отношения R (32), n элементов $\mathbb{Z}_p$, $m + 2n + 3$ элементов $\mathbb{G}_1$ и $n + 3$ элементов $\mathbb{G}_2$.

Алгоритм доказывающего $\pi \leftarrow \mathsf{Prove}(R, \sigma, a_1, \ldots, a_l, a_{l+1}, \ldots, a_m)$

Идея заключается в построении пары матриц $(\Pi_1, \Pi_2) \leftarrow \mathsf{ProofMatrix}(R, x, w)$, где $\Pi_1 \in \mathbb{F}^{k_1 \times m_1}$, $\Pi_2 \in \mathbb{F}^{k_2 \times m_2}$, необходимых для формирования доказательства вида $\pi = (g^{\pi_1}, h^{\pi_2}) = ([\pi_1]_1, [\pi_2]_2) = (\Pi_1[\sigma_1]_1, \Pi_2[\sigma_2]_2)$. Таким образом, для отношения R (32), ключа σ (33), входа ϕ и секретного свидетеля w, где $(\phi \parallel w) = (a_1, \ldots, a_l, a_{l+1}, \ldots, a_m)$, выполняются следующие шаги:

1. Выводятся случайные элементы аддитивного поля $r, s \in \mathbb{Z}_p$.
2. Вычисляется матрица Π размера $3 \times (m + 2n + 4)$. В данном случае, при использовании стратегии аффинного доказательства первая строка матрицы Π с известными элементами поля $A_\alpha, A_\beta, A_\gamma, A_\delta, A_i$ и полиномами $A(x), A_h(x)$ соответствующих степеней $(n-1)$ и $(n-2)$ имеет следующий вид:
$$A_\Pi^{(1)} = A_\alpha \alpha + A_\beta \beta + A_\gamma \gamma + A_\delta \delta + A(x) + \sum_{i=0}^{l} A_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma + \\ + \sum_{i=l+1}^{m} A_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta + A_h(x)t(x)/\delta.$$
Выражения для $B_\Pi^{(2)}$ и $C_\Pi^{(3)}$ формируются аналогичным образом и содержатся во второй и третьей строках мартицы Π соответственно.
3. Вычисляется доказательство $\pi = \Pi\sigma = ([A]_1, [C]_1, [B]_2)$, где
$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta, \quad B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta,$$
$$C = \left(\sum_{i=l+1}^{m} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) + h(x)t(x) \right) / \delta + As + Br - rs\delta. \quad (34)$$

Значения $[A]_1$ и $[C]_1$ вычисляются линейно из Π_1 и $[\sigma_1]_1$ ($[A, C]_1 = \Pi_1[\sigma_1]_1$), а $[B]_2$ вычисляется линейно из Π_2 и $[\sigma_2]_2$ ($[B]_2 = \Pi_2[\sigma_2]_2$).

4. На выход подаётся доказательство π.

Значения r и s используются для рандомизации компонент A, B и C доказательства π (34) — введение свойства ZK. Значения α и β в ключе доказательства σ (33) предназначены для согласования A, B и C друг с другом в доказательстве π (34) при выборе $(a_0, \ldots, a_m)$.

Размер доказательства π (34) составляет два элемента $\mathbb{G}_1$ и один элемент $\mathbb{G}_2$.

Алгоритм верификатора $0/1 \leftarrow \mathsf{Vfy}(R, \sigma, a_1, \ldots, a_l, \pi)$

Идея заключается в выработке арифметической схемы $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$, которая соответствует матрицам $T_1, \ldots, T_\eta \in \mathbb{F}^{(m_1+k_1)\times(m_2+k_2)}$, где $\mathbf{t} : \mathbb{F}^{m_1+k_1+m_2+k_2} \to \mathbb{F}^\eta$ — квадратичный многовариантный целевой полином. Доказательство $\pi = ([\pi_1]_1, [\pi_2]_2) \in \in \mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$ (34) принимается, если для всех матриц $T_1, \ldots, T_\eta$ выполняется равенство $\mathbf{t}(\sigma, \pi) = 0$ в следующем виде:

$$\begin{bmatrix}\sigma_1 \\ \pi_1\end{bmatrix}_1 \cdot T_i \begin{bmatrix}\sigma_2 \\ \pi_2\end{bmatrix}_2 = [0]_T,$$

где $[0]_T = e(g, h)^0$ соответствует нейтральному элементу результирующей группы билинейного спаривания $\mathbb{G}_T$. Таким образом, алгоритм Vfy выполняется на основании отношения R (32), ключа доказательства σ (33), входа $\phi = (a_1, \ldots, a_l)$ и доказательства π (34) в виде следующих шагов:

1. Разделение доказательства на компоненты: $\pi = ([A]_1, [C]_1, [B]_2) \in \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2$.
2. Проверка верификационного уравнения $\mathbf{t}(\sigma, \pi) = 0$ за счёт равенства

$$[A]_1 \cdot [B]_2 = [\alpha]_1 \cdot [\beta]_2 + \sum_{i=0}^{l} a_i \left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right]_1 \cdot [\gamma]_2 + [C]_1 \cdot [\delta]_2. \quad (35)$$

3. В случае выполнения теста (35) доказательство π (34) принимается.

Произведение $\alpha\beta$ в верификации (35) гарантирует, что A и B включают нетривиальные компоненты α и β. Это означает, что произведение AB включает линейную зависимость от α и β, которая уравновешивается за счёт C с последовательным выбором $(a_0, \ldots, a_l, a_{l+1}, \ldots, a_m)$ в трёх компонентах A, B и C доказательства π (34).

Роль γ и δ состоит в том, чтобы сделать два последних произведения проверочного уравнения (35) независимыми от первого произведения за счёт деления левых множителей на γ и δ соответственно. Это предотвращает смешивание и сопоставление элементов, предназначенных для разных продуктов в уравнении верификации (35).

Если протокол zk-SNARK включает элементы только одной из групп $\mathbb{G}_1$ или $\mathbb{G}_2$, то верификация (35) преобразуется в систему линейных уравнений, что полностью разрушает защищённость протокола. Поэтому для билинейного спаривания типа III [75] требуется включение элементов обеих групп.

Общая ссылочная строка не содержит раскрытия, так как компоненты σ_1 и σ_2 (33) содержат многовариантные полиномы Лорана, оцениваемые в элементах $\mathbb{Z}_p^*$. Проверочное уравнение рассматривается как равенство соответствующих полиномов Лорана.

Согласно лемме Шварца — Циппеля [76], верификация выполняется при рассмотрении A, B и C (34) как формальных многочленов в недетерминированных точках $\alpha, \beta, \gamma, \delta$ и x, иначе верификация имеет пренебрежимо малую вероятность успеха. Тестирование $\sigma_1 \cdot T\sigma_2 = 0$ выполняется, так как соответствующий формальный многовариантный полином Лорана равен нулю. Случай выполнения тестирования $\sigma_1 \cdot T\sigma_2 = 0$

для ненулевого полинома Лорана имеет пренебрежимо малую вероятность, так как отрицательные и положительные суммарные степени полиномиально ограничены λ.

Алгоритм имитирования доказательств $\pi \leftarrow \mathsf{Sim}(R, \tau, a_1, \ldots, a_l)$

Строится доказательство $\pi = (g^{\pi_1}, h^{\pi_2}) = ([\pi_1]_1, [\pi_2]_2)$. Для отношения R (32), ключа верификации τ и открытого состояния $\phi = (a_1, \ldots, a_l)$ выполняются следующие шаги:

1. Выбираются $A, B \in \mathbb{Z}_p$.
2. Вычисляется компонента C доказательства π:

$$C = \left(AB - \alpha\beta - \sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x)) \right) / \delta.$$

3. На выход подаётся доказательство $\pi = (A, B, C)$.

13. Протокол Э. Бен-Сассона, А. Кьезы, Д. Генкина и др.

В протоколе Э. Бен-Сассона, А. Кьезы, Д. Генкина и др. [54] формирователь ключей $G(1^\lambda, C)$ выбирает сообщение верификатора $\mathbf{q} \in \mathbb{F}^{m'}$ (которое зависит от схемы C, но не от её входа) для LIP (Linear Interactive Proof) и выводит ключ доказательства $\sigma = \mathsf{E}(\mathbf{q}) = (\mathsf{E}(q_i))_{i=1}^{m'}$. Начиная с $\sigma = \mathsf{E}(\mathbf{q})$, честный доказывающий P гомоморфно зашифровывает скалярные произведения $\mathsf{E}(\langle \pi, q_i \rangle)$ для $i = 1, \ldots, k+1$ и $\pi \in \mathbb{F}^m$ для линейных вероятностно-проверяемых доказательств (Linear Probabalistically Checkable Proofs, LPCP), выполняя $k+1$ гомоморфных зашифрований скалярных произведений. В качестве доказательства P возвращает полученные защищённые ответы. Процедура зашифрования имеет вид $\mathsf{E}(\gamma) = (g^\gamma, h^\gamma)$, где g, h являются образующими элементами групп $\mathbb{G}_1$, $\mathbb{G}_2$ порядка r соответственно. Схемы линейного гомоморфного шифрования выполняют преобразование вида $\mathsf{E}(a\gamma + b\delta) = \mathsf{E}(\gamma)^a E(\delta)^b$ с покоординатным умножением и возведением в степень. Схемы полностью гомоморфного шифрования приведены, например, в [77], где обеспечивается следующее свойство корректности: $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}(F, \mathsf{Enc}_{\mathsf{pk}}(x_1), \ldots, \mathsf{Enc}_{\mathsf{pk}}(x_n))) = F(x_1, \ldots, x_n)$, где $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$; k — параметр защиты (длина секретного ключа).

В работе [54] построение LPCP для отношения $\mathcal{R}_\mathcal{C}$ основано на QSP/QAP [37]. Источник [76] отмечает, что QSP для отношения $\mathcal{R}_\mathcal{C}$ даёт LPCP с тремя запросами, а QAP — LPCP с четырьмя запросами. В рассматриваемом случае применяется подход QAP из [37] с 5-запросным LPCP для отношения $\mathcal{R}_\mathcal{C}$.

Размер схемы определяется как общее количество узлов. Булева схема $C : \{0,1\}^n \times \times \{0,1\}^h \to \{0,1\}$ с α соединениями (проводами) и β (билинейными) вентилями индуцирует соответствующую систему квадратных уравнений $\mathcal{S}$ с $N_\mathrm{w} = \alpha$ переменными и $N_\mathrm{g} = \beta + h + 1$ ограничениями. Дополнительные $h+1$ ограничений гарантируют, что соединения секретного свидетельства h имеют логические значения и выходной вентиль выводит 0. Арифметическая схема $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ с α проводами и β (билинейными) вентилями индуцируют соответствующую систему квадратных уравнений $\mathcal{S}$ с $N_\mathrm{w} = \alpha$ переменными и $N_\mathrm{g} = \beta + l$ ограничениями. Без потери общности далее рассматривается отношение $\mathcal{R}_\mathcal{C} = \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^h : C(\mathbf{x}, \mathbf{w}) = 0\,(0^l)\}$.

Элемент с входами $x_1, \ldots, x_n \in \mathbb{F}$ называется билинейным, если выходом является $\langle \mathbf{a}, (1, x_1, \ldots, x_n) \rangle \cdot \langle \mathbf{b}, (1, x_1, \ldots, x_n) \rangle$ для некоторых $\mathbf{a}, \mathbf{b} \in \mathbb{F}^{n+1}$, где $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение.

Для определения бинарного отношения $\mathcal{R}_\mathcal{S}$ фиксируется система квадратных уравнений ранга 1 над $\mathbb{F}$ с набором $\mathcal{S} = ((\mathbf{a}_j, \mathbf{b}_j, \mathbf{c}_j)_{j=1}^{N_\mathrm{g}}, n)$, где $\mathbf{a}_j, \mathbf{b}_j, \mathbf{c}_j \in \mathbb{F}^{1+N_\mathrm{w}}$ и $n \leqslant N_\mathrm{w}$. Такая система $\mathcal{S}$ выполнима с входом $\mathbf{x} \in \mathbb{F}^n$, если есть секретное свидетельство

$\mathbf{w} \in \mathbb{F}^{N_\mathrm{w}}$ при $\mathbf{x} = (w_1, \ldots, w_n)$ и $\langle \mathbf{a}_j, (1, \mathbf{w}) \rangle \cdot \langle \mathbf{b}_j, (1, \mathbf{w}) \rangle = \langle \mathbf{c}_j, (1, \mathbf{w}) \rangle$ для всех $j \in [N_\mathrm{g}]$. В таком случае $\mathcal{S}(\mathbf{x}, \mathbf{w}) = 1$. Параметр N_g определяет количество ограничений, N_w — количество переменных, n — размер входа.

Фиксируется произвольное подмножество $S = \{\alpha_1, \ldots, \alpha_{N_\mathrm{g}}\}$ в $\mathbb{F}$, $|S| = N_\mathrm{g}$. Для $i \in \in \{0, 1, \ldots, N_\mathrm{w}\}$ определяется тройка функций $A_i, B_i, C_i : S \to \mathbb{F}$ следующим образом: для каждого $j \in [N_\mathrm{g}]$

$$A_i(\alpha_j) = \mathbf{a}_j(i),\ B_i(\alpha_j) = \mathbf{b}_j(i),\ C_i(\alpha_j) = \mathbf{c}_j(i).$$

Каждая функция A_i, B_i, C_i расширяется до многочлена от одной переменной степени $(N_\mathrm{g} - 1)$ над $\mathbb{F}$. Определяется Z_S как многочлен от одной переменной степени N_g над $\mathbb{F}$, равный нулю на S. Доказывающий P_LPCP на основе входных данных строит доказательство в виде вектора π элементов поля $\mathbb{F}$ (это результат выбора честным оракулом линейного доказательства).

Таким образом, на входе $(\mathbf{x}, \pi)$ верификатор $V^{\pi}_\mathrm{LPCP}(x) = (Q_\mathrm{LPCP}, D_\mathrm{LPCP})$ делает k запросов к оракулу (доказательству) π. Основываясь на внутренней случайности, независимо от $\mathbf{x}$ формируется k запросов $\mathbf{q_1}, \ldots, \mathbf{q_k} \in \mathbb{F}^m$ к π и информация о состоянии $\mathbf{u}$. На заданных $(\mathbf{x}, \mathbf{u}, k)$ оракул (доказывающий) отвечает $a_1 = \langle \pi, \mathbf{q_1} \rangle, \ldots, a_k = \langle \pi, \mathbf{q_k} \rangle$, а D_LPCP верифицирует доказательство.

Алгоритм вычисления доказательства P_LPCP

Для входа $\mathbf{x} \in \mathbb{F}^n$ и секретного свидетельства $\mathbf{w} \in \mathbb{F}^{N_\mathrm{w}}$, таких, что $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_\mathcal{S}$, алгоритм P_LPCP выполняется следующим образом:

1. Случайно и равновероятно выбираются $\delta_1, \delta_2, \delta_3 \in \mathbb{F}$.
2. Пусть $\mathbf{h} = (h_0, h_1, \ldots, h_{N_\mathrm{g}}) \in \mathbb{F}^{N_\mathrm{g}+1}$ — коэффициенты многочлена $H(z)$ от одной переменной:

$$H(z) = \frac{A(z)B(z) - C(z)}{Z_S(z)}.$$

Здесь A, B, C — многочлены от одной переменной степени N_g, которые определяются следующим образом:

$$A(z) = A_0(z) + \sum_{i=1}^{N_\mathrm{w}} w_i A_i(z) + \delta_1 Z_S(z),$$
$$B(z) = B_0(z) + \sum_{i=1}^{N_\mathrm{w}} w_i B_i(z) + \delta_2 Z_S(z),$$
$$C(z) = C_0(z) + \sum_{i=1}^{N_\mathrm{w}} w_i C_i(z) + \delta_3 Z_S(z).$$

3. Выводится вектор $\pi = (\delta_1, \delta_2, \delta_3, 1, \mathbf{w}, \mathbf{h}) \in \mathbb{F}^{3+(N_\mathrm{w}+1)+(N_\mathrm{g}+1)}$.

Алгоритм формирования запросов Q_LPCP

1. Случайно и равновероятно выбирается $\tau \in \mathbb{F}$.
2. Выводится пять запросов $(\mathbf{q}_1, \ldots, \mathbf{q}_5)$, $\mathbf{q}_i \in \mathbb{F}^{5+N_\mathrm{w}+N_\mathrm{g}}$, следующего вида:

$$\mathbf{q}_1 = Z_S(\tau), 0, 0, A_0(\tau), \ldots, A_{N_\mathrm{w}}(\tau), 0, \ldots, 0;$$
$$\mathbf{q}_2 = 0, Z_S(\tau), 0, B_0(\tau), \ldots, B_{N_\mathrm{w}}(\tau), 0, \ldots, 0;$$
$$\mathbf{q}_3 = 0, 0, Z_S(\tau), C_0(\tau), \ldots, C_{N_\mathrm{w}}(\tau), 0, \ldots, 0;$$
$$\mathbf{q}_4 = 0, \ldots, 0, 1, \tau, \tau^2, \ldots, \tau^{N_\mathrm{g}};$$
$$\mathbf{q}_5 = 0, 0, 0, 1, \tau, \tau^2, \ldots, \tau^n, 0, \ldots, 0.$$

3. Выводится $\mathbf{u} = (u_1, \ldots, u_{n+2})$, где $u_i = \tau^{i-1}$ для $i \in \{1, \ldots, n+1\}$ и $u_{n+2} = Z_S(\tau)$.

В заключение алгоритм принятия решения D_{LPCP} проверяет принадлежность $\mathbf{x} \in \mathcal{L}_{\mathcal{S}}$ путём повторной проверки информации о состоянии $\mathbf{u}$, созданной алгоритмом запроса Q_{LPCP}, а также элементами поля $a_1 = \langle \pi^*, \mathbf{q}_1 \rangle, \ldots, a_5 = \langle \pi^*, \mathbf{q}_5 \rangle$, которые являются ответами, связанными с линейным доказательством π^*. В общем случае доказательство π^* рассматривается как потенциально опасное, возможно сформированное нечестным доказывающим.

Алгоритм принятия решения D_{LPCP}

Для входа $\mathbf{x} \in \mathbb{F}^n$, информации о состоянии $\mathbf{u} = (u_1, \ldots, u_{n+2})$ и ответов $(a_1, \ldots, a_5) = \langle \pi, q_1 \rangle, \ldots, \langle \pi, q_5 \rangle$ верификатор D_{LPCP} принимает доказательство, если выполняются следующие равенства:

$$a_1 a_2 - a_3 - a_4 u_{n+2} = 0, \quad a_5 - u_1 - \sum_{i=1}^{n} x_i u_{i+1} = 0.$$

Описанный LPCP имеет пять запросов по $(5 + N_{\mathrm{w}} + N_{\mathrm{g}})$ элементов $\mathbb{F}$ и информацию о состоянии с $(n+2)$ элементами $\mathbb{F}$. Для Q_{LPCP} каждая координата запроса является оценкой полиномов Z_S, A, B, C степени не выше N_{g} от случайного $\tau \in \mathbb{F}$, а D_{LPCP} проверяет ноль двух многочленов степени 2. Поэтому LPCP имеет степень $(d_Q, d_D) = (N_{\mathrm{g}}, 2)$. Значение a_4 определяется от a_1, a_2, a_3, u_{n+2} через ограничение $a_1 a_2 - a_3 - a_4 u_{n+2} = 0$, поэтому a_4 также не раскрывает дополнительной информации. Значение a_5 содержит информацию о части $\mathbf{w}$, равной $\mathbf{x}$, которая известна верификатору по определению. Таким образом, $(a_1, \ldots, a_5, \mathbf{u})$ имеют независимые от $\mathbf{w}$ распределения.

14. Протокол Э. Бен-Сассона, А. Кьезы, Э. Тромера, М. Вирзы

Протокол zk-SNARK [65] основан на [49] и используется для доказательства/проверки выполнимости $\mathbb{F}_r$-арифметических схем. Отличие от [49] заключается в отсутствии предположений равенства $\mathbb{G}_1 = \mathbb{G}_2$, а также в увеличении размера ключа верификации в зависимости от размера n входа $\mathbf{x}$ в виде $n + O(n)$ вместо $3n + O(n)$. Кроме того, [65] устраняет обнаруженную уязвимость [78], оформленную как CVE-2019-7167 в отношении криптовалюты Zcash: pk'_A теперь начинается с индекса $n+1$, а $p\tilde{k}_A, p\tilde{k}'_A$ переопределены соответствующим образом.

Публичные параметры включают простое число r, две циклические группы $\mathbb{G}_1, \mathbb{G}_2$ порядка r с образующими P_1, P_2 соответственно и спаривание $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (где $\mathbb{G}_T$ также циклична порядка r). Защищённость протокола зависит от размера группы q алгоритма Диффи — Хеллмана, знания q-степени экспоненты и q-строгих предположений Диффи — Хеллмана [31, 79, 80] для q, полиномиально зависимого от размера схемы.

Алгоритм формирования ключей G

На вход принимается арифметическая схема $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ и выводятся ключи доказывающего pk и верификатора vk.

1. Вычисляются $(\mathbf{A}, \mathbf{B}, \mathbf{C}, Z) = \mathsf{QAPinst}(C)$, при этом $\mathbf{A}, \mathbf{B}, \mathbf{C}$ расширяются за счёт следующих значений:

$$A_{m+1} = B_{m+2} = C_{m+3} = Z,$$
$$A_{m+2} = A_{m+3} = B_{m+1} = B_{m+3} = C_{m+1} = C_{m+2} = 0.$$

2. Производится случайная выборка $\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \beta, \gamma \in \mathbb{F}_r$.

3. Устанавливается ключ $\mathsf{pk} = (C, pk_A, pk'_A, pk_B, pk'_B, pk_C, pk'_C, pk_K, pk_H)$, где

$$\begin{gathered} pk_A = \{A_i(\tau)\rho_A P_1\}_{i=0}^{m+3},\ pk'_A = \{A_i(\tau)\alpha_A\rho_A P_1\}_{i=n+1}^{m+3}, \\ pk_B = \{B_i(\tau)\rho_B P_2\}_{i=0}^{m+3},\ pk'_B = \{B_i(\tau)\alpha_B\rho_B P_1\}_{i=0}^{m+3}, \\ pk_C = \{C_i(\tau)\rho_A\rho_B P_1\}_{i=0}^{m+3},\ pk'_C = \{C_i(\tau)\alpha_C\rho_A\rho_B P_1\}_{i=0}^{m+3}, \\ pk_K = \{\beta(A_i(\tau)\rho_A + B_i(\tau)\rho_B + C_i(\tau)\rho_A\rho_B)P_1\}_{i=0}^{m+3}, \\ pk_H = \{\tau^i P_1\}_{i=1}^{d}\ . \end{gathered} \tag{36}$$

По сравнению с составляющими ключа доказывающего протокола Р. Джентри и др. [37] в выражениях (9) и (8) ключ доказывающего [65] в выражении (36) содержит на три элемента больше для каждого $pk_A, pk_B, pk_C, pk'_A, pk'_B, pk'_C, pk_K$, а смещения $\alpha_A, \alpha_B, \alpha_C$ для pk'_A, pk'_B, pk'_C применяются только по индексам $i = = 0, \ldots, m+3$.

4. Устанавливается ключ $\mathsf{vk} = (vk_A, vk_B, vk_C, vk_\gamma, vk^1_{\beta\gamma}, vk^2_{\beta\gamma}, vk_Z, vk_{IC})$, где

$$\begin{gathered} vk_A = \alpha_A P_2,\ vk_B = \alpha_B P_1,\ vk_C = \alpha_C P_2, \\ vk_\gamma = \gamma P_2,\ vk^1_{\beta\gamma} = \gamma\beta P_1,\ vk^2_{\beta\gamma} = \gamma\beta P_2, \\ vk_Z = Z(\tau)\rho_A\rho_B P_2,\ vk_{IC} = (A_i(\tau)\rho_A P_1)_{i=0}^{n}. \end{gathered}$$

5. Выводятся ключи доказывающего и верификатора $(\mathsf{pk}, \mathsf{vk})$.

При вызове схемы $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^l$ с a проводами и b билинейными вентилями формирователь ключей выводит pk с $(6a+b+l+25)$ элементами $\mathbb{G}_1$ и $(a+4)$ элементами $\mathbb{G}_2$, vk с $(n+3)$ элементами $\mathbb{G}_1$ и пятью элементами $\mathbb{G}_2$.

Алгоритм доказывающего P

Принимается ключ pk, вход $\mathbf{x} \in \mathbb{F}_r^n$, секретное свидетельство $\mathbf{a} \in \mathbb{F}_r^h$ и выводится доказательство π:

1. Вычисляются $(\mathbf{A}, \mathbf{B}, \mathbf{C}, Z) = \mathsf{QAPinst}(C)$.
2. Вычисляется $\mathbf{s} = \mathsf{QAPinst}(C, \mathbf{x}, \mathbf{a}) \in \mathbb{F}_r^m$.
3. Производится случайная выборка $\delta_1, \delta_2, \delta_1 \in \mathbb{F}_r$.
4. Вычисляется $\mathbf{h} = (h_0, h_1, \ldots, h_d) \in \mathbb{F}_r^{d+1}$, который содержит коэффициенты $H(z) = (A(z)B(z) - C(z))/Z(z)$, где $A, B, C \in \mathbb{F}_r[z]$ ($\mathbb{F}_r[z]$ — кольцо многочленов над $\mathbb{F}$ от одной переменной) следующего вида:

$$\begin{aligned} A(z) &= A_0(z) + \sum_{i=1}^{m} s_i A_i(z) + \delta_1 Z(z), \\ B(z) &= B_0(z) + \sum_{i=1}^{m} s_i B_i(z) + \delta_2 Z(z), \\ C(z) &= C_0(z) + \sum_{i=1}^{m} s_i C_i(z) + \delta_3 Z(z). \end{aligned}$$

5. Устанавливается ключ $p\tilde{k}_A$, соответствующий pk_A с обнулением $pk_{A,i} = 0$ для $i = 0, 1, \ldots, n$. Ключ $p\tilde{k}'_A$ соответствует pk'_A, но с добавлением $n+1$ нулей.
6. Фиксируется $\mathbf{c} = (1, \mathbf{s}, \delta_1, \delta_2, \delta_3) \in \mathbb{F}_r^{4+m}$ и вычисляются

$$\begin{gathered} \pi_A = \langle \mathbf{c}, p\tilde{k}_A \rangle,\ \pi'_A = \langle \mathbf{c}, p\tilde{k}'_A \rangle,\ \pi_B = \langle \mathbf{c}, pk_B \rangle,\ \pi'_B = \langle \mathbf{c}, pk'_B \rangle, \\ \pi_C = \langle \mathbf{c}, pk_C \rangle,\ \pi'_C = \langle \mathbf{c}, pk'_C \rangle,\ \pi_K = \langle \mathbf{c}, pk_K \rangle,\ \pi_H = \langle \mathbf{h}, pk_H \rangle. \end{gathered}$$

7. Выводится доказательство $\pi = (\pi_A, \pi'_A, \pi_B, \pi'_B, \pi_C, \pi'_C, \pi_K, \pi_H)$, содержащее семь элементов $\mathbb{G}_1$ и один элемент $\mathbb{G}_2$.

Алгоритм верификатора V

Принимается ключ vk, вход $\mathbf{x} \in \mathbb{F}_r^n$, доказательство π и выводится бит решения:

1. Вычисляется $vk_{\mathbf{x}} = vk_{IC,0} + \sum_{i=1}^{n} x_i vk_{IC,i} \in \mathbb{G}_1$.
2. Проверяется подлинность обязательств для A, B, C:

$$e(\pi_A, vk_A) = e(\pi'_A, P_2),\ e(vk_B, \pi_B) = e(\pi'_B, P_2),\ e(\pi_C, vk_C) = e(\pi'_C, P_2)\ .$$

3. Проверяется, что использовались аналогичные коэффициенты:

$$e(\pi_K, vk_\gamma) = e(vk_{\mathbf{x}} + \pi_A + \pi_C, vk^2_{\beta\gamma}) e(vk^1_{\beta\gamma}, \pi_B)\ .$$

4. Проверяется делимость QAP:

$$e(vk_{\mathbf{x}} + \pi_A, \pi_B) = e(\pi_H, vk_Z) e(\pi_C, P_2).$$

5. Доказательство принимается, если все перечисленные проверки верны.

Заключение

Рассмотрены базовые примеры протоколов DV [22, 31, 32, 37, 42, 55, 63] и PV [37, 49, 55, 63] zk-SNARK, для которых представлены алгоритмы формирования ключей, доказательств достоверности вычислений и их верификации. Описано формирование публичных и секретных параметров в виде ключей доказательства и верификации, публикуемых в форме главных ссылочных строк [32, 37, 41, 52, 63, 74, 81] и др. Представлены варианты протоколов zk-SNARK для выполнимости дискретных функций с заданными значениями выходов, связанными с открытыми и секретными входами. Рассмотрены алгебраические преобразования, сводящие задачу проверки корректности вычисления дискретных функций к проверке множества полиномиальных уравнений низкой степени, которые получили наименования QAP, SAP, QSP, SSP, QPP [37, 50–52, 55] и др. Для передачи информации о полиноме достаточно передать его значение, вычисленное в секретной точке [4], в результате чего полином произвольной степени сводится к одному значению поля. Протоколы zk-SNARK также строятся для схем с распределёнными и аутентифицированными данными [63].

Применяемые алгоритмы используют широкий спектр различных криптографических преобразований, основанных на задачах RSA, Диффи — Хеллмана, а также на различных вариациях задач о знании экспонент [37, 76]. Кроме того, используются цифровые подписи, схемы гомоморфного шифрования [44, 67, 77], билинейные спаривания на эллиптических кривых и другие криптографические примитивы.

В таблице рассмотренные протоколы zk-SNARK классифицированы по применяемому до криптографических преобразований математическому аппарату, используемым криптографическим примитивам, схемам верификации и решаемым задачам. Для наглядности вариации криптографических преобразований, связанных с задачей Диффи — Хеллмана и знанием экспонент, обозначены DH, билинейное спаривание — BP, эллиптические кривые — ECC, полностью гомоморфное шифрование — FHE, секретная/публичная верификация — PV/DV. Решаемые протоколами zk-SNARK задачи представлены верифицируемыми вычислениями VC и доказательствами знания.

Общим для всех рассмотренных протоколов zk-SNARK является использование публичной CRS, ключей доказательства и верификации. Защищённость практически всех рассмотренных протоколов основана на вариантах задачи Диффи — Хеллмана, знании экспонент и билинейном спаривании. Тем не менее для решения прикладных

задач с использованием поразрядных операций целесообразнее выбирать протоколы zk-SNARK с полиномиальными наборами QSP/SSP. Набор полиномов QAP повышает быстродействие для случая работы с целыми элементами поля. Более специфичным вариантом является набор QPP, где провода представлены многочленами, что также может повысить производительность.

Сравнительный анализ рассмотренных протоколов zk-SNARK

Протокол	Матем. аппарат	Криптопримитивы	Сх. вериф.	Реш. задача
Й. Грот [31]	Перестановка	DH, BP, ECC	PV	Знания
Р. Дженнаро и др. [34]	QAP	FHE, симметр. алг.	DV	VC
П. Фаузи и др. [41]	Циклический сдвиг	DH, BP, ECC	PV	Знания
Р. Дженнаро и др. [37]	QSP, QAP	RSA, DH, BP, ECC	PV, DV	Знания
Б. Парно и др. [49]	QAP	DH, BP, ECC	PV	VC
Х. Липмаа [52]	QSP, SSP, QAP, SAP	DH, BP, ECC	PV	Знания
Х. Липмаа [52]	Циклический сдвиг	DH, BP, ECC	PV	Знания
А. Косба и др. [55]	QPP	DH, BP, ECC	PV, DV	Знания
Г. Данезис и др. [51]	SSP	DH, BP, ECC	PV	Знания
К. Костелло и др. [62]	QAP	DH, BP, ECC	PV	VC
М. Бакес и др. [63]	QAP	DH, BP, ECC, FHE	PV, DV	Знания
Й. Грот [74]	QAP	DH, BP, ECC	PV, DV	Знания
Э. Бен-Сассон и др. [54]	QAP	FHE	PV	Знания
Э. Бен-Сассон и др. [65]	QAP	DH, BP, ECC	PV	Знания

Несмотря на общирный список представленных протоколов zk-SNARK, он не является исчерпывающим, а содержит исторически базовые конструкции. Отдельными вопросами являются анализ способов повышения производительности и выбор наиболее исследованного с точки зрения безопасности протокола. В рамках практической реализации необходим выбор протокола zk-SNARK с учётом ресурсов конкретной распределённой вычислительной системы и на основе таких параметров, как трудозатраты на формирование главной ссылочной строки, построение и верификацию доказательств, а также размер главной ссылочной строки и доказательств.

ЛИТЕРАТУРА

1. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5 proposal], 2021. 213 p. `https://raw.githubusercontent.com/Zcash/zips/master/protocol/protocol.pdf`.
2. *Черемушкин А. В.* Криптографические протоколы. Основные свойства и уязвимости. М.: Изд. центр «Академия», 2009. 272 с.
3. *Запечников С. В.* Криптографическая защита процессов обработки информации в недоверенной среде: достижения, проблемы, перспективы // Вестник современных цифровых технологий. 2019. № 1. С. 4–16.
4. *Petkus M.* Why and How zk-SNARK Works. ArXiv, abs/1906.07221. 2019. 65 p. `https://arxiv.org/pdf/1906.07221.pdf`.
5. *Goldreich O., Micali S., and Wigderson A.* How to play any mental game or a completeness theorem for protocols with honest majority // Proc. STOC'87. N.Y.: ACM, 1987. P. 218–229.
6. *Ben-Or M., Goldwasser S., and Wigderson A.* Completeness theorems for non-cryptographic fault-tolerant distributed computation // Proc. STOC'88. N.Y.: ACM, 1988. P. 1–10.
7. *Gueron S., Lindell Y., Nof A., and Pinkas B.* Fast garbling of circuits under standard assumptions // Proc. CCS'15. N.Y.: ACM, 2015. P. 567–578.
8. *Wang X., Ranellucci S., and Katz J.* Global-scale Secure Multiparty Computation. IACR eprint Archive. 2017. 35 p. `https://eprint.iacr.org/2017/189.pdf`.

9. *Boyen X. and Waters B.* Compact group signatures without random oracles // LNCS. 2006. V. 4004. P. 427–444.
10. *Chase M., Kohlweiss M., Lysyanskaya A., and Meiklejohn S* Malleable proof systems and applications // LNCS. 2012. V. 7237. P. 281–300.
11. *Belenkiy M., Chase M., Kohlweiss M., and Lysyanskaya A.* P-signatures and noninteractive anonymous credentials // LNCS. 2008. V. 4948. P. 356–374.
12. *Belenkiy M., Camenisch J., Chase M., et al.* Randomizable proofs and delegatable anonymous credentials // LNCS. 2009. V. 5677. P. 108–125.
13. *Katz J., Myers S., and Ostrovsky R.* Cryptographic counters and applications to electronic voting // LNCS. 2001. V. 2045. P. 78–92.
14. *Lipmaa H.* Two Simple Code-Verification Voting Protocols. Cryptology ePrint Archive. Report 2011/317. 2011. https://eprint.iacr.org/2011/317.
15. *Konda C., Connor M., Westland D., et al.* Nightfall. https://raw.githubusercontent.com/EYBlockchain/nightfall/master/doc/whitepaper/nightfall-v1.pdf. 2019.
16. *Diamond B.* ZSL Proof of Concept. https://github.com/ConsenSys/quorum/wiki/ZSL#protocol.
17. Quorum — ZSL Integration: Proof of Concept. Technical Design Document. 23 p. https://github.com/ConsenSys/zsl-q/blob/master/docs/ZSL-Quorum-POC_TDD_v1.3pub.pdf.
18. Welcome to zkSync. https://zksync.io/faq/.
19. *Doreian B. W. and Erickson R. R.* PIVX: Protected Instant Verified Transaction. 25 p. https://pivx.org/files/whitepapers/PIVX_Non_Technical_Whitepaper_v2.0.pdf.
20. *Pertsev A., Semenov R., and Storm R.* Tornado Cash Privacy Solution. Version 1.4. 2019. 3 p. https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf.
21. *Danezis G., Fournet C., Kohlweiss M., and Parno B.* Pinocchio coin: building zerocoin from a succinct pairing-based proof system // Proc. PETShop'13. N.Y.: ACM, 2013. P. 27–30.
22. *Bitansky N., Canetti R., Chiesa A., et al.* The hunting of the SNARK // J. Cryptology. 2016. No. 30. P. 989–1066.
23. *Kolesnikov V., Kumaresan R., Rosulek M., and Trieu N.* Efficient batched oblivious PRF with applications to private set intersection // Proc. CCS'2016. N.Y.: ACM, 2016. P. 818–829.
24. *Rindal P. and Rosulek M.* Malicious-secure private set intersection via dual execution // Proc. CCS'2017. N.Y.: ACM, 2017. P. 1229–1242.
25. *Ion M., Kreuter B., Nergiz E., et al.* Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. IACR eprint Archive. 2017. 14 p. https://eprint.iacr.org/2017/738.
26. *Bonawitz K., Ivanov V., Kreuter B., et al.* Practical secure aggregation for federated learning on user-held data // Proc. NIPS'2016. N.Y.: ACM, 2016. P. 1175–1191.
27. *Corrigan-Gibbs H. and Boneh D.* Prio: Private, robust, and scalable computation of aggregate statistics // Proc. NSDI'17. USENIX Ass., 2017. P. 259–282.
28. *Angel S., Chen H., Laine K., and Setty S.* PIR with Compressed Queries and Amortized Query Processing. IACR eprint Archive. 2017. 18 p. https://eprint.iacr.org/2017/1142.pdf.
29. *Cheu A., Smith A., Ullman J., et al.* Distributed differential privacy via shuffling // LNCS. 2019. V. 11476. P. 375–403.
30. *Bittau A., Erlingsson Ú., Maniatis P., et al.* PROCHLO: Strong privacy for analytics in the crowd // Proc. SOSP'17. N.Y.: ACM, 2017. P. 441–459.
31. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments // LNCS. 2010. V. 6477. P. 321–340.

32. *Lipmaa H.* Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments // LNCS. 2012. V. 7194. P. 169–189.
33. *Tao T. and Vu V.* Additive Combinatorics. Cambridge Studies in Advanced Mathematics. No. 105. Cambridge University Press, 2006. 532 p.
34. *Gennaro R., Gentry C., and Parno B.* Non-interactive verifiable computing: Outsourcing computation to untrusted workers // LNCS. 2010. V. 6223. P. 465–482.
35. *Yao A.* Protocols for secure computations // 23rd Ann. Symp. Foundations of Computer Science. 1982. P. 160–164.
36. *Yao A.* How to generate and exchange secrets // 27th Ann. Symp. Foundations of Computer Science. 1986. P. 162–167.
37. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs // LNCS. 2013. V. 7881. P. 626–645.
38. *Pankova A.* Succinct Non-Interactive Arguments from Quadratic Arithmetic Programs. Technical report. University of Tartu. Cybernetica AS, 2013. 28 p.
39. *Parno B., Raykova M., and Vaikuntanathan V.* How to delegate and verify in public: Verifiable computation from attribute-based encryption // LNCS. 2012. V. 7194. P. 422–439.
40. *Groth J., Ostrovsky R., and Sahai A.* Perfect non-interactive zero knowledge for NP // LNCS. 2006. V. 4004. P. 339–358.
41. *Fauzi P., Lipmaa H., and Zhang B.* Efficient modular NIZK arguments from shift and product // LNCS. 2013. V. 8257. P. 92–121.
42. *Reitwiebner C.* zkSNARKs in a Nutshell. 2017. 15 p. `https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/`.
43. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* Recursive Composition and Bootstrapping for Snarks and Proof-Carrying data. IACR Cryptology ePrint Archive. 2012. 61 p. `http://eprint.iacr.org/2012/095`.
44. *Boneh D., Segev G., and Waters B.* Targeted malleability: homomorphic encryption for restricted computations // Proc. ITCS'12. N.Y.: ACM, 2012. P. 350–366.
45. *Valiant P.* Incrementally verifiable computation or proofs of knowledge imply time/space efficiency // LNCS. 2008. V. 4948. P. 1–18.
46. *Paillier P.* Public-key cryptosystems based on composite degree residuosity classes // LNCS. 1999. V. 1592. P. 223–238.
47. *Лось А. Б., Нестеренко А. Ю., Рожков М. И.* Криптографические методы защиты информации: учеб. для академического бакалавриата. 2-е изд. М.: Юрайт, 2017. 473 с.
48. *Chase M., Kohlweiss M., Lysyanskaya A., and Meiklejohn S.* Malleable proof systems and applications // LNCS. 2012. V. 7237. P. 281–300.
49. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation // Proc. 34th IEEE Symp. Security and Privacy. Oakland, 2013. P. 238–252.
50. *Lipmaa H.* Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes // LNCS. 2013. V. 8269. P. 41–60.
51. *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.
52. *Lipmaa H.* Almost Optimal Short Adaptive Non-Interactive Zero Knowledge. Tech. Rep. 2014/396. IACR, 2014. 20 p. `http://eprint.iacr.org/2014/396`.
53. *Blum M., Feldman P., and Micali S.* Non-interactive zero-knowledge and its applications // Proc. STOC'88. N.Y.: ACM, 1988. P. 103–112.
54. *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.

55. *Kosba A. E., Papadopoulos D., Papamanthou C., et al.* TRUESET: Nearly practical verifiable set computations. Cryptology ePrint Archive. Report 2014/160. 2014. 30 p. http://eprint.iacr.org/2014/160.
56. *Shoup V.* A new polynomial factorization algorithm and its implementation // J. Symbolic Computation. 1995. V. 20. No. 4. P. 363–397.
57. *Shoup V.* NTL: Number Theory Library. http://www.shoup.net/ntl/.
58. *Granlund T.* GMP: The GNU Multiple Precision Arithmetic Library. 2006. http://gmplib.org/.
59. *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.
60. *Kissner L. and Song D.* Privacy-preserving set operations // LNCS. 2005. V. 3621. P. 241–257.
61. *Papamanthou C., Tamassia R., and Triandopoulos N.* Optimal verification of operations on dynamic sets. Cryptology ePrint Archive. Paper 2010/455. 2010. 31 p. https://eprint.iacr.org/2010/455.
62. *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.
63. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data // Proc. IEEE Symp. SP'15. IEEE Computer Society, USA, 2015. P. 271–286.
64. *Fournet C., Kohlweiss M., Danezis G., and Luo Z.* ZQL: A compiler for privacy-preserving data processing // Proc. 22nd USENIX Conf. SEC'13. USENIX Ass., 2013. P. 163–178.
65. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Updated version, 2019. 37 p. https://eprint.iacr.org/2013/879.pdf.
66. *Backes M., Fiore D., and Reischuk R. M.* Verifiable delegation of computation on outsourced data // Proc. CCS'13. N.Y.: ACM, 2013. P. 863–874.
67. *Gennaro R. and Wichs D.* Fully homomorphic message authenticators // LNCS. 2013. V. 8270. P. 301–320.
68. *Camenisch J., Kohlweiss M., and Soriente C.* An accumulator based on bilinear maps and efficient revocation for anonymous credentials // LNCS. 2009. V. 5443. P. 481–500.
69. *Bernstein D. J., Duif N., Lange T., et al.* High-speed high-security signatures // J. Cryptogr. Engineering. 2012. V. 2. No. 2. P. 77–89.
70. Supercop. http://bench.cr.yp.to/supercop.html.
71. *Bowe S., Gabizon A., and Green M. D.* A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive. Report 2017/602. 2017. 25 p. https://ia.cr/2017/602.
72. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct non-interactive zero knowledge for a von Neumann architecture // Proc. 23rd USENIX Security Symp. San Diego, CA, USA, 2014. P. 781–796.
73. *Gabizon A.* On the security of the BCTV Pinocchio zk-SNARK variant. Cryptology ePrint Archive. Paper 2019/119. 2019. 9 p. https://eprint.iacr.org/2019/119.
74. *Groth J.* On the size of pairing-based non-interactive arguments // LNCS. 2016. V. 9666. P. 305–326.
75. *Galbraith S. D., Paterson K. G., and Smart N. P.* Pairings for cryptographers // Discr. Appl. Math. 2008. V. 156. Iss. 16. P. 3113–3121.
76. *Bitansky N., Chiesa A., Ishai Y., et al.* Succinct non-interactive arguments via linear interactive proofs // LNCS. 2013. V. 7785. P. 315–333.

77. *Armknecht F., Boyd C., Carr C., et al.* Guide to Fully Homomorphic Encryption. Cryptology ePrint Archive. 35 p. https://eprint.iacr.org/2015/1192.pdf.
78. *Gabizon A.* On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Report 2019/119. 2019. 9 p. https://ia.cr/2019/119.
79. *Boneh D. and Boyen X.* Secure identity based encryption without random oracles // LNCS. 2004. V. 3152. P. 443–459.
80. *Gennaro R.* Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks // LNCS. 2004. V. 3152. P. 220–236.
81. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs // Proc. IEEE Symp. Security and Privacy, San Jose, CA, USA, 2015. P. 287–304.

REFERENCES

1. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5 proposal], 2021. 213 p. https://raw.githubusercontent.com/Zcash/zips/master/protocol/protocol.pdf.
2. *Cheremushkin A. V.* Kriptograficheskie protokoly. Osnovnye svoystva i uyazvimosti [Cryptographic Protocols. Basic Properties and Vulnerabilities]. Moscow, Akademiya Publ., 2009. 272 p. (in Russian)
3. *Zapechnikov S. V.* Kriptograficheskaya zashchita protsessov obrabotki informatsii v nedoverennoy srede: dostizheniya, problemy, perspektivy [Cryptographic protection of information processing processes in an untrusted environment: achievements, problems, prospects]. Vestnik Sovremennykh Tsifrovykh Tekhnologiy, 2019, no. 1, pp. 4–16. (in Russian)
4. *Petkus M.* Why and How zk-SNARK Works. ArXiv, abs/1906.07221. 2019. 65 p. https://arxiv.org/pdf/1906.07221.pdf.
5. *Goldreich O., Micali S., and Wigderson A.* How to play any mental game or a completeness theorem for protocols with honest majority. Proc. STOC'87, N.Y., ACM, 1987, pp. 218–229.
6. *Ben-Or M., Goldwasser S., and Wigderson A.* Completeness theorems for non-cryptographic fault-tolerant distributed computation. Proc. STOC'88, N.Y., ACM, 1988, pp. 1–10.
7. *Gueron S., Lindell Y., Nof A., and Pinkas B.* Fast garbling of circuits under standard assumptions. Proc. CCS'15, N.Y., ACM, 2015, pp. 567–578.
8. *Wang X., Ranellucci S., and Katz J.* Global-scale Secure Multiparty Computation. IACR eprint Archive, 2017. 35 p. https://eprint.iacr.org/2017/189.pdf.
9. *Boyen X. and Waters B.* Compact group signatures without random oracles, LNCS, 2006, vol. 4004, pp. 427–444.
10. *Chase M., Kohlweiss M., Lysyanskaya A., and Meiklejohn S* Malleable proof systems and applications. LNCS, 2012, vol. 7237, pp. 281–300.
11. *Belenkiy M., Chase M., Kohlweiss M., and Lysyanskaya A.* P-signatures and noninteractive anonymous credentials. LNCS, 2008, vol. 4948, pp. 356–374.
12. *Belenkiy M., Camenisch J., Chase M., et al.* Randomizable proofs and delegatable anonymous credentials. LNCS, 2009, vol. 5677, pp. 108–125.
13. *Katz J., Myers S., and Ostrovsky R.* Cryptographic counters and applications to electronic voting. LNCS, 2001, vol. 2045, pp. 78–92.
14. *Lipmaa H.* Two Simple Code-Verification Voting Protocols. Cryptology ePrint Archive. Report 2011/317, 2011. https://eprint.iacr.org/2011/317.
15. *Konda C., Connor M., Westland D., et al.* Nightfall. https://raw.githubusercontent.com/EYBlockchain/nightfall/master/doc/whitepaper/nightfall-v1.pdf. 2019.

16. *Diamond B.* ZSL Proof of Concept. https://github.com/ConsenSys/quorum/wiki/ZSL#protocol.
17. Quorum — ZSL Integration: Proof of Concept. Technical Design Document. 23 p. https://github.com/ConsenSys/zsl-q/blob/master/docs/ZSL-Quorum-POC_TDD_v1.3pub.pdf.
18. Welcome to zkSync. https://zksync.io/faq/.
19. *Doreian B. W. and Erickson R. R.* PIVX: Protected Instant Verified Transaction. 25 p. https://pivx.org/files/whitepapers/PIVX_Non_Technical_Whitepaper_v2.0.pdf.
20. *Pertsev A., Semenov R., and Storm R.* Tornado Cash Privacy Solution. Version 1.4. 2019. 3 p. https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf.
21. *Danezis G., Fournet C., Kohlweiss M., and Parno B.* Pinocchio coin: building zerocoin from a succinct pairing-based proof system. Proc. PETShop'13, N.Y., ACM, 2013, pp. 27–30.
22. *Bitansky N., Canetti R., Chiesa A., et al.* The hunting of the SNARK. J. Cryptology, 2016, no. 30, pp. 989–1066.
23. *Kolesnikov V., Kumaresan R., Rosulek M., and Trieu N.* Efficient batched oblivious PRF with applications to private set intersection. Proc. CCS'2016, N.Y., ACM, 2016, pp. 818–829.
24. *Rindal P. and Rosulek M.* Malicious-secure private set intersection via dual execution. Proc. CCS'2017, N.Y., ACM, 2017, pp. 1229–1242.
25. *Ion M., Kreuter B., Nergiz E., et al.* Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. IACR eprint Archive, 2017. 14 p. https://eprint.iacr.org/2017/738.
26. *Bonawitz K., Ivanov V., Kreuter B., et al.* Practical secure aggregation for federated learning on user-held data. Proc. NIPS'2016, N.Y., ACM, 2016, pp. 1175–1191.
27. *Corrigan-Gibbs H. and Boneh D.* Prio: Private, robust, and scalable computation of aggregate statistics. Proc. NSDI'17, USENIX Ass., 2017, pp. 259–282.
28. *Angel S., Chen H., Laine K., and Setty S.* PIR with Compressed Queries and Amortized Query Processing. IACR eprint Archive, 2017. 18 p. https://eprint.iacr.org/2017/1142.pdf.
29. *Cheu A., Smith A., Ullman J., et al.* Distributed differential privacy via shuffling. LNCS, 2019, vol. 11476, pp. 375–403.
30. *Bittau A., Erlingsson Ú., Maniatis P., et al.* PROCHLO: Strong privacy for analytics in the crowd. Proc. SOSP'17, N.Y., ACM, 2017, pp. 441–459.
31. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments. LNCS, 2010, vol. 6477, pp. 321–340.
32. *Lipmaa H.* Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. LNCS, 2012, vol. 7194, pp. 169–189.
33. *Tao T. and Vu V.* Additive Combinatorics. Cambridge Studies in Advanced Mathematics, no. 105. Cambridge University Press, 2006. 532 p.
34. *Gennaro R., Gentry C., and Parno B.* Non-interactive verifiable computing: Outsourcing computation to untrusted workers. LNCS, 2010, vol. 6223, pp. 465–482.
35. *Yao A.* Protocols for secure computations. 23rd Ann. Symp. Foundations of Computer Science, 1982, pp. 160–164.
36. *Yao A.* How to generate and exchange secrets. 27th Ann. Symp. Foundations of Computer Science, 1986, pp. 162–167.
37. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs. LNCS, 2013, vol. 7881, pp. 626–645.
38. *Pankova A.* Succinct Non-Interactive Arguments from Quadratic Arithmetic Programs. Technical report, University of Tartu. Cybernetica AS, 2013. 28 p.

39. *Parno B., Raykova M., and Vaikuntanathan V.* How to delegate and verify in public: Verifiable computation from attribute-based encryption. LNCS, 2012, vol. 7194, pp. 422–439.
40. *Groth J., Ostrovsky R., and Sahai A.* Perfect non-interactive zero knowledge for NP. LNCS, 2006, vol. 4004, pp. 339–358.
41. *Fauzi P., Lipmaa H., and Zhang B.* Efficient modular NIZK arguments from shift and product. LNCS, 2013, vol. 8257, pp. 92–121.
42. *Reitwiebner C.* zkSNARKs in a Nutshell. 2017. 15 p. `https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/`.
43. *Bitansky N., Canetti R., Chiesa A., and Tromer E.* Recursive Composition and Bootstrapping for Snarks and Proof-Carrying data. IACR Cryptology ePrint Archive, 2012. 61 p. `http://eprint.iacr.org/2012/095`.
44. *Boneh D., Segev G., and Waters B.* Targeted malleability: homomorphic encryption for restricted computations. Proc. ITCS'12, N.Y., ACM, 2012, pp. 350–366.
45. *Valiant P.* Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. LNCS, 2008, vol. 4948, pp. 1–18.
46. *Paillier P.* Public-key cryptosystems based on composite degree residuosity classes. LNCS, 1999, vol. 1592, pp. 223–238.
47. *Los' A. B., Nesterenko A. Yu., and Rozhkov M. I.* Kriptograficheskie metody zashchity informatsii: uchebnik dlya akademicheskogo bakalavriata [Cryptographic Methods of Information Protection: Textbook for Academic Undergraduate]. Moscow, Yurayt Publ., 2017. 473 p. (in Russian)
48. *Chase M., Kohlweiss M., Lysyanskaya A., and Meiklejohn S.* Malleable proof systems and applications. LNCS, 2012, vol. 7237, pp. 281–300.
49. *Parno B., Howell J., Gentry C., and Raykova M.* Pinocchio: Nearly practical verifiable computation. Proc. 34th IEEE Symp. Security and Privacy, Oakland, 2013, pp. 238–252.
50. *Lipmaa H.* Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. LNCS, 2013, vol. 8269, pp. 41–60.
51. *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.
52. *Lipmaa H.* Almost Optimal Short Adaptive Non-Interactive Zero Knowledge. Tech. Rep. 2014/396, IACR, 2014. 20 p. `http://eprint.iacr.org/2014/396`.
53. *Blum M., Feldman P., and Micali S.* Non-interactive zero-knowledge and its applications. Proc. STOC'88, N.Y., ACM, 1988, pp. 103–112.
54. *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.
55. *Kosba A. E., Papadopoulos D., Papamanthou C., et al.* TRUESET: Nearly practical verifiable set computations. Cryptology ePrint Archive. Report 2014/160, 2014. 30 p. `http://eprint.iacr.org/2014/160`.
56. *Shoup V.* A new polynomial factorization algorithm and its implementation. J. Symbolic Computation, 1995, vol. 20, no. 4, pp. 363–397.
57. *Shoup V.* NTL: Number Theory Library. `http://www.shoup.net/ntl/`.
58. *Granlund T.* GMP: The GNU Multiple Precision Arithmetic Library. 2006. `http://gmplib.org/`.
59. *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.
60. *Kissner L. and Song D.* Privacy-preserving set operations. LNCS, 2005, vol. 3621, pp. 241–257.

61. *Papamanthou C., Tamassia R., and Triandopoulos N.* Optimal verification of operations on dynamic sets. Cryptology ePrint Archive. Paper 2010/455, 2010. 31 p. https://eprint.iacr.org/2010/455.
62. *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.
63. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. Proc. IEEE Symp. SP'15, IEEE Computer Society, USA, 2015, pp. 271–286.
64. *Fournet C., Kohlweiss M., Danezis G., and Luo Z.* ZQL: A compiler for privacy-preserving data processing // Proc. 22nd USENIX Conf. SEC'13. USENIX Ass., 2013. P. 163–178.
65. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. Updated version, 2019. 37 p. https://eprint.iacr.org/2013/879.pdf.
66. *Backes M., Fiore D., and Reischuk R. M.* Verifiable delegation of computation on outsourced data. Proc. CCS'13, N.Y., ACM, 2013, pp. 863–874.
67. *Gennaro R. and Wichs D.* Fully homomorphic message authenticators. LNCS, 2013, vol. 8270, pp. 301–320.
68. *Camenisch J., Kohlweiss M., and Soriente C.* An accumulator based on bilinear maps and efficient revocation for anonymous credentials. LNCS, 2009, vol. 5443, pp. 481–500.
69. *Bernstein D. J., Duif N., Lange T., et al.* High-speed high-security signatures. J. Cryptogr. Engineering, 2012, vol. 2, no. 2, pp. 77–89.
70. Supercop. http://bench.cr.yp.to/supercop.html.
71. *Bowe S., Gabizon A., and Green M. D.* A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive. Report 2017/602, 2017. 25 p. https://ia.cr/2017/602.
72. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Succinct non-interactive zero knowledge for a von Neumann architecture. Proc. 23rd USENIX Security Symp., San Diego, CA, USA, 2014, pp. 781–796.
73. *Gabizon A.* On the security of the BCTV Pinocchio zk-SNARK variant. Cryptology ePrint Archive. Paper 2019/119, 2019. 9 p. https://eprint.iacr.org/2019/119.
74. *Groth J.* On the size of pairing-based non-interactive arguments. LNCS, 2016, vol. 9666, pp. 305–326.
75. *Galbraith S. D., Paterson K. G., and Smart N. P.* Pairings for cryptographers. Discr. Appl. Math., 2008, vol. 156, iss. 16, pp. 3113–3121.
76. *Bitansky N., Chiesa A., Ishai Y., et al.* Succinct non-interactive arguments via linear interactive proofs. LNCS, 2013, vol. 7785, pp. 315–333.
77. *Armknecht F., Boyd C., Carr C., et al.* Guide to Fully Homomorphic Encryption. Cryptology ePrint Archive. 35 p. https://eprint.iacr.org/2015/1192.pdf.
78. *Gabizon A.* On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Report 2019/119, 2019. 9 p. https://ia.cr/2019/119.
79. *Boneh D. and Boyen X.* Secure identity based encryption without random oracles. LNCS, 2004, vol. 3152, pp. 443–459.
80. *Gennaro R.* Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. LNCS, 2004, vol. 3152, pp. 220–236.
81. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs. Proc. IEEE Symp. Security and Privacy, San Jose, CA, USA, 2015, pp. 287–304.