МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 003.26 + 004.056 + 001.99 DOI 10.17223/20710410/61/3

ЗАЩИЩЁННОЕ ФОРМИРОВАНИЕ ПУБЛИЧНЫХ ПАРАМЕТРОВ И УСТРАНЕНИЕ УЯЗВИМОСТЕЙ КРАТКИХ НЕИНТЕРАКТИВНЫХ АРГУМЕНТОВ С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

И. В. Мартыненков

АО «КВАНТ-ТЕЛЕКОМ», г. Москва, Россия

E-mail: mivpost@yandex.ru

Рассматриваются известные методы устранения уязвимостей кратких неинтерактивных аргументов с нулевым разглашением на основе корректировки уравнений верификации доказательств, значений публичных параметров в виде главных ссылочных строк и ключей формирования доказательств. Описаны способы защищённого формирования главных ссылочных строк с использованием доверенной третьей стороны и многостороннего взаимодействия.

Ключевые слова: *краткие неинтерактивные аргументы, публичные параметры, главные ссылочные строки, защищённость.*

SECURE FORMATION OF PUBLIC PARAMETERS AND ELIMINATION OF VULNERABILITIES OF ZERO-KNOWLEDGE SUCCINT NON-INTERACTIVE ARGUMENTS OF KNOWLEDGE

I. V. Martynenkov

JSC “KVANT-TELECOM”, Moscow, Russia

The methods of eliminating vulnerabilities of zero-knowledge succinct non-interactive arguments of knowlede are considered. The methods are based on the security of public parameters formation in the form of common reference strings using a trusted third party and multilateral interaction. The multilateral formation of the common reference strings uses the only honest party with a fixed and unlimited set of participants, as well as verification of the reliability of the results. Examples of increasing the level of security of zero-knowledge succinct non-interactive arguments of knowledge based on the correction of proof verification equations and the values of the common reference strings, eliminating redundant elements from the common reference strings and the keys of proof formation are given. The protocols that develop the construction of the common reference strings from static to updatable and universal versions are mentioned.

Keywords: *succinct non-interactive arguments, public parameters, common reference strings, security.*

Введение

Описаны способы устранения уязвимостей протоколов zk-SNARK [1], основанных на защищённом формировании публичных параметров в виде главных ссылочных строк (Common Reference String, CRS) с использованием доверенной третьей стороны и многостороннего взаимодействия. Для протоколов zk-SNARK [2, 3], используемых в криптовалюте Zcash [4], рассматривается многостороннее формирование CRS [5] с единственной честной стороной. Для протокола zk-SNARK [6] на примере [7] рассмотрено более защищённое формирование CRS с неограниченным набором сторон в расширяемом онлайн-режиме, верификацией результатов и единственной честной стороной. Повышение уровня защищённости протокола zk-SNARK [6] также представлено переработанной версией [8], в которой выполнена корректировка уравнений верификации и значений, включаемых в CRS. Согласно источникам [9, 10], указывается ошибка описания протокола zk-SNARK [3], основанного на протоколе zk-SNARK «Pinocchio» [2]. Уязвимость [9, 10] опирается на избыточные элементы, включаемые в CRS и ключ формирования доказательств, которые позволяют строить доказательства для произвольных открытых входов. Приводится метод [9] устранения указанной уязвимости.

В качестве упоминания приводится набор протоколов [11–13], развивающий формирование CRS [7] от статической до обновляемой и универсальной версий, в том числе для протокола zk-SNARK [6] и криптовалюты Zcash [4]. Согласно источникам [14, 15], кратко отмечена возможность повышения уровня защищённости протокола zk-SNARK [2] для случая сговора всех сторон формирования CRS при условии предварительной проверки доказательств.

1. Протокол надёжного формирования CRS с фиксированным набором сторон

В [14] представлен метод формирования CRS для случая сговора всех задействованных сторон, при котором нарушители способны реконструировать «лазейку» τ и подделывать доказательства π. В работе [5] устраняются проблемы защищённости [14] и приведён многосторонний протокол формирования CRS для протоколов zk-SNARK [2, 3], используемых в криптовалюте Zcash [4]. Если хотя бы одна сторона является честной, то конструирование мошеннических доказательств становится невозможным. Дополнительно обеспечивается свойство нулевого разглашения (Zero-Knowledge, ZK), даже если все стороны являются нарушителями.

В протоколе формирования CRS [5] принимают участие n сторон, координатор и верификатор. Роль последних может выполняться одним сервером. Проверка корректности результатов протокола происходит после его полного выполнения, что может оказаться недостатком и потребует перевыполнения всего тяжеловесного процесса формирования CRS. Протокол состоит из четырёх раундов, где каждая сторона P_i, $i = 1, \ldots, n$, может отправить своё сообщение после получения сообщения от P_{i-1}, а P_1 начинает работу после приёма сообщения инициализации. В пределах каждого раунда стороны могут отправлять сообщения для координатора параллельно. Таким образом, протокол формирования CRS [5] имеет вид:

Раунд № 1. Для $i = 1, \ldots, n$ стороны P_i выполняют следующие шаги:

1. Выводятся наборы случайных элементов из $\mathbb{F}_r^*$ (для наглядности здесь и далее i-индексы вида $\rho_{B,i}$ не приводятся, а подразумеваются по умолчанию):

$$\begin{gathered}\mathsf{secrets}_i = \{\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \beta, \gamma\},\\ \mathsf{elements}_i = \{\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \beta, \gamma, \rho_A\alpha_A, \rho_B\alpha_B, \rho_A\rho_B, \rho_A\rho_B\alpha_C, \beta\gamma\}_i.\end{gathered} \tag{1}$$

2. На основе (1) вычисляется набор элементов, для чего P_i выбирает случайные $f_1, f_2, f_3 \in \mathbb{G}_2 \setminus \{0\}$, $f_4, f_5, f_6, f_7, f_8 \in \mathbb{G}_1 \setminus \{0\}$, а затем сохраняет наборы элементов $\mathbb{G}_2$ и $\mathbb{G}_1$ соответственно:

$$\begin{gathered}\mathsf{e}_i^{(2)} = \{f_1, f_1\rho_A, f_1\rho_A\alpha_A, f_1\rho_A\rho_B\alpha_C, f_1\rho_A\rho_B, f_1\rho_A\rho_B\alpha_B, f_2, f_2\beta, f_2\beta\gamma, f_3, f_3\tau\},\\ \mathsf{e}_i^{(1)} = \{f_4, f_4\alpha_A, f_5, f_5\alpha_C, f_6, f_6\rho_B, f_7, f_7\rho_A, f_8, f_8\gamma\}.\end{gathered}$$

3. Вычисляется результирующий вектор $\mathsf{e}_i = \left(\mathsf{e}_i^{(1)} \parallel \mathsf{e}_i^{(2)}\right)$.
4. Стороны P_i широковещательно передают $h_i = \mathsf{COMMIT}(\mathsf{e}_i)$, соответствующее вычислению хеш-функции (в работе [5] используется BLAKE-2).

Относительно e_i требуется, чтобы для каждого $s \in \mathsf{secrets}_i$ (1) при $s \in \mathbb{F}_r^*$ всегда присутствовала необходимая s-пара (p, q), где $s \cdot p = q$. Для конкретной группы используется термин $\mathbb{G}$-s-пара. Например, по $f_1, f_1\rho_A, f_1\rho_A\rho_B$ строятся ρ_A-пара $(f_1, f_1\rho_A)$, ρ_B-пара $(f_1\rho_A, f_1\rho_A\rho_B)$ и $\rho_A\rho_B$-пара $(f_1, f_1\rho_A\rho_B)$.

Раунд № 2.

Ч а с т ь 1 . Выполняется раскрытие обязательств, при котором стороны P_i широковещательно передают e_i, а верификатор проверяет соответствие $h_i = \mathsf{COMMIT}(\mathsf{e}_i)$. Далее $\mathsf{elements}_i$ (1) принимают наименование зафиксированных элементов, так как к текущему моменту выполнения протокола для каждого $s \in \mathsf{elements}_i$ стороны отправили s-пары для $\mathbb{G}_1$ и $\mathbb{G}_2$ в виде rp_s^1 и rp_s^2. Для представленных ниже значений переданные сторонами P_i элементы и s-пары имеют следующий вид:

$$\begin{aligned}
\tau &: (\mathsf{rp}_\tau^1, \mathsf{rp}_\tau^2) = (g, \tau g), & \gamma &: (\mathsf{rp}_\gamma^1, \mathsf{rp}_\gamma^2) = (g, \gamma g),\\
\rho_A &: (\mathsf{rp}_{\rho_A}^1, \mathsf{rp}_{\rho_A}^2) = (g, \rho_A g), & \rho_A\alpha_A &: (\mathsf{rp}_{\rho_A\alpha_A}^1, \mathsf{rp}_{\rho_A\alpha_A}^2) = (g, \rho_A\alpha_A g),\\
\rho_B &: (\mathsf{rp}_{\rho_B}^1, \mathsf{rp}_{\rho_B}^2) = (g, \rho_B g), & \rho_B\alpha_B &: (\mathsf{rp}_{\rho_B\alpha_B}^1, \mathsf{rp}_{\rho_B\alpha_B}^2) = (\rho_A g, \rho_A\rho_B\alpha_B g),\\
\alpha_A &: (\mathsf{rp}_{\alpha_A}^1, \mathsf{rp}_{\alpha_A}^2) = (\rho_A g, \rho_A\alpha_A g), & \rho_A\rho_B &: (\mathsf{rp}_{\rho_A\rho_B}^1, \mathsf{rp}_{\rho_A\rho_B}^2) = (g, \rho_A\rho_B g),\\
\alpha_B &: (\mathsf{rp}_{\alpha_B}^1, \mathsf{rp}_{\alpha_B}^2) = (\rho_A\rho_B g, \rho_A\rho_B\alpha_B g), & \rho_A\rho_B\alpha_C &: (\mathsf{rp}_{\rho_A\rho_B\alpha_C}^1, \mathsf{rp}_{\rho_A\rho_B\alpha_C}^2) = (g, \rho_A\rho_B\alpha_C g),\\
\alpha_C &: (\mathsf{rp}_{\alpha_C}^1, \mathsf{rp}_{\alpha_C}^2) = (\rho_A\rho_B g, \rho_A\rho_B\alpha_C g), & \beta\gamma &: (\mathsf{rp}_{\beta\gamma}^1, \mathsf{rp}_{\beta\gamma}^2) = (g, \beta\gamma g).\\
\beta &: (\mathsf{rp}_\beta^1, \mathsf{rp}_\beta^2) = (\gamma g, \beta\gamma g),
\end{aligned} \tag{2}$$

Ч а с т ь 2 . Проверяется факт фиксации сторонами P_i пар элементов из $\mathbb{G}_1$ и $\mathbb{G}_2$, входящих в rp_s^1 и rp_s^2, соответствующих s-парам одинаковых элементов $s \in \mathbb{F}_r^*$. Для $i \in \{1, \ldots, n\}$, $s \in \mathsf{elements}_i$ верификатор запускает проверку

$$\mathsf{SameRatio}(\mathsf{rp}_s^1, \mathsf{rp}_s^2) = \mathsf{SameRatio}((p, q), (f, H)). \tag{3}$$

Проверка (3) успешна, если при $p, q \in \mathbb{G}_1$ и $f, H \in \mathbb{G}_2$ все четыре элемента не равны 0 и $e(p, H) = e(q, f)$ для билинейного спаривания $e(ag_1, bg_2) = g_T^{ab}$.

Ч а с т ь 3 . Выполняется доказательство и проверка знаний дискретных логарифмов. Сторона P_1 вычисляет дайджест сообщений раунда № 1 $h = \mathsf{COMMIT}(h_1 \parallel \ldots \parallel \parallel h_n)$ и широковещательно его передаёт. Затем для $i = 1, \ldots, n$ выполняются следующие шаги:

1. Для $s \in \mathsf{secrets}_i$ (1) фиксируется $h_{i,s} = (h \parallel \mathsf{rp}_s^1)$. Все стороны P_i и верификатор имеют элементы для вычисления текущего шага.
2. Для каждого $s \in \mathsf{secrets}_i$ сторонами P_i широковещательно передаются доказательства $\pi_{i,s}$:
$$\pi_{i,s} = \mathsf{NIZK}(\mathsf{rp}_s^1, h_{i,s}). \tag{4}$$
Функция NIZK (4) основана на протоколе Шнорра [16, 17], которая по s-паре $\mathsf{rp}_s = (f, H = sf)$ и строке h выводит рандомизированную строку — доказательство знания строки s. Для функции (4) выбирается случайное a и $R = af$. Вычисляется $c = \mathsf{COMMIT}(R \parallel h)$, которое интерпретируется как элемент $\mathbb{F}_r$, например взятием первых $\log r$ бит, а также $u = a + cs$. Результатом является пара (R, u).
3. Для каждого $s \in \mathsf{secrets}_i$ верификатором проводится проверка
$$1/0 = \mathsf{VERIFY} - \mathsf{NIZK}(\mathsf{rp}_s^1, \pi_{i,s}, h_{i,s}). \tag{5}$$
Функция (5) проверяет, действительно ли доказательство соответствует заданному h. Для этого вычисляется $c = \mathsf{COMMIT}(R \parallel h)$ и доказательство принимается, если $uf = R + cH$.

Ч а с т ь 4 . Для $\tau = \tau_1, \ldots, \tau_n$ (2) вычисляется вектор
$$\mathsf{POWERS}_r = ((1, \tau, \tau^2, \ldots, \tau^d)g_1, (1, \tau, \tau^2, \ldots, \tau^d)g_2).$$

1. Стороной P_1 выполняется широковещательная передача $V_1 = (1, \tau, \tau^2, \ldots, \tau^d)g_1$ и $V_1' = (1, \tau, \tau^2, \ldots, \tau^d)g_2$, а для $i = 2, \ldots, n$ — широковещательная передача сторонами P_i значений $V_i = \mathsf{powerMult}(V_{i-1}, \tau_i)$ и $V_i' = \mathsf{powerMult}(V_{i-1}', \tau_{i-1})$. Например, для вектора $V \in \mathbb{G}^{d+1}$ и $a \in \mathbb{F}_r$ функция $\mathsf{powerMult}(V, a)_{i \in \{0,\ldots,d\}} = \{a^i V : i = 0, \ldots, d\} \in \mathbb{G}^{d+1}$.
2. Для $i \in \{2, \ldots, n\}$ верификатор подтверждает корректность ранее переданных векторов. Выполняются проверки
$$\begin{gathered}\mathsf{SameRatioSeq}(V_1, \mathsf{rp}_{\tau_1}^{(2)}), \quad \mathsf{SameRatioSeq}(V_1', (V_{1,0}, V_{1,1})), \\ \mathsf{SameRatioSeq}(V_i, (V_{i,0}', V_{i,1}')), \quad \mathsf{SameRatioSeq}(V_i', (V_{i,0}, V_{i,1})), \\ \mathsf{SameRatio}((V_{i-1,1}, V_{i,1}), \mathsf{rp}_{\tau_i}^{(2)}).\end{gathered} \tag{6}$$
Если проверки (6) выполняются, то на выход подаются компоненты ключа доказательства ($PK_H = V_n, PK_H' = V_n'$). Функция подтверждения соответствия двух пар элементов s-парам SameRatio() описана в (3). Функция SameRatioSeq() для $V \in \mathbb{G}_1^d$ и $\mathsf{rp}_s \in \mathbb{G}_2^2$ проверяет, что каждые два последовательных элемента V являются s-парой. Она соответствует вызову $\mathsf{SameRatio}(V', \mathsf{rp}_s)$ для $V' = ((V_0, V_1), (V_1, V_2), \ldots, (V_{d-1}, V_d))$.
3. Выполняется проверка корней $Z(x)$. Для обеспечения нулевого разглашения требуется, чтобы τ являлось корнем $Z(x) = x^d - 1$. Для этого верификатор и все P_i проверяют равенство $Z(\tau)g_1 = (\tau^d - 1)g_1 = V_{n,d} - V_{n,0} \neq 1$, иначе протокол перезапускается.
4. Используется набор векторов квадратичной арифметической программы (Quadratic Arithmetic Program, QAP). Координатор вычисляет QAP-векторы доказательства $\mathbf{A}, \mathbf{B}, \mathbf{B_2}, \mathbf{C}$ в точке τ, где, в отличие от [3], $A_{m+1} = B_{m+1} = C_{m+1} = Z[\tau]g_1 = (\tau^d - 1)g_1$:
$$\begin{gathered}\mathbf{A} = g_1\{A_i(\tau) : i = 0, \ldots, m+1\}, \quad \mathbf{B} = g_1\{B_i(\tau) : i = 0, \ldots, m+1\}, \\ \mathbf{B}_2 = g_2\{B_i(\tau) : i = 0, \ldots, m+1\}, \quad \mathbf{C} = g_1\{C_i(\tau) : i = 0, \ldots, m+1\}.\end{gathered} \tag{7}$$

Раунд № 3.

Часть 1. Координатор широковещательно передает векторы $\mathbf{A}, \mathbf{B}, \mathbf{B_2}, \mathbf{C}$ (7).

Часть 2. Для получения различных элементов ключей выполняется подпротокол выработки случайных коэффициентов RCPC. Для элементов $\alpha = (\alpha_1, \ldots, \alpha_n) \in$ $\in \mathsf{elements}_i$ (1) вычисляются элементы ключей

$$\begin{gathered} PK_A = \mathsf{RCPC}(\mathbf{A}, \rho_A), \quad PK_B = \mathsf{RCPC}(\mathbf{B}_2, \rho_B), \quad PK_C = \mathsf{RCPC}(\mathbf{C}, \rho_A\rho_B), \\ PK'_A = \mathsf{RCPC}(\mathbf{A}, \rho_A\alpha_A), \quad PK'_B = \mathsf{RCPC}(\mathbf{B}, \rho_B\alpha_B), \quad PK'_C = \mathsf{RCPC}(\mathbf{C}, \rho_A\rho_B\alpha_C), \\ temp_B = \mathsf{RCPC}(\mathbf{B}, \rho_B), \quad VK_Z = \mathsf{RCPC}(g_2 Z(\tau) = g_2(\tau^d - 1), \rho_A\rho_B), \\ VK_A = \mathsf{RCPC}(g_2, \alpha_A), \quad VK_B = \mathsf{RCPC}(g_1, \alpha_B), \quad VK_C = \mathsf{RCPC}(g_2, \alpha_C). \end{gathered} \tag{8}$$

Для расчётов (8) не требуется s-пара для каждого $s \in \mathsf{secrets}_i$ каждой группы. Например, требуется только $\mathbb{G}_2$-ρ_A-пара, которая используется для вычисления PK_A, а $\mathbb{G}_1$-ρ_A-пара не используется.

Подпротокол RCPC (8) покоординатно умножает вектор элементов из $\mathbb{G}_1$ на скаляр $\alpha \in \mathbb{F}_r^*$. Перед вызовом подпротокола для каждого $i \in \{1, \ldots, n\}$ стороны P_i передали $\mathbb{G}_2$-α_i-пары rp_{α_i}, доступные верификатору. Таким образом, $\mathsf{RCPC}(V, \alpha)$ принимает $V \in \mathbb{G}_1^d$ и $\alpha_i \in \mathbb{F}_r^*$ для $i \in \{1, \ldots, n\}$. Сторона P_1 вычисляет $V_1 = \alpha_1 V$, а для $i = 2, \ldots, n$ стороны P_i широковещательно передают $V_i = \alpha_i V_{i-1}$. В итоге стороны выводят $V_n = \alpha V$.

Раунд № 4. Вычисляются компоненты ключей, содержащие элемент β, для чего стороны P_i или только координатор рассчитывают $V = PK_A + temp_B + PK_C$. Затем стороны вычисляют

$$\begin{gathered} PK_K = \mathsf{RCPC}(V, \beta), \quad VK_\gamma = \mathsf{RCPC}(g_2, \gamma), \\ VK^{(1)}_{\beta\gamma} = \mathsf{RCPC}(g_1, \beta\gamma), \quad VK^{(2)}_{\beta\gamma} = \mathsf{RCPC}(g_2, \beta\gamma). \end{gathered}$$

В заключение верификатор запускает функцию verifyRCPC() на входных параметрах в виде результатов работы раундов № 3, 4. Верификация успешна, если все вызовы подтверждены.

Для векторов $S, T \in \mathbb{G}_1^d$ и $\mathbb{G}_2$-α-пары rp_α функция $\mathsf{sameRatio}((S, T), \mathsf{rp}_\alpha)$ (3) возвращает $\mathsf{sameRatio}(V, \mathsf{rp}_\alpha)$, где $V_i = (S_i, T_i)$. Функция $\mathsf{verifyRCPC}(V, \alpha)$ принимает V, а также $V_1, \ldots, V_n \in \mathbb{G}_1^d$ и $\mathbb{G}_2$-α_i-пары rp_{α_i} для $i \in \{1, \ldots, n\}$. Затем запускается $\mathsf{sameRatio}((V, V_1), \mathsf{rp}_{\alpha_1})$, а для $i = 2, \ldots, n$ запускается $\mathsf{sameRatio}((V_{i-1}, V_i), \mathsf{rp}_{\alpha_i})$. Если все итерации успешны, верификация подтверждается.

В работе [15] отмечается, что злонамеренный выбор секретных параметров, зависящих от $\tau, \rho_A, \rho_B, \alpha_A, \alpha_C, \gamma, \beta \in \mathbb{F}_r^*$, может нарушить свойство нулевого разглашения протокола [2]. Даже при отсутствии честных сторон P_i формирование CRS способом [5] устраняет данную уязвимость протокола [2] при условии проверки доказательства перед его отправкой.

2. Протокол надёжного формирования CRS с неограниченным и расширяемым набором сторон

По сравнению с работой [5], в [7] рассмотрено более защищённое формирование CRS протокола zk-SNARK [6]. Алгоритмы формирования доказательства и верификации соответствуют протоколу zk-SNARK [6]. Протокол [7] работает с неограниченным набором сторон в расширяемом онлайн-режиме без их предварительного выбора, не требует хранения конфиденциальных значений в течение длительного времени и предварительной фиксации случайных значений. Доказательства будут достоверны, если

хотя бы одна из N сторон является честной. Дополнительную роль играет ненадёжный координатор — детерминированная функция, причём любая сторона способна проверить корректность его сообщений. В частности, верификатор формирования CRS [7] независимо вычисляет сообщения координатора и проверяет их достоверность. В роли верификатора может выступать как координатор, так и любая сторона протокола.

В [7] арифметическая схема $\mathbf{C}$ над $\mathbb{F}_p$ рассматривается как чередующиеся слои умножения/деления $\mathbf{C}_1, \ldots, \mathbf{C}_d$ и их линейные комбинации $\mathbf{L}_1, \ldots, \mathbf{L}_d$. Вход схемы $\mathbf{x}$ разделяется на непересекающиеся наборы $\mathbf{x}^1, \ldots, \mathbf{x}^d$, соответствующие слоям. Целью вычислений является вывод $\mathbf{C}(\mathbf{x})\mathbf{g}$ для произвольного входа $\mathbf{x}$. Детальнее, рассматривается произведение $\mathbf{x} = (\mathbf{x}_1 \cdot \ldots \cdot \mathbf{x}_N \cdot \mathbf{x}')$, где $\mathbf{x}_i \in (\mathbb{F}_p^*)^t$ — вход стороны P_i, а $\mathbf{x}'$ — выход функции рандомизации RB. Уровни схемы обозначаются как $\mathbf{C}_1, \mathbf{L}_1, \ldots, \mathbf{C}_d, \mathbf{L}_d$, и протокол выполняет d рабочих фаз. Для выполнения одной фазы фиксируется уровень $l \in \{1, \ldots, d\}$, $\mathbf{C} = \mathbf{C}_l$ и $\mathbf{L} = \mathbf{L}_l$. Предполагается, что для всех вентилей $\mathbf{g}$ предыдущих уровней $\mathbf{C}_1, \mathbf{L}_1, \ldots, \mathbf{C}_{l-1}, \mathbf{L}_{l-1}$ уже вычислен набор значений $[\mathbf{g}] \in \mathbf{G} = \mathbb{G}_1 \times \mathbb{G}_2$.

По сравнению с протоколом zk-SNARK [6] в CRS [7] добавлены элементы

$$\{x^i : i = n, \ldots, 2n-2\}, \quad \{\alpha x^i : i = 1, \ldots, n-1\}, \quad \{\beta x^i : i = 1, \ldots, n-1\} \tag{9}$$

и убраны элементы $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\gamma : i = 0, \ldots, l\}$, γ, которые являются линейной комбинацией компонентов CRS и добавленных элементов (9). В [7] протокол zk-SNARK [6] вычисляется с помощью схемы с двумя слоями. При этом слой $\mathbf{C}_1$ имеет вход $\mathbf{x}^1 = \{x, \alpha, \beta\}$ и вычисляет (9). Слой $\mathbf{L}_1$ вычисляет $\{x^i t(x) : i = 0, \ldots, n-2\}$ как линейные комбинации x^i, $i \in \{0, \ldots, 2n-2\}$, где $\deg(t(x)) = n$, а также $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x)) : i = 0, \ldots, m\}$ как линейные комбинации результатов $\mathbf{C}_1$. Слой $\mathbf{C}_2$ имеет вход $\mathbf{x}^2 = \{\delta\}$ и вычисляет δ, $\{(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta : i = l+1, \ldots, m\}$, $\{x^i t(x)/\delta : i = 0, \ldots, n-2\}$.

Для упрощения надёжное формирование CRS [7] для протокола zk-SNARK [6] представлено для одной группы и аналогично для других групп. Используются следующие значения: n — степень полиномов QAP [6]; $[x]$ — набор значений всех открытых входов; $\{1, \ldots, l\}$ — индексы открытых входов; $\{l+1, \ldots, m\}$ — индексы секретных входов. Значение M является вычисляемым выходом в $\mathbb{G}_1$, $\mathbb{G}_2$ или $\mathbf{G} = \mathbb{G}_1 \times \mathbb{G}_2$; P_j — стороны протокола при $j \in \{1, \ldots, N\}$; $[M]^j$ — значение M после внесения сторонами $P_1, \ldots, P_j$ долей распределённых вычислений; $[M]^0$ — начальное значение; $\mathbf{g} = (g_1, g_2)$ — порождающие элементы групп $\mathbb{G}_1, \mathbb{G}_2$. Выполняются два раунда протокола [7].

Раунд № 1. Используются следующие вспомогательные функции. Формирование доказательства знания $\alpha \in \mathbb{F}_p^*$ выполняется функцией $\mathsf{POK}(\alpha, \mathbf{v})$, которая строит $r = R([\alpha]_1, \mathbf{v}) \in \mathbb{G}_2^*$ и на выход подает $([\alpha]_1, \alpha r)$, где R — хеш-функция; $[\alpha]_1 = \alpha g_1$ при $\langle g_1 \rangle = \mathbb{G}_1$. Верификация доказательства знания $\alpha \in \mathbb{F}_p^*$ выполняется за счёт функции $\mathsf{CheckPOK}(a, \mathbf{v}, b)$ при $a \in \mathbb{G}_1^*$, $b \in \mathbb{G}_2^*$, которая строит $r = R(a, \mathbf{v}) \in \mathbb{G}_2^*$ и на выход подаёт $\mathsf{SameRatio}((g_1, a), (r, b))$. Функция $\mathsf{SameRatio}$ (3) соответствует [5].

Функция $\mathsf{Consistent}(A, B, C)$ [7] использует $(A, B) \in \mathbb{G}_1^2$ или $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2 = \mathbf{G}^2$, а также $C \in \mathbb{G}_2^*$ или $C \in (\mathbb{G}_2^*)^2$. Если $C \in (\mathbb{G}_2^*)^2$, то вычисляется $r = \mathsf{SameRatio}((A_1, B_1), (C_1, C_2))$, иначе $r = \mathsf{SameRatio}((A_1, B_1), (g_2, C))$. Если $(A, B) \in \mathbb{G}_1^2$, то на выход подаётся r, иначе выводится r и $\mathsf{SameRatio}((A_1, B_1), (A_2, B_2))$. Верификатор протокола для $j \in \{1, \ldots, N\}$ вычисляет $r_{\alpha,j} = R([\alpha_j]_1, \mathsf{transcript}_{1,j-1})$, $r_{\beta,j} = R([\beta_j]_1, \mathsf{transcript}_{1,j-1})$, $r_{x,j} = R([x_j]_1, \mathsf{transcript}_{1,j-1})$. Функция $\mathsf{transcript}_{1,j-1}$ выводит сообщения протокола до момента отправки стороной P_j нового сообщения.

Функция RB принимает временной слот J, целое число k и возвращает случайное $a \in \mathbb{F}_p^*$. Пусть $(J-1)$ — временной интервал, в который P_N отправляет сообщение.

Выполняются следующие шаги:

1. Для каждого входа x схемы $\mathbf{C}$ выполняется инициализация значений

$$\{[x^i]^0 = \mathbf{g} : i = 1, \dots, n-1\}, \quad \{[x^i]^0 = g_1 : i = n, \dots, 2n-2\},$$
$$\{[\alpha x^i]^0 = g_1 : i = 0, \dots, n-1\}, \quad [\beta]^0 = \mathbf{g}, \quad \{[\beta x^i]^0 = g_1 : i = 1, \dots, n-1\}.$$

2. Для $j \in \{1, \dots, N\}$ стороны протокола P_j вычисляют набор значений

$$[\alpha_j]_1, \quad [\beta_j]_1, \quad [x_j]_1,$$
$$y_{\alpha,j} = \mathsf{POK}(\alpha_j, \mathsf{transcript}_{1,j-1}), \quad y_{\beta,j} = \mathsf{POK}(\beta_j, \mathsf{transcript}_{1,j-1}),$$
$$y_{x,j} = \mathsf{POK}(x_j, \mathsf{transcript}_{1,j-1}),$$
$$\{[x^i]^j = x_j^i [x^i]^{j-1} : i = 1, \dots, 2n-2\}, \quad \{[\alpha x^i]^j = \alpha_j x_j^i [\alpha x^i]^{j-1} : i = 0, \dots, n-1\},$$
$$\{[\beta x^i]^j = \beta_j x_j^i [\beta x^i]^{j-1} : i = 0, \dots, n-1\}.$$

3. Координатор выводит $(x', \alpha', \beta') = \mathsf{RB}(J, 3)$, определяются значения

$$\{[x^i] = x'^i [x^i]^N : i = 1, \dots, 2n-2\}, \quad \{[\alpha x^i] = \alpha' x'^i [\alpha x^i]^N : i = 0, \dots, n-1\},$$
$$\{[\beta x^i] = \beta' x'^i [\beta x^i]^N : i = 0, \dots, n-1\}.$$

4. Достоверность полученных значений выполняется проверками

$$\mathsf{CheckPOK}([\alpha_j]_1, \mathsf{transcript}_{1,j-1}, y_{\alpha,j}), \quad \mathsf{CheckPOK}([\beta_j]_1, \mathsf{transcript}_{1,j-1}, y_{\beta,j}),$$
$$\mathsf{CheckPOK}([x_j]_1, \mathsf{transcript}_{1,j-1}, y_{x,j}),$$
$$\mathsf{consistent}([\alpha]^{j-1} - [\alpha]^j, (r_{\alpha,j}, y_{\alpha,j})), \quad \mathsf{consistent}([\beta]^{j-1} - [\beta]^j, (r_{\beta,j}, y_{\beta,j})),$$
$$\mathsf{consistent}([x]^{j-1} - [x]^j, (r_{x,j}, y_{x,j})),$$
$$\{\mathsf{consistent}([x^{i-1}]^j - [x^i]^j, [x]^j) : i = 1, \dots, 2n-2\},$$
$$\{\mathsf{consistent}([x^i]_1^j - [\alpha x^i]^j, [\alpha]^j) : i = 1, \dots, n-1\},$$
$$\{\mathsf{consistent}([x^i]_1^j - [\beta x^i]^j, [\beta]^j) : i = 1, \dots, n-1\}.$$

5. На выход раунда № 1 подаётся набор значений

$$M_1 = \big\{\{[x^i] : i = 0, \dots, n-1\}, \ \{[x^i]_1 : i = n, \dots, 2n-2\},$$
$$\{[\alpha x^i]_1 : i = 0, \dots, n-1\}, \ [\beta], \ [\delta], \ \{[\beta x^i]_1 : i = 1, \dots, n-1\}\big\}. \tag{10}$$

Раунд № 2. Выполняются следующие шаги:

1. Для каждого входа x схемы $\mathbf{C}$ вычисляются

$$K_i = \{\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta : i = l+1, \dots, m\},$$
$$H_i = \{t(x)x^i/\delta : i = 0, \dots, n-2\}.$$

2. Выполняется инициализация необходимых для раунда № 2 значений:

$$\{[K_i]^0 = K_i' : i = l+1, \dots, m\}, \quad \{[H_i]^0 = H_i' : i = l+1, \dots, m\}, \quad [\delta]^0 = \mathbf{g}.$$

3. Для $j \in \{1, \ldots, N\}$ стороны протокола P_j вычисляют набор значений

$$[\delta_j]_1, \quad y_{\delta,j} = \mathsf{POK}(\delta_j, \mathsf{transcript}_{2,j-1}), \quad [\delta]^j = [\delta]^{j-1}/\delta_j,$$
$$\{[K_i]^j = ([K_i]^{j-1})/\delta_j : i = l+1, \ldots, m\}, \quad \{[H_i]^j = ([H_i]^{j-1})/\delta_j : i = 0, \ldots, n-2\}.$$

4. Аналогично раунду № 1, пусть $(J-1)$ — временной интервал, в который P_N отправляет сообщение. Выводится $\delta' = \mathsf{RB}(J, 1)$ и определяются значения

$$[\delta] = [\delta]^N/\delta', \quad [K_i]_1 = [K_i]^N/\delta', \quad [H_i]_1 = [H_i]^N/\delta'.$$

5. Верификатор для $j \in \{1, \ldots, N\}$ вычисляет $r_{\delta,j} = R([\delta_j]_1, \mathsf{transcript}_{2,j-1})$. Достоверность полученных значений выполняется проверками

$$\mathsf{CheckPOK}([\delta_j]_1, \mathsf{transcript}_{2,j-1}, y_{\delta,j}), \quad \mathsf{consistent}([\delta]^{j-1} - [\delta]^j, (r_{\delta,j}, y_{\delta,j})),$$
$$\{\mathsf{consistent}([K_i]^j - [K_i]^{j-1}, [\delta_j]) : i = l+1, \ldots, m\},$$
$$\{\mathsf{consistent}([H_i]^j - [H_i]^{j-1}, [\delta_j]) : i = 0, \ldots, n-2\}.$$

6. На выход раунда № 2 подаётся следующий набор значений:

$$M_2 = \big\{[\delta], \ \{[K_i]_1 : i = l+1, \ldots, m\}, \ \{[H_i]_1 : i = 0, \ldots, n-2\}\big\}. \tag{11}$$

Результирующая CRS [7] основана на M_1 (10) и M_2 (11) и принимает вид

$$\{[x^i] : i = 0, \ldots, n-1\}, \{[x^i]_1 : i = n, \ldots, 2n-2\}, \{[\alpha x^i]_1 : i = 0, \ldots, n-1\},$$
$$[\beta], \{[\beta x^i]_1 : i = 1, \ldots, n-1\}, \{[x^i t(x)/\delta]_1 : i = 0, \ldots, n-2\},$$
$$\{[(\beta u_i(x) + \alpha v_i(x) + w_i(x))/\delta]_1 : i = l+1, \ldots, m\}.$$

По сравнению с работой [7], в [11] представлено альтернативное доказательство защищённости информации, содержащейся в CRS протокола zk-SNARK [6]. Для этого используется 2-раундовая процедура формирования и проверки корректности публичных значений. Протокол [11] формирует надёжную CRS и сохраняет конфиденциальность секретного входа для случая с единственной честной стороной. Защищённость сохраняется даже при доступе злоумышленников ко всем этапам рерандомизации CRS, кроме одного. В результате протокол [11] формирует новые обновляемые публичные параметры CRS для Zcash [4] и проводит их верификацию, развивая идеи [7] на основе протокола zk-SNARK [6]. Свойство обновляемости CRS позволяет динамичному набору сторон вносить в публичные параметры секретную случайность неограниченное количество раз.

На каждом этапе протокола [7] требуется наличие хотя бы одной честной стороны. При этом второй этап [7] зависит от первого, поэтому формируемая CRS не может обновляться. В протоколе [12] любая сторона в любое время способна обновить и подтвердить корректность обновлённой CRS. Это верно, если одна из старых CRS достоверна и/или одна из сторон обновления является честной. Таким образом, CRS [12] может непрерывно рерандомизироваться с сохранением результатов всех предыдущих обновлений, удовлетворяя доверию всех сторон, заинтересованных в надёжности формируемых публичных значений. Кроме того, CRS [12] не привязывается к конкретной дискретной функции, поэтому является универсальной CRS (Universal CRS, UCRS) и подходит для различных дискретных функций. В определённом смысле протокол формирования CRS [12] всё ещё использует доверенную третью сторону, однако уверенность в защищённости CRS повышается, потому что только одна предыдущая сторона должна уничтожать введённую секретную случайность.

Например, согласно [12], CRS протокола zk-SNARK [18] может быть обновлена, в то время как CRS протокола zk-SNARK «Pinocchio» [2] не является обновляемой и требует вмешательства доверенной третьей стороны. Источник [12] также демонстрирует способ нарушения защищённости исторически базового протокола zk-SNARK «Pinocchio» [2].

Работа [13] продолжает развитие идей [12], где используются обновляемые универсальные структурированные CRS (Structured UCRS, SUCRS) и приводится метод, в котором ненадёжные стороны повышают производительность пакетной верификации доказательств. Для произвольных дискретных функций доказательство протокола [13] составляет 256 байт, все компоненты которого принадлежат одной группе.

3. Устранение уязвимости протокола zk-SNARK Э. Бен-Сассона, А. Кьезы, Э. Тромера, М. Вирзы

Источник [9] раскрывает ошибку в описании протокола zk-SNARK [3]. По сравнению с исходной версией протокола zk-SNARK «Pinocchio» [2], схема [3] включает в CRS избыточные элементы, которые важно не раскрывать. Уязвимость позволяет при наличии корректного доказательства для некоторого открытого входа создавать корректные доказательства для любого открытого входа. В [9] представлено также доказательство надёжности протокола zk-SNARK [3] при исключении из CRS данных элементов и удовлетворении QAP определённым алгебраическим условиям.

Источник [10] описывает отличия протоколов zk-SNARK [2] и [3]. Показана уязвимость [3], в которой нарушитель представляет ложные открытый вход и доказательство, принимаемые верификатором. Согласно [10], устранение уязвимостей [3] требует издержек и компромиссов производительности. На основе результатов [10] в исправленной версии работы [3] в виде [19] атака была ослаблена. Однако в [9] представлена более серьёзная уязвимость протокола zk-SNARK [3] на основе избыточных элементов ключа формирования доказательства. Протоколы zk-SNARK [5, 20, 21] и реализация [22] основаны на [3], поэтому унаследовали проблемы защищённости. Реализация [23] не использует избыточные элементы, поэтому не подвержена атаке [9].

Для описания устранения уязвимости [9] используется нотация протокола zk-SNARK [3], где m — размер QAP; d — степень QAP; n — размер открытого входа $x \in \mathbb{F}^n$; QAP имеет форму $\big\{\{A_i(X), B_i(X), C_i(X) : i = 0, \ldots, m\}, Z(X)\big\}$; степени $A_i, B_i, C_i \in \mathbb{F}[X]$ не выше d; $Z \in \mathbb{F}[X]$ и имеет степень d; $[x]_i = g_i^x$ при $\langle g_i \rangle = \mathbb{G}_i$.

Таким образом, в [3] элементы $\mathsf{pk}'_{\mathsf{A,i}} = [\alpha_A \rho_A A_i(\tau)]_1$ не используются доказывающим и верификатором, однако позволяют доказывающему заменять открытые входы на основе корректного доказательства. Это возможно за счёт введения множителя для компоненты доказательства π_A, что изменяет значение открытого входа, связанного с исходным доказательством, с $x = (x_1, \ldots, x_n) \in \mathbb{F}^N$ на $x' = (x'_1, \ldots, x'_n) \in \mathbb{F}^N$. Первое уравнение верификации [3] $e(\pi'_A, g_2) = e(\pi_A, [\alpha_A]_2)$ для компонент доказательства $\pi_A = [\rho_A A_{mid}(\tau)]_1, \pi'_A = [\alpha_A \rho_A A_{mid}(\tau)]_1$, где $A_{mid} = \sum_{i=0}^{m} x_i A_i - \sum_{i=0}^{n} x_i A_i$, призвано недопустить нежелательное поведение, но избыточные элементы также позволяют злонамеренному доказывающему добавить аналогичный множитель к π'_A. В результате для изменённого открытого входа верификация становится успешной. Подробности атаки приведены в [9].

Уязвимость протокола zk-SNARK [3] устраняется исключением из ключа доказательства набора $\mathsf{pk}'_{\mathsf{A,i}} = [\alpha_A \rho_A A_i(\tau)]_1$ и использованием QAP с линейно независимыми полиномами $\{A_i : i = 0, \ldots, n\}$ [10], которые не пересекаются в индексах $i \in \{0, \ldots, n\}$

и $i \in \{n+1, \ldots, m\}$. В результате изменения протокола zk-SNARK [3] с учётом модификаций [9] принимают следующий вид:

Алгоритм формирования ключей

1. Выводятся случайные секретные значения $\tau, \rho_A, \rho_B, \alpha_A, \alpha_B, \alpha_C, \gamma, \beta \in \mathbb{F}_p^*$.
2. Для $i \in \{0, \ldots, d\}$ вычисляются τ^i.
3. Для $i \in \{0, \ldots, m\}$ вычисляются компоненты ключа доказательства:

$$\rho_A A_i(\tau), \quad \rho_B B_i(\tau), \quad \alpha_B \rho_B B_i(\tau), \quad \rho_A \rho_B C_i(\tau),$$
$$\alpha_C \rho_A \rho_B C_i(\tau), \quad \beta(\rho_A A_i(\tau) + \rho_B B_i(\tau) + \rho_A \rho_B C_i(\tau)).$$

4. Формируется набор $(\alpha_A, \alpha_B, \alpha_C, \gamma, \beta\gamma, \rho_A \rho_B Z(\tau))$ и для $i \in \{n+1, \ldots, m\}$ вычисляются $\alpha_A \rho_A A_i(\tau)$.

Алгоритм верификации

Вычисляется $\mathsf{PI}(x) = \rho_A A_0(\tau) + \sum\limits_{i=1}^{n} x_i \rho_A A_i(\tau)$ и проверяется выполнение следующих уравнений:

$$\pi'_A = \alpha_A \pi_A, \quad \pi'_B = \alpha_B \pi_B, \quad \pi'_C = \alpha_C \pi_C,$$
$$\gamma \pi_K = \beta\gamma(\mathsf{PI}(x) + \pi_A + \pi_B + \pi_C), \quad (\mathsf{PI}(x) + \pi_A)\pi_B = \pi_C + \pi_H Z(\tau) \rho_A \rho_B.$$

4. Устранение уязвимости протокола zk-SNARK Й. Грота

Протокол zk-SNARK [8] основан на программах квадратичной арифметики (Square Arithmetic Program, SAP) и билинейном спаривании. В [8] приняты меры по устранению проблем защищённости протокола zk-SNARK [6, 1], в котором уравнение верификации для известного полинома $f(\phi)$ и секретных α, β, δ имеет вид $e(A, B) = e(g^{\alpha}, h^{\beta}) e(g^{f(\phi)}, h) e(C, h^{\delta})$. В данном случае противник может изменить текущее доказательство в другое доказательство того же состояния ϕ. При этом он имеет два способа рандомизации компонент доказательства (A, B, C).

В первом случае противник для некоторого r переназначает компоненты доказательства в виде $A' = A^r$, $B' = B^{1/r}$, $C' = C$. Атака устраняется добавлением проверки $e(A, h) = e(g, B)$, а для значения $r = -1$ уравнение верификации корректируется для использования $e(Ag^{\alpha\delta}, Bh^{\beta\delta})$ вместо $e(A, B)$. Во втором случае противник выполняет переназначение вида $A' = A$, $B' = Bh^{r\delta}$, $C' = A^r C$. Для устранения атаки в CRS включаются h^{δ}, $g^{\gamma\delta}$ и $h^{\gamma\delta}$, а g^{δ} исключается. Но если противник устанавливает $B' = Bh^{r\delta}$, то единственным возможным значением, удовлетворяющим представленному ниже уравнению верификации, будет $A' = Ag^{r\delta}$ вместо доступного $A' = A$, которое противник вычислить не способен ввиду отсутствия g^{δ} в CRS. Значение $r = \Phi(\tau)$ соответствует значению полинома Φ в секретной «лазейке» τ. Результирующие уравнения верификации представлены ниже в алгоритме верификации доказательств.

В работе [8] используется частный случай QAP под названием SAP, где $u_i(x) = v_i(x)$ для всех i. Формальное определение SAP имеет следующий вид:

$$R = \big(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, l, \{u_i(x), w_i(x) : i = 0, \ldots, m\}, t(x)\big).$$

Билинейная группа определяет конечное поле $\mathbb{Z}_p$ для простого $p > 2^{\lambda-1}$, $1 \leqslant l \leqslant m$, $u_i(x), w_i(x), t(x) \in \mathbb{Z}_p[x]$ и $u_i(x), w_i(x)$ имеют строго более низкую степень, чем $n = \deg(t(x))$. Множество $S = \{u_i(x) : i = 0, \ldots, l\}$ линейно независимо, всякий полином

$u_i(x) \in S$ линейно независим от множества $\{u_j(x) : j = l+1, \ldots, m\}$. В результате при $s_0 = 1$ программа SAP определяет следующее бинарное отношение:

$$R = \Big\{(\phi, w) : \phi = (s_1, \ldots, s_l) \in \mathbb{Z}_p^l, w = (s_{l+1}, \ldots, s_m) \in \mathbb{Z}_p^{m-l}, \\ \exists h(x) \in \mathbb{Z}_p[x] \Big(\deg(h) \leqslant n-2 \;\&\; \big(\sum_{i=0}^{m} s_i u_i(x)\big)^2 = \sum_{i=0}^{m} s_i w_i(x) + h(x)t(x)\Big)\Big\}. \tag{12}$$

Алгоритм формирования ключей $(\mathsf{crs}_{\mathsf{sap}}, \rho) \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{sap})$

На основе отношения R (12) и параметра защиты λ вырабатываются CRS и ρ — ключ проверки корректности CRS. Алгоритм Setup выполняет следующие шаги:

1. Выводятся случайные элементы для формирования ключа верификации $\tau = (\alpha, \beta, \gamma, \delta, x) \in \mathbb{Z}_p^5$, такие, что $t(x) \neq 0$.
2. Вычисляется CRS

$$\mathsf{crs}_{\mathsf{sap}} = \big(g^\alpha, g^\gamma, g^x, g^{\alpha\delta}, g^{\gamma\delta t(x)}, g^{\gamma^2\delta t(x)^2}, g^{(\alpha+\beta)\gamma\delta t(x)}, h, h^\beta, h^\delta, h^{\beta\delta}, h^{\gamma\delta t(x)}, h^{\delta^2}, \\ \{g^{\gamma\delta x^i}, h^{\gamma\delta x^i}, g^{\gamma^2\delta t(x) x^i} : i = 0, \ldots, n-1\}, \{g^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x)} : i = 0, \ldots, l\}, \\ \{g^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x)} : i = l+1, \ldots, m\}\big). \tag{13}$$

3. Вычисляется ключ ρ без раскрытия значения g^δ, предназначенный для проверки корректности CRS (13):

$$\rho = (g^\alpha, h^\beta, g^\gamma, h^\delta, g^x).$$

4. На выход подаётся тройка $(\mathsf{crs}, \tau, \rho)$.

Строка CRS (13) содержит $m + 2n + 5$ элементов из $\mathbb{G}_1$ и $n + 3$ элементов из $\mathbb{G}_2$. Критически важно, чтобы CRS не содержала значений вида g, g^b, в которых b является небольшой величиной, например битом, за счёт чего нарушитель может восстановить данную переменную и решить задачу дискретного логарифмирования.

Алгоритм проверки CRS-структуры $0/1 \leftarrow \mathsf{UpdateVerify}(1^\lambda, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \rho)$

1. Выполняется проверка набора уравнений

$$\begin{aligned}
&e(g^{\alpha\delta}, h) = e(g^\alpha, h^\delta), && e(g^{\gamma\delta}, h^\beta) = e(g^\gamma, h^{\beta\delta}),\\
&e(g^{\gamma\delta}, h) = e(g^\gamma, h^\delta), && e(g^{\gamma\delta}, h^\delta) = e(g^\gamma, h^{\delta^2}),\\
&e(g^{\gamma\delta t(x)}, h^{\gamma\delta}) = && e(g^{\gamma\delta t(x)}, h^{\gamma\delta}) = e(g^{\gamma\delta}, h^{\gamma\delta t(x)}),\\
&\quad = e(g^{\gamma\delta(t(x)-t_0)\gamma\delta/x}, h^{\gamma\delta x}) e(g^{-t_0\gamma\delta}, h^{\gamma\delta}), &&\\
&e(g^{\gamma^2\delta t(x)^2}, h^\delta) = e(g^{\gamma\delta t(x)}, h^{\gamma\delta t(x)}), && e(g^{(\alpha+\beta)\gamma\delta t(x)}, h^\delta) =\\
&e(g^\gamma, h^\delta) = e(g, h^{\gamma\delta}), && \quad = e(g^{\alpha\delta}, h^{\gamma\delta t(x)}) e(g^{\gamma\delta t(x)}, h^{\beta\delta}).
\end{aligned} \tag{14}$$

2. Выполняется проверка набора уравнений для различных индексов i:

$$\begin{aligned}
0 \leqslant i \leqslant n-2 \;&:\; e(g^x, h^{\gamma x^i}) = e(g, h^{\gamma x^{i+1}}),\quad e(g^{\gamma\delta x^i}, h^{\gamma\delta x}) = e(g^{\gamma\delta x^{i+1}}, h^{\gamma\delta}),\\
1 \leqslant i \leqslant n-1 \;&:\; e(g^{\gamma^2\delta t(x) x^i}, h^{\gamma\delta}) = e(g^{\gamma\delta t(x)}, h^{\gamma\delta x^i}),\\
0 \leqslant i \leqslant l \;&:\; e(g^{\gamma\delta w_i(x) + \delta(\alpha+\beta)u_i(x)}, h^{\gamma\delta}) =\\
&\quad = e(g^{\gamma\delta}, h^{\gamma\delta w_i(x)}) e(g^{\alpha\delta}, h^{\gamma\delta u_i(x)}) e(g^{\gamma\delta u_i(x)}, h^{\beta\delta}),\\
l+1 \leqslant i \leqslant m \;&:\; e(g^{\gamma^2\delta w_i(x) + \gamma\delta(\alpha+\beta)u_i(x)}, h^\delta) =\\
&\quad = e(g^{\gamma\delta}, h^{\gamma\delta w_i(x)}) e(g^{\alpha\delta}, h^{\gamma\delta u_i(x)}) e(g^{\gamma\delta u_i(x)}, h^{\beta\delta}).
\end{aligned} \tag{15}$$

3. Если все проверки (14) и (15) выполняются, то на выход подаётся 1, иначе 0.

Алгоритм доказывающего $\pi \leftarrow \mathsf{Prove}(\mathsf{info}, \phi, w)$

Алгоритм доказывающего анализирует публичные ϕ и приватные w значения входов-выходов логических элементов схемы $(1, a_1, \ldots, a_m)$ и встраивает оценки SAP-полиномов $a_i u_i(x)$, полученные в случайной секретной точке x, в один элемент проверки в $\mathbb{G}_1$ и в один элемент проверки в $\mathbb{G}_2$. На их основе строится третий элемент доказательства, показателем которого является произведение показателей первых двух элементов; $\mathsf{info} = \left(\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, e(g^{\alpha\delta}, h^{\beta\delta})\right)$. Алгоритм выполняет следующие шаги:

1. Выбирается случайный элемент $r \in \mathbb{Z}_p$ для обеспечения свойства ZK.
2. Выполняется выборка публичных и приватных значений входов-выходов логических элементов схемы $(\phi \parallel w)$: $(a_0, a_1, \ldots, a_m) \leftarrow \mathsf{Parse}(1, \phi, w)$.
3. Вычисляется многочлен-делитель

$$h(x) = \left(\left(\sum_{i=0}^{m} a_i u_i(x)\right)^2 - \sum_{i=0}^{m} a_i w_i(x)\right) / t(x).$$

4. Вычисляются компоненты доказательства $\pi = (A, B, C)$:

$$\begin{gathered} A = g^{\gamma\delta\left(rt(x) + \sum\limits_{i=0}^{m} a_i u_i(x)\right)}, \\ B = h^{\gamma\delta\left(rt(x) + \sum\limits_{i=0}^{m} a_i u_i(x)\right)}, \\ C = g^{\gamma\delta\left(\sum\limits_{i=l+1}^{m} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x)) + r(\alpha+\beta)t(x) + \gamma t(x)(r^2 t(x) + h(x) + 2r\sum\limits_{i=0}^{m} a_i u_i(x)\right)}. \end{gathered} \tag{16}$$

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

Размер доказательства (16) составляет два элемента из $\mathbb{G}_1$ и один элемент из $\mathbb{G}_2$.

Алгоритм верификатора $0/1 \leftarrow \mathsf{Verify}(\mathsf{info}, \phi, \pi)$

С использованием билинейного спаривания верификатор проверяет, что компоненты A и B доказательства π (16) имеют общие показатели степеней. Также проверяется, что показатель степени компонента C доказательства π является произведением показателей первых двух компонент. Алгоритм Verify выполняет следующие шаги:

1. Выполняется выборка открытых значений входов логических элементов схемы ϕ: $(a_0, a_1, \ldots, a_l) \leftarrow \mathsf{Parse}(1, \phi)$.
2. Выполняется разделение компонент доказательства (16):

$$(A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \leftarrow \mathsf{Parse}(\pi).$$

3. Выполняется проверка уравнений верификации

$$\begin{gathered} e(Ag^{\alpha\delta}, Bh^{\beta\delta}) = e(g^{\alpha\delta}, h^{\beta\delta}) e\left(g^{\sum\limits_{i=0}^{l} a_i\delta(\gamma w_i(x) + (\alpha+\beta)u_i(x))}, h^{\gamma\delta}\right) e(C, h^{\delta}), \\ e(A, h) = e(g, B). \end{gathered} \tag{17}$$

   Компонента A однозначно определяет B через второе уравнение (17), а пара (A, B) однозначно определяет C через первое уравнение (17).
4. Если все проверки (17) выполняются, то выводится 1, иначе 0.

Алгоритм моделирования доказательства $\pi \leftarrow \mathsf{Sim}(\mathsf{bp}, \mathsf{sap}, \mathsf{crs}_{\mathsf{sap}}, \tau, \phi)$

За счёт «лазейки» τ возможно вывести корректные доказательства. Для этого выполняются следующие шаги:

1. Выполняется выборка открытых значений входов логических элементов схемы ϕ: $(a_0, a_1, \ldots, a_l) \leftarrow \mathsf{Parse}(1, \phi)$.
2. Выбирается случайное значение $\mu \in \mathbb{Z}_p$.
3. Вычисляются компоненты доказательства $(A, B = g^{\mu\delta}, h^{\mu\delta})$.
4. Вычисляется компонента $C = g^{(\mu^2\delta + (\alpha+\beta)\mu\delta - \gamma\delta \sum\limits_{i=0}^{l} a_i(\gamma w_i(x) + (\alpha+\beta)u_i(x)))}$.
5. На выход подаётся доказательство $\pi = (A, B, C)$.

Заключение

Рассмотрены способы устранения уязвимостей протоколов zk-SNARK, основанных на нарушении защищённости формирования CRS [5, 7, 11–15]. Представленные протоколы построения надёжных CRS являются самостоятельными и тяжеловесными конструкциями, которыми возможно расширять протоколы zk-SNARK, например [2, 3, 6]. Проверка корректности формируемых CRS, например [5, 7], происходит после выполнения всего протокола, что в случае выявления частых нарушений может оказаться недостатком и требует повторного выполнения всего процесса построения CRS. Описаны более защищённые версии прототоколов zk-SNARK [6, 3] в виде модернизированных схем [8–10]. Отмечается [24], что рассмотренные в работе протоколы zk-SNARK [2, 3, 6] имеют фиксированный размер доказательств и постоянное количество уравнений верификации, что дополнительно обосновывает целесообразность их практического применения. Проблемы защищённости других производительных протоколов zk-SNARK [24], например [19, 20, 25–29], в публичном доступе не представлены, поэтому их также можно рассматривать в качестве кандидатов на практическое применение.

ЛИТЕРАТУРА

1. *Мартыненков И. В.* Краткие неинтерактивные аргументы с нулевым разглашением на основе наборов полиномов // Прикладная дискретная математика. 2023. № 59. С. 34–72.
2. *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.
3. *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.
4. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5]. 2021. 213 p.
5. *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. Paper 2017/602. 2017. 25 p. `https://ia.cr/2017/602.`
6. *Groth J.* On the size of pairing-based non-interactive arguments // LNCS. 2016. V. 9666. P. 305–326.
7. *Bowe S., Gabizon A., and Miers I.* Scalable Multi-Party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive. Paper 2017/1050. 2017. 24 p. `https://eprint.iacr.org/2017/1050.`
8. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. Cryptology ePrint Archive. 2017. 36 p. `https://eprint.iacr.org/2017/540.pdf.`
9. *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.`

10. *Parno B.* A Note on the Unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive. Paper 2015/437. 2015. 4 p. https://eprint.iacr.org/2015/437.
11. *Maller M.* A Proof of Security for the Sapling Generation of zk-SNARK Parameters in the Generic Group Model. 2018. 12 p. https://github.com/zcash/saplingsecurity-analysis/blob/master/MaryMallerUpdated.pdf.
12. *Groth J., Kohlweiss M., Maller M., et al.* Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. Cryptology ePrint Archive. Paper 2018/280. 2018. 38 p. https://eprint.iacr.org/2018/280.
13. *Maller M., Bowe S., Kohlweiss M., and Meiklejohn S.* Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings. Cryptology ePrint Archive. Paper 2019/099. 2019. 20 p. https://eprint.iacr.org/2019/099.
14. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs // IEEE Symp. SP 2015. San Jose, CA, USA, 2015. P. 287–304.
15. *Campanelli M., Gennaro R., Goldfeder S., and Nizzardo L.* Zero-knowledge contingent payments revisited: Attacks and payments for services // Proc. ACM SIGSAC Conf. CCS'17. N.Y.: ACM, 2017. P. 229–243.
16. *Schnorr C.* Efficient identification and signatures for smart cards // LNCS. 1990. V. 435. P. 239–252.
17. *Черёмушкин А. В.* Криптографические протоколы. Основные свойства и уязвимости. М.: Издательский центр «Академия», 2009. 272 с.
18. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments // LNCS. 2010. V. 6477. P. 321–340.
19. *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.
20. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data // Proc. 2015 IEEE Symp. Security and Privacy. San Jose, CA, USA, 2015. P. 271–286.
21. *Fuchsbauer G.* Subversion-Zero-Knowledge Snarks. Cryptology ePrint Archive. Paper 2017/587. 2017. 32 p. https://eprint.iacr.org/2017/587.
22. zkSNARKs implementation in JavaScript & WASM. https://github.com/iden3/snarkjs.
23. C++ library for zkSNARKs. https://github.com/scipr-lab/libsnark.
24. *Мартыненков И. В.* Способы повышения производительности кратких неинтерактивных аргументов с нулевым разглашением и анализ достигнутых результатов // Прикладная дискретная математика. 2023. № 60. С. 40–58.
25. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs // LNCS. 2013. V. 7881. P. 626–645.
26. *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.
27. *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.
28. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves // LNCS. 2014. V. 8617. P. 276–294.
29. *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.

REFERENCES

1. *Martynenkov I. V.* Kratkie neinteraktivnye argumenty s nulevym razglasheniem na osnove naborov polinomov [Zero-knowledge succinct non-interactive arguments of knowledge based on sets of polynomials]. Prikladnaya Diskretnaya Matematika, 2023, no. 59, pp. 34–72. (in Russian)
2. *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.
3. *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.
4. *Hopwood D., Bowe S., Hornby T., and Wilcox N.* Zcash Protocol Specification. Version 2021.2.16 [NU5], 2021. 213 p.
5. *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. Paper 2017/602, 2017. 25 p. `https://ia.cr/2017/602.`
6. *Groth J.* On the size of pairing-based non-interactive arguments. LNCS, 2016. vol. 9666, pp. 305–326.
7. *Bowe S., Gabizon A., and Miers I.* Scalable Multi-Party Computation for zk-SNARK Parameters in the Random Beacon Model. Cryptology ePrint Archive. Paper 2017/1050, 2017. 24 p. `https://eprint.iacr.org/2017/1050.`
8. *Groth J. and Maller M.* Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. Cryptology ePrint Archive, 2017. 36 p. `https://eprint.iacr.org/2017/540.pdf.`
9. *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.`
10. *Parno B.* A Note on the Unsoundness of vnTinyRAM's SNARK. Cryptology ePrint Archive. Paper 2015/437, 2015. 4 p. `https://eprint.iacr.org/2015/437.`
11. *Maller M.* A Proof of Security for the Sapling Generation of zk-SNARK Parameters in the Generic Group Model. 2018. 12 p. `https://github.com/zcash/saplingsecurity-analysis/blob/master/MaryMallerUpdated.pdf.`
12. *Groth J., Kohlweiss M., Maller M., et al.* Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. Cryptology ePrint Archive. Paper 2018/280, 2018. 38 p. `https://eprint.iacr.org/2018/280.`
13. *Maller M., Bowe S., Kohlweiss M., and Meiklejohn S.* Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings. Cryptology ePrint Archive. Paper 2019/099, 2019. 20 p. `https://eprint.iacr.org/2019/099.`
14. *Ben-Sasson E., Chiesa A., Green M., et al.* Secure sampling of public parameters for succinct zero knowledge proofs. IEEE Symp. SP 2015, San Jose, CA, USA, 2015, pp. 287–304.
15. *Campanelli M., Gennaro R., Goldfeder S., and Nizzardo L.* Zero-knowledge contingent payments revisited: Attacks and payments for services. Proc. ACM SIGSAC Conf. CCS'17, N.Y., ACM, 2017, pp. 229–243.
16. *Schnorr C.* Efficient identification and signatures for smart cards. LNCS, 1990, vol. 435, pp. 239–252.
17. *Cheremushkin A. V.* Kriptograficheskie protokoly. Osnovnye svojstva i uyazvimosti [Cryptographic Protocols. Basic Properties and Vulnerabilities]. Moscow, Akademiya Publ., 2009. 272 p. (in Russian)
18. *Groth J.* Short pairing-based non-interactive zero-knowledge arguments. LNCS, 2010, vol. 6477, pp. 321–340.

19. *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`.
20. *Backes M., Barbosa M., Fiore D., and Reischuk R. M.* ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data. Proc. 2015 IEEE Symp. Security and Privacy, San Jose, CA, USA, 2015, pp. 271–286.
21. *Fuchsbauer G.* Subversion-Zero-Knowledge Snarks. Cryptology ePrint Archive. Paper 2017/587, 2017. 32 p. `https://eprint.iacr.org/2017/587`.
22. zkSNARKs implementation in JavaScript & WASM. `https://github.com/iden3/snarkjs`.
23. C++ library for zkSNARKs. `https://github.com/scipr-lab/libsnark`.
24. *Martynenkov I. V.* Sposoby povysheniya proizvoditel'nosti kratkikh neinteraktivnykh argumentov s nulevym razglasheniem i analiz dostignutykh rezul'tatov [Ways to improve the performance of zero-knowledge succinct non-interactive arguments of knowledge and analysis of the results achieved]. Prikladnaya Diskretnaya Matematika, 2023, no. 60, pp. 40–58. (in Russian)
25. *Gennaro R., Gentry C., Parno B., and Raykova M.* Quadratic span programs and succinct NIZKs without PCPs. LNCS, 2013, vol. 7881, pp. 626–645.
26. *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.
27. *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.
28. *Ben-Sasson E., Chiesa A., Tromer E., and Virza M.* Scalable zero knowledge via cycles of elliptic curves. LNCS, 2014, vol. 8617, pp. 276–294.
29. *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.