МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2023 № 59

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

Адрес редакции и издателя: 634050, г. Томск, пр. Ленина, 36
E-mail: pank@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Редактор-переводчик *Т. В. Бутузова*
Верстка *И. А. Панкратовой*

Подписано к печати 13.03.2023. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 14,8. Тираж 300 экз.
Заказ № 5369. Цена свободная. Дата выхода в свет 17.03.2023.

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

СОДЕРЖАНИЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 512.54.05, 512.543.7 DOI 10.17223/20710410/59/1

ОБ УРАВНЕНИЯХ В СВОБОДНЫХ МОНОИДАХ И ПОЛУГРУППАХ С ОГРАНИЧЕНИЯМИ НА РЕШЕНИЯ[1]

В. Г. Дурнев, А. И. Зеткина

Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия

E-mail: durnev@unuyar.ac.ru, a.zetkina1@uniyar.ac.ru

Изучаются алгоритмические проблемы для уравнений в свободных моноидах и полугруппах (уравнения в словах и длинах) с дополнительными ограничениями на решения. Доказывается, что невозможно построить алгоритм, позволяющий по произвольной системе уравнений в словах и длинах в свободном моноиде (свободной полугруппе) ранга 2 с одним дополнительным ограничением на решение в форме принадлежности одной его компоненты языку сбалансированных слов или равенства проекций двух компонент решения на одну выделенную свободную образующую определить, имеет ли она решение. Аналогичный результат установлен для систем неравенств в словах.

Ключевые слова: *системы уравнений в свободных моноидах и свободных полугруппах, уравнения в словах и длинах, уравнения с ограничениями на решения.*

ON EQUATIONS IN FREE MONOIDS AND SEMIGROUPS WITH RESTRICTIONS ON SOLUTIONS

V. G. Durnev, A. I. Zetkina

P. G. Demidov Yaroslavl State University, Yaroslavl, Russia

We study algorithmic problems for equations in free monoids and semigroups (equations in words and lengths) with additional restrictions on the solutions. It is proved that it is impossible to construct an algorithm that solves an arbitrary system of equations in words and lengths in a free monoid (free semigroup) of rank 2 with an additional constraint on the solution in the form that one of its components belongs to the language of balanced words or the equality of the projections of two components of the solution into a distinguished free generator to determine whether it has a solution. A similar result is obtained for systems of inequalities in words.

Keywords: *systems of equations in free monoids and free semigroups, equations in words and lengths, equations with restrictions on solutions.*

[1]Работа поддержана грантом РФФИ № 19-52-26006.

Введение

Содержащиеся в работе результаты анонсированы без доказательств в тезисах [1, 2]. Приводится их подробное доказательство. Кроме того, теорема 3 существенно усиливает анонсированный в тезисах результат. По нашему мнению, этот результат максимально близок к окончательному.

Через M_m будем обозначать свободный моноид, т. е. свободную полугруппу ранга m с пустым словом в качестве нейтрального элемента и со свободными образующими $a_1, \ldots, a_m$, а через F_m — свободную группу с теми же свободными образующими. Через S_m будем обозначать свободную полугруппу без пустого слова с теми же самыми свободными образующими $a_1, \ldots, a_m$. Доказанные в работе теоремы справедливы одновременно для свободного моноида M_m и для свободной полугруппы S_m при $m \geqslant 2$. Доказательство проводится, как правило, для свободного моноида M_2. Его легко перенести на свободный моноид M_m и свободную полугруппу S_m для любого $m \geqslant 2$. Заметим, что почти до конца XX в. в нашей стране использовались понятия «*свободная полугруппа с пустым словом в качестве нейтрального элемента*», «*свободная полугруппа с нейтральным элементом*» и «*свободная полугруппа с единицей*». Большинство результатов в этой области не зависело от того, рассматривается ли свободная полугруппа с единицей или без неё. И только в работе Н. А. Перязева [3] был получен важный результат, справедливый для M_2, но не для S_2.

К концу XX в. под влиянием работ группы Н. Бурбаки понятие «*свободная полугруппа с единицей*» стало заменяться более кратким понятием «*свободный моноид*». Одним из первых его стал употреблять Н. А. Перязев [3].

Свободные образующие a_1 и a_2 свободного моноида M_2 будем обозначать через a и b соответственно.

Определение 1. *Системой уравнений* с неизвестными $x_1, \ldots, x_n$ в свободном моноиде M_m (свободной полугруппе S_m) называется выражение вида

$$\mathop{\&}_{i=1}^{k}\big(w_i(x_1, \ldots, x_n, a_1, \ldots, a_m) \,=\, u_i(x_1, \ldots, x_n, a_1, \ldots, a_m)\big), \tag{1}$$

где $w_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$ и $u_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$ — слова в алфавите

$$\{\, x_1, x_2, \ldots, x_n, a_1, a_2, \ldots, a_m \,\}.$$

Определение 2. Набор $\langle g_1, \ldots, g_n \rangle$ элементов свободного моноида M_m (свободной полугруппы S_m) называется *решением* системы (1), если при любом i $(i = 1, \ldots, k)$ в свободном моноиде M_m (свободной полугруппе S_m) выполняется равенство

$$w_i(g_1, \ldots, g_n, a_1, \ldots, a_m) \,=\, u_i(g_1, \ldots, g_n, a_1, \ldots, a_m).$$

Определение 3. Две системы уравнений с одними и теми же неизвестными называются *эквивалентными*, если множества их решений совпадают.

Заметим, что при $m \geqslant 2$ система уравнений $\mathop{\&}_{i=1}^{k}\big(w_i \,=\, u_i\big)$ равносильна одному уравнению

$$w_1 a_1 w_2 a_1 \ldots a_1 w_k \, w_1 a_2 w_2 a_2 \ldots a_2 w_k \,=\, u_1 a_1 u_2 a_1 \ldots a_1 u_k \, u_1 a_2 u_2 a_2 \ldots a_2 u_k.$$

Эта равносильность позволяет удалять из формул знак конъюнкции &.

Для удаления из формул знака дизъюнкции $\vee$ можно применить установленную Н. К. Косовским в работе [4] эквивалентность ($m \geqslant 2$)

$$M_m \vdash (\forall x_1)(\forall x_2)(\forall x_3)(\forall x_4)((x_1 = x_2 \vee x_3 = x_4) \iff$$
$$\iff (\exists y_1)(\exists y_2)(\exists y_3)(\exists y_4) w(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, a_1, a_2) =$$
$$= u(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, a_1, a_2)),$$

где $w(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, a_1, a_2)$ и $u(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, a_1, a_2)$ — некоторые слова в алфавите $\{x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, a_1, a_2\}$.

В работе [5] показано, что число новых переменных y_1, y_2, y_3 и y_4 можно уменьшить до двух — построены слова $w(x_1, x_2, x_3, x_4, y_1, y_2, a_1, a_2)$ и $u(x_1, x_2, x_3, x_4, y_1, y_2, a_1, a_2)$ в алфавите $\{x_1, x_2, x_3, x_4, y_1, y_2, a_1, a_2\}$, для которых справедлива эквивалентность

$$M_m \vdash (\forall x_1)(\forall x_2)(\forall x_3)(\forall x_4)((x_1 = x_2 \vee x_3 = x_4) \iff$$
$$\iff (\exists y_1)(\exists y_2) w(x_1, x_2, x_3, x_4, y_1, y_2, a_1, a_2) = u(x_1, x_2, x_3, x_4, y_1, y_2, a_1, a_2)).$$

1. Алгоритмические проблемы для уравнений в свободных моноидах и полугруппах (обзор результатов)

В 60-е годы XX в. А. А. Марков предложил использовать системы уравнений в свободной полугруппе S_m (в свободном моноиде M_m) в качестве одного из подходов к отрицательному решению 10-й проблемы Д. Гильберта.

Системы уравнений в свободных полугруппах (в свободных моноидах) также называются *системами уравнений в словах*. Первые результаты в исследовании систем уравнений в словах были получены А. А. Марковым (не опубликовано) и Ю. И. Хмелевским [6] в конце 60-х годов прошлого века.

В эти же годы было начато изучение систем уравнений в словах и длинах, т. е. систем вида

$$\mathop{\&}_{i=1}^{m} (w_i = u_i) \;\&\; \mathop{\&}_{\{i,j\} \in A} (|x_i| = |x_j|),$$

где через $|x| = |y|$ обозначен предикат «*длины слов x и y равны*». Первые результаты в исследовании систем уравнений в словах и длинах получены в начале 70-х годов в работах Ю. В. Матиясевича [7] и Н. К. Косовского [8, 9].

Для слова w в алфавите Σ и буквы a этого алфавита через $|w|_a$ будем обозначать число вхождений буквы a в слово w.

В 1972—1973 гг. первый автор начал рассматривать системы уравнений в словах и длинах с дополнительным предикатом $|x|_a = |y|_a$ — «*проекции слов x и y на выделенную букву a равны*». В работе [10], в частности, доказано, что можно указать такое однопараметрическое семейство систем уравнений в свободной нециклической полугруппе с единицей или без неё

$$\big(w(x, x_1, \ldots, x_n, a, b) = v(x, x_1, \ldots, x_n, a, b)\big) \;\&\; \mathop{\&}_{\{i,j\} \in A} (|x_i| = |x_j| \,\&\, |x_i|_a = |x_j|_a)$$

с неизвестными $x_1, \ldots, x_n$, константами a и b и параметром x, где A — некоторое подмножество множества $\{\{t, s\} : 1 \leqslant t, s \leqslant n\}$, что невозможен алгоритм, позволяющий для произвольного натурального m определить, имеет ли решение система уравнений

$$w(a^m, x_1, \ldots, x_n, a, b) = v(a^m, x_1, \ldots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\} \in A} (|x_i| = |x_j| \;\&\; |x_i|_a = |x_j|_a).$$

В этой же работе отмечено, что результат остаётся верным, если предикат $|x| = |y|$ & & $|x|_a = |y|_a$ заменить предикатом $|x|_b = |y|_b$ & $|x|_a = |y|_a$.

Аналогичный результат содержится в работе [11], опубликованной в 1988 г.

В 1976 г. Г. С. Маканин получил в теории уравнений в словах фундаментальный результат [12, 13] — он построил алгоритм, позволяющий по произвольной системе уравнений в свободной полугруппе $S(m)$ (свободном моноиде M_m) определить, имеет ли она решение. Несколько позже в работе [14] Г. С. Маканин построил алгоритм, позволяющий по произвольной системе уравнений в свободной группе F_m определить, имеет ли она решение.

После фундаментальных результатов Г. С. Маканина особый интерес стал представлять вопрос о существовании аналогичных алгоритмов для уравнений в свободных моноидах, полугруппах и группах с различными «не слишком сложными» и «достаточно естественными» ограничениями на решения.

В работах [15, 16] доказана алгоритмическая неразрешимость позитивной $\exists\forall\exists^3$-теории любой конечно порождённой нециклической свободной полугруппы (с пустым словом или без него).

Вопрос о разрешимости позитивной теории свободной полугруппы счётного ранга в кандидатской диссертации первого автора был легко сведён к следующей проблеме:

Существует ли алгоритм, позволяющий для произвольного уравнения

$$w(x_1, \ldots, x_n, a_1, \ldots, a_m) = u(x_1, \ldots, x_n, a_1, \ldots, a_m)$$

в свободной полугруппе счётного ранга (с пустым словом или без него) определить, имеет ли оно такое решение g_1, …, g_n, что

$$g_1 \in M_{m_1},\ g_2 \in M_{m_2},\ \ldots,\ g_n \in M_{m_n},$$

где $m_1 \leqslant m_2 \leqslant \ldots \leqslant m_n$; M_{m_i} — свободная полугруппа с образующими $a_1, \ldots, a_{m_i}$?

Ю. М. Важенин и Б. В. Розенблат в работе [17], используя результат Г. С. Маканина, доказали, что существует алгоритм для решения последней задачи, это позволило им установить *разрешимость позитивной теории свободной полугруппы счётного ранга.*

Аналогичным образом вводятся понятия системы уравнений в свободной группе и её решения.

Определение 4. *Системой уравнений* с неизвестными $x_1, \ldots, x_n$ в свободной группе F_m называется выражение вида

$$\mathop{\&}_{i=1}^{k} w_i(x_1, \ldots, x_n, a_1, \ldots, a_m) = u_i(x_1, \ldots, x_n, a_1, \ldots, a_m), \tag{2}$$

где $w_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$ и $u_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$ — слова в алфавите

$$\{\, x_1, x_1^{-1}, x_2, x_2^{-1}, \ldots, x_n, x_n^{-1}, a_1, a_1^{-1}, a_2, a_2^{-1}, \ldots, a_m, a_m^{-1}, \,\}.$$

Определение 5. Набор $\langle g_1, \ldots, g_n \rangle$ элементов свободной группы F_m называется *решением* системы (2), если при любом i $(i = 1, \ldots, k)$ в свободной группе F_m выполняется равенство

$$w_i(g_1, \ldots, g_n, a_1, \ldots, a_m) = u_i(g_1, \ldots, g_n, a_1, \ldots, a_m).$$

Определение 6. Две системы уравнений с одними и теми же неизвестными называются *эквивалентными*, если множества их решений совпадают.

Используя уравнение А. И. Мальцева (приводится в [18]) $[x, a_1] = ([x, a_2]\, y^2)^2$, имеющее в свободной группе F_m при $m \geqslant 2$ лишь тривиальное решение $x = 1$ и $y = 1$, можно любую систему уравнений $\mathop{\&}\limits_{i=1}^{k} w_i = u_i$ в свободной группе F_m заменить одним равносильным ей уравнением $w(x_1, \ldots, x_n, a_1, \ldots, a_m) = 1$ (без введения новых переменных). Это позволяет удалять из формул, относящихся к свободной группе, знак конъюнкции & без введения новых переменных.

Г. А. Гуревич (доказательство приведено Г. С. Маканиным в работе [18]) установил, что и дизъюнкцию уравнений

$$\bigvee_{i=1}^{k} w_i(x_1, \ldots, x_n, a_1, \ldots, a_m) = u_i(x_1, \ldots, x_n, a_1, \ldots, a_m)$$

в свободной группе F_m можно заменить одним равносильным ей уравнением $u(x_1, \ldots, x_n, a_1, \ldots, a_m) = 1$ (без введения новых переменных). Доказано, что для свободной группы F_m справедлива эквивалентность

$$F_m \vdash (\forall x_1)(\forall x_2)\Big(\mathop{\&}_{\varepsilon,\delta=\pm 1} (x_1 a_1^{\varepsilon} x_1 a_1^{-\varepsilon})(x_2 a_2^{\delta} x_2 a_2^{-\delta}) = (x_2 a_2^{\delta} x_2 a_2^{-\delta})(x_1 a_1^{\varepsilon} x_1 a_1^{-\varepsilon}) \iff$$
$$\iff (x_1 = 1 \vee x_2 = 1)\Big).$$

Это позволяет удалять из формул, относящихся к свободной группе F_m при $m \geqslant 2$, знак дизъюнкции $\vee$ без введения новых переменных.

Вопрос о разрешимости позитивной теории свободной группы был сведен Ю. И. Мерзляковым [19] к следующей проблеме:

Существует ли алгоритм, позволяющий для произвольного уравнения

$$w(x_1, \ldots, x_n, a_1, \ldots, a_m) = 1$$

в свободной группе счётного ранга определить, имеет ли оно такое решение $g_1, \ldots, g_n$, что

$$g_1 \in F_{m_1},\ g_2 \in F_{m_2},\ \ldots,\ g_n \in F_{m_n},$$

где $m_1 \leqslant m_2 \leqslant \ldots \leqslant m_n$; F_{m_i} — свободная группа с образующими $a_1, \ldots, a_{m_i}$?

Г. С. Маканин [18] построил искомый алгоритм и тем самым доказал *разрешимость позитивной теории свободной группы*. Это был первый шаг на пути решения известной проблемы А. Тарского о разрешимости элементарной теории свободной нециклической группы.

Обобщая эти ситуации, Г. С. Маканин поставил в «Коуровской тетради» [20] следующую проблему для уравнений в свободных группах:

9.25. Указать алгоритм, который по уравнению

$$w(x_1, \ldots, x_n, a_1, \ldots, a_m) = 1$$

в свободной группе F_m и списку конечно порождённых подгрупп $H_1, \ldots, H_n$ группы F_m позволял бы узнать, существует ли решение этого уравнения с условием

$$x_1 \in H_1,\ \ldots,\ x_n \in H_n.$$

Первые положительные результаты в направлении решения этой проблемы получил А. Ш. Малхасян в работе [21].

К. Шульц [22] рассмотрел аналогичную проблему для уравнений в свободных моноидах (свободных полугруппах) с регулярными ограничениями на решения и доказал, что существует алгоритм, который по уравнению

$$w(x_1, \ldots, x_n, a_1, \ldots, a_m) = u(x_1, \ldots, x_n, a_1, \ldots, a_m)$$

в свободном моноиде M_m и списку регулярных подмножеств (языков) $H_1, \ldots, H_n$ моноида M_m позволяет узнать, существует ли решение этого уравнения с условием

$$x_1 \in H_1, \ \ldots, \ x_n \in H_n.$$

Так как каждая конечно порождённая подполугруппа (с единицей) свободного моноида M_m является регулярным подмножеством (языком), то решённая К. Шульцем проблема для уравнений с ограничениями на решения в свободных полугруппах является естественным аналогом проблемы Г. С. Маканина.

V. Diekert в работах [23, 24] построил алгоритм, позволяющий по произвольному уравнению

$$w(x_1, \ldots, x_n, a_1, \ldots, a_m) = 1$$

в свободной группе F_m и списку регулярных подмножеств (языков) $H_1, \ldots, H_n$ группы F_m определить, существует ли решение этого уравнения с условием

$$x_1 \in H_1, \ \ldots, \ x_n \in H_n.$$

Тем самым решена и проблема Г. С. Маканина.

Сказанное позволяет считать, что дальнейшее исследование различных обобщений проблемы Г. С. Маканина для свободных групп, моноидов и полугрупп, получающихся путём ослабления ограничений, налагаемых на подгруппы (подполугруппы, подмоноиды, языки) $H_1, \ldots, H_n$, представляет большой интерес.

В силу теоремы К. Шульца для получения алгоритмически неразрешимых проблем для уравнений в свободных моноидах (полугруппах) с подполугрупповыми ограничениями на решения необходимо рассматривать, в первую очередь, бесконечно порождённые свободные подполугруппы, среди которых имеются как нерегулярные, так и регулярные языки, например подполугруппа, порождённая всевозможными словами вида $ab^n a$ $(n = 1, 2, \ldots)$, свободно ими порождается и является регулярным языком.

2. Алгоритмические проблемы для уравнений в свободных моноидах и полугруппах (новые результаты)

В литературе по формальным языкам и грамматикам достаточно часто встречается рекурсивный язык L_1 в алфавите $\{a, b\}$, который состоит из всех слов w, для которых $|w|_a = |w|_b$, — язык *сбалансированных*, или *уравновешенных*, *слов* в алфавите $\{a, b\}$. Пользуясь известным критерием свободности для подполугрупп свободной полугруппы, легко доказать, что L_1 — *свободная подполугруппа счётного ранга*. Конечно, рекурсивный язык L_1 не является регулярным, однако с точки зрения сложности разрешимости для него алгоритмических проблем он скорее «ближе» к регулярным языкам, чем к произвольным рекурсивным.

Поэтому представляют интерес, на наш взгляд, следующие две теоремы.

Теорема 1. Можно указать такое однопараметрическое семейство уравнений с ограничениями на решения в свободном моноиде M_2 (в свободной полугруппе S_2)

$$w(x, x_1, \ldots, x_n, a, b) = v(x, x_1, \ldots, x_n, a, b) \ \& \ \mathop{\&}_{\{i,j\} \in A} |x_i| = |x_j| \ \& \ |x_1|_b = |x_2|_b$$

с неизвестными $x_1, \ldots, x_n$, константами a и b и параметром x, где A — некоторое подмножество множества $\{\{t, s\} : 1 \leqslant t, s \leqslant n\}$, что невозможен алгоритм, позволяющий для произвольного натурального числа m определить, имеет ли решение уравнение с ограничениями на решения

$$w(a^m, x_1, \ldots, x_n, a, b) = v(a^m, x_1, \ldots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \;\&\; |x_1|_b = |x_2|_b.$$

Теорема 2. Можно указать такое однопараметрическое семейство уравнений с ограничениями на решения в свободном моноиде M_2 (в свободной полугруппе S_2)

$$w(x, x_1, \ldots, x_n, a, b) = v(x, x_1, \ldots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \;\&\; x_1 \in L_1$$

с неизвестными $x_1, \ldots, x_n$, константами a и b и параметром x, где A — некоторое подмножество множества $\{\{t, s\} : 1 \leqslant t, s \leqslant n\}$, что невозможен алгоритм, позволяющий для произвольного натурального числа m определить, имеет ли решение уравнение с ограничениями на решения

$$w(a^m, x_1, \ldots, x_n, a, b) = v(a^m, x_1, \ldots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \;\&\; x_1 \in L_1.$$

Доказательство теорем 1 и 2 проведём по единой схеме с некоторыми отличиями. Прежде всего в позитивной $\exists$-теории моноида M_2 (полугруппы S_2) с использованием предиката $|x| = |y|$ равенства длин выразим ряд вспомогательных предикатов.

В дальнейшем будем работать с натуральными числами, а не с целыми неотрицательными, чтобы дать единые доказательства и для свободного моноида M_2 (свободной полугруппы с пустым словом), и для свободной полугруппы без пустого слова S_2.

Пусть α — это буква a или b,

$$\mathbb{N}_\alpha(x) \;\rightleftharpoons\; (x\,\alpha = \alpha\,x \;\&\; (\exists y)\, x = ay).$$

Если X — элемент моноида M_2, то формула $\mathbb{N}_\alpha(X)$ истинна на моноиде M_2 тогда и только тогда, когда X — натуральная степень буквы α.

Для свободной полугруппы S_2 эта формула примет более простой вид:

$$\mathbb{N}_\alpha(x) \;\rightleftharpoons\; (x\,\alpha = \alpha\,x).$$

Рассмотрим предикат $R(x, y)$, истинный тогда и только тогда, когда найдётся такое натуральное число t, что $x = a^t$, а $y = b^t$. Справедлива эквивалентность

$$R(x, y) \iff \mathbb{N}_a(x) \;\&\; \mathbb{N}_b(y) \;\&\; |x| = |y|.$$

Введём необходимый для дальнейшего предикат делимости $D(x, y)$, истинный тогда и только тогда, когда найдутся такие натуральные числа s и t, что $x = a^s$, $y = a^t$ и s делит t. Справедлива эквивалентность

$$D(x, y) \iff \Big(\mathbb{N}_a(x) \;\&\; \mathbb{N}_a(y) \;\&\; (\exists v)(\exists u)\big(x = av \;\&\; u(vb) = (vb)u \;\&\; |u| = |y|\big)\Big).$$

Для свободной полугруппы S_2 надо несколько изменить эту эквивалентность — она примет следующий вид:

$$D(x, y) \iff \Big(\mathbb{N}_a(x) \;\&\; \mathbb{N}_a(y) \;\&\; (x = a \vee (\exists v)(\exists u)\big(x = av \;\&\; u(vb) = (vb)u \;\&\; |u| = |y|\big)\Big).$$

Введём основной для дальнейшего предикат $M(x,y,z)$, истинный тогда и только тогда, когда найдутся такие натуральные числа s, t и r, что $x=a^s$, $y=a^t$, $z=a^r$ и $st=r$. Имеют место такие эквивалентности:

$$M(x,y,z) \iff \Big(\mathbb{N}_a(x)\ \&\ \mathbb{N}_a(y)\ \&\ \mathbb{N}_a(z)\ \&$$

$$\&\ (\exists v)(\exists u)(\exists w)\big(u(bx)=(bx)u\ \&\ R(y,v)\ \&\ w=vz\ \&\ |u|=|w|\ \&\ |u|_b=|w|_b\big)\Big),$$

$$M(x,y,z) \iff \Big(\mathbb{N}_a(x)\ \&\ \mathbb{N}_a(y)\ \&\ \mathbb{N}_a(z)\ \&\ (\exists v)(\exists u)(\exists w)(\exists p)(\exists q)\big(u(bx)=(bx)u\ \&$$

$$\&\ R(y,v)\ \&\ p=vz\ \&\ |u|=|p|\ \&\ R(z,w)\ \&\ q=uyw\ \&\ q\in L_1\big)\Big).$$

Воспользуемся следующим вариантом непосредственного следствия фундаментальной теоремы Ю. В. Матиясевича [25] о диофантовости рекурсивно перечислимых множеств:

Для произвольного рекурсивно перечислимого множества натуральных чисел можно построить такую формулу $\Phi_A(x_1)$ вида

$$(\exists x_2)\ldots(\exists x_m)\ \Psi,$$

где $\Psi=\mathop{\&}\limits_{i=1}^{s}\varphi_i$ и каждая формула φ_i имеет один из следующих видов:

$$x_l+x_j=x_k,\quad x_j=x_l,\quad x_lx_j=x_k,\quad x_j=c$$

(здесь c — натуральное число), что для произвольного натурального числа n имеем $n\in A$ тогда и только тогда, когда формула $\Phi_A(n)$ истинна на множестве натуральных чисел.

Воспользовавшись хорошо известной эквивалентностью

$$xy=z \quad\iff\quad (x+y)^2=x^2+2z+y^2,$$

можно считать, что в формулу $\Phi_A(x_1)$ подформулы вида $x_l\,x_j=x_t$ входят лишь при $l=j$, т. е. они имеют вид $x_l^2=x_t$.

Используем следующее утверждение, принадлежащее Дж. Робинсон [26], на которое любезно обратил наше внимание анонимный рецензент одной нашей статьи:

Если m, n и L — натуральные числа, $m\leqslant n$ и $L>n^2$, то $(L+m)\,|\,(L^2-n)$ тогда и только тогда, когда $n=m^2$.

Условие $L>n^2$ можно заменить условием

$$(n+1)\,|\,L\ \&\ (n+2)\,|\,L.$$

Это позволяет в формуле $\Phi_A(x_1)$ заменить подформулу $\mathop{\&}\limits_{i=1}^{p}x_{l_i}^2=x_{t_i}$ на подформулу

$$U=Y^2\ \&\ Z=\sum_{i=1}^{p}x_{t_i}\ \&\ (Z+1)\,|\,Y\ \&\ (Z+2)\,|\,Y\ \&$$

$$\&\ \mathop{\&}_{i=1}^{p}x_{t_i}=x_{l_i}+u_i\ \&\ \mathop{\&}_{i=1}^{p}(Y+x_{l_i})\,|\,(U-x_{t_i}),$$

где U, Y и Z — новые переменные. Поэтому можно считать, что в формуле $\Phi_A(x_1)$ лишь одна подформула φ_i имеет вид $x_l^2=x_k$, а все остальные подформулы φ_i имеют один из следующих видов (c — натуральное число):

$$x_l+x_j=x_k,\quad x_j=x_k,\quad x_l\,|\,x_j,\quad x_j=c.$$

По формуле $\Phi_A(x_1)$ построим формулу $\Phi_A^{(1)}(x_1)$:

$$\Phi_A^{(1)}(x_1) \rightleftharpoons (\exists x_2)\dots(\exists x_m)\left(\Psi_1 \;\&\; \left(\mathop{\&}_{i=2}^{m}\mathbb{N}_a(x_i)\right)\right).$$

Здесь Ψ_1 получается из Ψ заменой каждой подформулы φ_i вида $x_l + x_j = x_k$ на $x_l x_j = x_k$, вида $x_l \,|\, x_j$ — на $D(x_l, x_j)$, вида $x_j = x_k$ — на $x_j = x_k$, вида $x_j = c$ — на $x_j = a^c$ и вида $x_l^2 = x_k$ — на $M(x_l, x_l, x_k)$. Напомним, что подформула последнего вида лишь одна и только в неё, причём лишь один раз, входит предикат $|x|_b = |y|_b$ или предикат $x \in L_1$.

Подходящим образом переименовав переменные в формуле $\Phi_A^{(1)}(x_1)$, можем считать, что в формулу $\Phi_A^{(1)}(x)$ входят лишь переменные $x, x_1, \dots, x_n$.

Удалим из формулы $\Phi_A^{(1)}(x)$ конъюнкцию $\&$ (а в случае свободной полугруппы S_2 и дизъюнкцию $\vee$, при этом не будем выделять новые переменные) и приведём её к виду

$$w(x, x_1, \dots, x_n, a, b) = v(x, x_1, \dots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\}\in A}|x_i| = |x_j| \;\&\; |x_1|_b = |x_2|_b$$

или виду

$$w(x, x_1, \dots, x_n, a, b) = v(x, x_1, \dots, x_n, a, b) \;\&\; \mathop{\&}_{\{i,j\}\in A}|x_i| = |x_j| \;\&\; x_1 \in L_1.$$

Полученную формулу обозначим через $\Phi_A^{(2)}(x)$. Тогда $k \in A$, если и только если формула $\Phi_A^{(2)}(a^k)$ истинна на свободном моноиде M_2 (на свободной полугруппе S_2).

Для завершения доказательства теорем достаточно взять в качестве A рекурсивно перечислимое, но нерекурсивное множество натуральных чисел. ■

3. Алгоритмические проблемы для систем неравенств в свободных моноидах и полугруппах

V. Diekert предложил (устное сообщение Ю. В. Матиясевича) изучать в свободных моноидах и свободных полугруппах системы неравенств вида

$$\mathop{\&}_{i=1}^{k} w_i(x_1, \dots, x_n, a_1, \dots, a_m) \leqslant u_i(x_1, \dots, x_n, a_1, \dots, a_m), \tag{3}$$

где для слов w и u в алфавите образующих свободного моноида запись $w \leqslant u$ означает, что последовательность букв w является подпоследовательностью букв u, т. е. существуют такое число $k \leqslant |w|$ и такие слова $w_1, \dots, w_k, u_1, \dots, u_k, u_{k+1}$, что

$$w = w_1 \dots w_k, \quad u = u_1 w_1 u_2 \dots u_k w_k u_{k+1}.$$

Системы (3) можно рассматривать как обобщение систем уравнений (1), так как $w = u$ тогда и только тогда, когда $w \leqslant u \;\&\; u \leqslant w$.

Отношение $w \leqslant u$ является отношением частичного порядка на свободном моноиде M_m (свободной полугруппе S_m), т. е. оно рефлексивно, транзитивно и антисимметрично. Это ещё один довод для обоснования естественности рассмотрения систем неравенств вида (1).

Вопрос об алгоритмической разрешимости проблемы совместности для систем неравенств (3) в настоящее время открыт. Но если к отношению $w \leqslant u$ добавить предикат равенства длин, то получим алгоритмически неразрешимую задачу.

В дальнейшем равенство $w = u$ будем использовать как сокращённую запись конъюнкции неравенств $w \leqslant u \;\&\; u \leqslant w$.

Теорема 3. Невозможен алгоритм, позволяющий для произвольной системы неравенств с одним ограничением вида

$$\mathop{\&}_{i=1}^{k} w_i(x_1,\dots,x_n,a,b) \leqslant u_i(x_1,\dots,x_n,a,b) \ \&\ |x_1| = |x_2|$$

определить, имеет ли она решение в свободном моноиде M_2.

Доказательство. Докажем теорему по той же схеме, что и теоремы 1 и 2, но с некоторыми изменениями.

Предикаты $R(x,y)$ и $M(x,y,z)$ зададим эквивалентностями

$$R(x,y) \iff \mathbb{N}_a(x)\ \&\ \mathbb{N}_b(y)\ \&\ (\exists h)((ab)h = h(ab)\ \&\ x \leqslant h\ \&\ y \leqslant h\ \&\ |xy| = |h|,$$

$$M(x,y,z) \iff \Big(\mathbb{N}_a(x)\ \&\ \mathbb{N}_a(y)\ \&\ \mathbb{N}_a(z)\ \&$$

$$\&\ (\exists v)(\exists u)(\exists w)\big(u(bx) = (bx)u\ \&\ R(y,v)\ \&\ w = vz\ \&\ z \leqslant u\ \&\ v \leqslant u\ \&\ |w| = |u|\big)\Big).$$

Покажем, что предикат $M(x,y,z)$ истинен тогда и только тогда, когда найдутся такие натуральные числа s, t и r, что $x = a^s$, $y = a^t$, $z = a^r$ и $st = r$.

Если предикат $M(x,y,z)$ истинен, то существуют такие натуральные числа s, t и r, что $x = a^s$, $y = a^t$, $z = a^r$. Остаётся показать, что $st = r$. Нетрудно видеть, что $u = (ba^s)^l$ для некоторого натурального числа l, $v = b^t$ и $w = b^t a^r$. Тогда неравенства $z \leqslant u\ \&\ v \leqslant u$ влекут неравенства $r \leqslant sl\ \&\ t \leqslant l$, а равенство $|w| = |u|$ даёт равенство $r + t = sl + l$. Поэтому $r = sl\ \&\ t = l$, значит, $r = st$. Обратное утверждение очевидно.

Заметим, что справедлива следующая эквивалентность, которая далее будет значительно усилена:

$$M(x,y,z) \iff \Big(\mathbb{N}_a(x)\ \&\ \mathbb{N}_a(y)\ \&\ \mathbb{N}_a(z)\ \&$$

$$\&\ (\exists v)(\exists u)(\exists w)(\exists h)\big(\mathbb{N}_b(v)\ \&\ (ab)h = h(ab)\ \&\ y \leqslant h\ \&\ v \leqslant h\ \&\ u(bx) = (bx)u\ \&$$

$$\&\ w = vz\ \&\ z \leqslant u\ \&\ v \leqslant u\ \&\ |wyv| = |uh|\big)\Big).$$

В правой части эквивалентности только одно вхождение предиката равенства длин.

Как и ранее, воспользуемся следующим вариантом непосредственного следствия фундаментальной теоремы Ю. В. Матиясевича [25] о диофантовости рекурсивно перечислимых множеств:

Для произвольного рекурсивно перечислимого множества A натуральных чисел можно построить такую формулу $\Phi_A(x_1)$ вида

$$(\exists x_2)\dots(\exists x_m)\ \Psi,$$

где $\Psi = \mathop{\&}_{i=1}^{s} \varphi_i$ и каждая формула φ_i имеет один из следующих видов:

$$x_l + x_j = x_k,\quad x_j = x_k,\quad x_l x_j = x_k,\quad x_j = c$$

(здесь c — натуральное число), что для произвольного натурального числа n имеем $n \in A$ тогда и только тогда, когда формула $\Phi_A(n)$ истинна на множестве натуральных чисел.

По формуле $\Phi_A(x_1)$ построим формулу $\Phi_A^{(1)}(x_1)$ следующим образом:

$$\Phi_A^{(1)}(x_1) \rightleftharpoons (\exists x_2)\dots(\exists x_m)\left(\Psi_1\ \&\ \left(\mathop{\&}_{i=2}^{m} \mathbb{N}_a(x_i)\right)\right),$$

где Ψ_1 получается из Ψ заменой каждой подформулы φ_i вида $x_l + x_j = x_k$ на $x_l x_j = x_k$, вида $x_j = x_k$ — на $x_j = x_k$, вида $x_j = c$ — на $x_j = a^c$ и вида $x_l x_j = x_k$ — на $M(x_l, x_j, x_k)$.

Подходящим образом переименовав переменные в формуле $\Phi_A^{(1)}(x_1)$, можем считать, что в формулу $\Phi_A^{(1)}(x)$ входят лишь переменные $x, x_1, \ldots, x_n$.

При построении формулы $\Phi_A^{(1)}(x_1)$ по формуле $\Phi_A(x_1)$ мы заменяли каждую подформулу $x_l x_j = x_k$ на $M(x_l, x_j, x_k)$. В итоге конъюнкция $\mathop{\&}\limits_{p=1}^{m} x_{l_p} x_{j_p} = x_{k_p}$ заменилась на $\mathop{\&}\limits_{p=1}^{m} M(x_{l_p}, x_{j_p}, x_{k_p})$ и справедлива следующая эквивалентность:

$M_2 \models \mathop{\&}\limits_{p=1}^{m} M(x_{l_p}, x_{j_p}, x_{k_p})$ тогда и только тогда, когда найдутся такие натуральные числа s_p, t_p и r_p $(p = 1, \ldots, m)$, что $x_{l_p} = a^{s_p}$, $y_{j_p} = a^{t_p}$, $z_{k_p} = a^{r_p}$ и $s_p t_p = r_p$.

Выпишем в явном виде конъюнкцию $\mathop{\&}\limits_{p=1}^{m} M(x_{l_p}, x_{j_p}, x_{k_p})$ и приведём её к виду

$$\mathop{\&}_{p=1}^{m} (\exists y)(\exists z)\Big(\mathbb{N}_a(x_{l_p}) \;\&\; \mathbb{N}_a(x_{j_p}) \;\&\; \mathbb{N}_a(x_{k_p}) \;\&$$
$$\&\; (\exists v_p)(\exists u_p)(\exists h_p)\big(u_p(bx_{l_p}) = (bx_{l_p})u_p \;\&\; v_p b = b v_p \;\&$$
$$\&\; abh_p = h_p ab \;\&\; x_{j_p} \leqslant h_p \;\&\; v_p \leqslant h_p \;\&\; x_{k_p} \leqslant u_p \;\&\; v_p \leqslant u_p\big) \;\&$$
$$\&\; \big(y = x_{j_1} v_1 \ldots x_{j_m} v_m x_{k_1} v_1 \ldots x_{k_m} v_m \;\&\; z = h_1 \ldots h_m u_1 \ldots u_m \;\&\; |y| = |z|\big)\Big).$$

Последнюю формулу также будем обозначать через $\mathop{\&}\limits_{p=1}^{m} M(x_{l_p}, x_{j_p}, x_{k_p})$.

Покажем, что для произвольных слов A_p, B_p и C_p $(p = 1, \ldots, m)$ на свободном моноиде M_2 формула

$$\mathop{\&}_{p=1}^{m} M(A_p, B_p, C_p) \tag{4}$$

истинна тогда и только тогда, когда существуют такие натуральные числа s_p, t_p и r_p $(p = 1, \ldots, m)$, что $A_p = a^{s_p}$, $B_p = a^{t_p}$, $C = a^{r_p}$ и $r_p = s_p t_p$.

Если на свободном моноиде M_2 истинна формула (4), то существуют такие натуральные числа s_p, t_p и r_p $(p = 1, \ldots, m)$, что $A_p = a^{s_p}$, $B_p = a^{t_p}$, $C = a^{r_p}$. Остаётся показать, что $r_p = s_p t_p$.

Нетрудно понять, что существуют такие числа α_p и γ_p $(p = 1, \ldots, m)$, что $h_p = (ab)^{\alpha_p}$ и $v_p = b^{\gamma_p}$. Тогда неравенства $x_{j_p} \leqslant h_p \;\&\; v_p \leqslant h_p$ влекут неравенства $t_p \leqslant \alpha_p \;\&\; \gamma_p \leqslant \alpha_p$.

Аналогично существуют такие числа β_p $(p = 1, \ldots, m)$, что $u_p = (ba^{s_p})^{\beta_p}$. Тогда неравенства $x_{k_p} \leqslant u_p \;\&\; v_p \leqslant u_p$ влекут неравенства $r_p \leqslant s_p \beta_p \;\&\; \gamma_p \leqslant \beta_p$.

Равенство $|y| = |z|$ влечет равенство

$$t_1 + \gamma_1 + \ldots + t_m + \gamma_m + r_1 + \gamma_1 + \ldots + r_m + \gamma_m =$$
$$= \alpha_1 + \alpha_1 + \ldots + \alpha_m + \alpha_m + s_1\beta_1 + \beta_1 + \ldots + s_m\beta_m + \beta_m,$$

что вместе с предыдущими неравенствами

$$\mathop{\&}_{p=1}^{m} (t_p \leqslant \alpha_p \;\&\; \gamma_p \leqslant \alpha_p) \;\&\; \mathop{\&}_{p=1}^{m} (r_p \leqslant s_p\beta_p \;\&\; \gamma_p \leqslant \beta_p)$$

даёт систему равенств $\mathop{\&}\limits_{p=1}^{m} (t_p = \alpha_p \;\&\; \gamma_p = \alpha_p \;\&\; r_p = s_p\beta_p \;\&\; \beta_p = \gamma_p)$, из которых следует требуемое: $\mathop{\&}\limits_{p=1}^{m} r_p = s_p t_p$.

Обратное утверждение доказывается по той же схеме.

Приведём полученную формулу $\Phi_A^{(1)}(x)$ к виду

$$\mathop{\&}_{i=1}^{k} w_i(x_1,\dots,x_n,a,b) \leqslant u_i(x_1,\dots,x_n,a,b) \ \&\ |x_1| = |x_2|.$$

Полученную формулу обозначим через $\Phi_A^{(2)}(x)$. Тогда для произвольного натурального числа m справедлива следующая эквивалентность: $m \in A$ тогда и только тогда, когда формула $\Phi_A^{(2)}(a^m)$ истинна на моноиде M_2.

Для завершения доказательства теоремы достаточно взять в качестве A рекурсивно перечислимое, но нерекурсивное множество натуральных чисел. ■

Заключение

В работе построено такое однопараметрическое семейство уравнений с ограничениями на решения в свободном моноиде M_2 (свободной полугруппе S_2)

$$w(x,x_1,\dots,x_n,a,b) = v(x,x_1,\dots,x_n,a,b) \ \&\ \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \ \&\ |x_1|_b = |x_2|_b$$

с неизвестными $x_1,\dots,x_n$, константами a и b и параметром x, где A — некоторое подмножество множества $\{\{t,s\} : 1 \leqslant t,s \leqslant n\}$, что невозможен алгоритм, позволяющий для произвольного натурального числа m определить, имеет ли решение следующее уравнение с ограничениями на решения:

$$w(a^m,x_1,\dots,x_n,a,b) = v(a^m,x_1,\dots,x_n,a,b) \ \&\ \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \ \&\ |x_1|_b = |x_2|_b.$$

Построено такое однопараметрическое семейство уравнений с ограничениями на решения в свободном моноиде M_2

$$w(x,x_1,\dots,x_n,a,b) = v(x,x_1,\dots,x_n,a,b) \ \&\ \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \ \&\ x_1 \in L_1$$

с неизвестными $x_1,\dots,x_n$, константами a и b и параметром x, где A — некоторое подмножество множества $\{\{t,s\} : 1 \leqslant t,s \leqslant n\}$, что невозможен алгоритм, позволяющий для произвольного натурального числа m определить, имеет ли решение следующее уравнение с ограничениями на решения:

$$w(a^m,x_1,\dots,x_n,a,b) = v(a^m,x_1,\dots,x_n,a,b) \ \&\ \mathop{\&}_{\{i,j\}\in A} |x_i| = |x_j| \ \&\ x_1 \in L_1.$$

Для введённого V. Diekert отношения частичного порядка доказано, что невозможно построить алгоритм, позволяющий для произвольной системы неравенств с одним ограничением вида

$$\mathop{\&}_{i=1}^{k} w_i(x_1,\dots,x_n,a,b) \leqslant u_i(x_1,\dots,x_n,a,b) \ \&\ |x_1| = |x_2|$$

определить, имеет ли она решение в свободном моноиде M_2 (свободной полугруппе S_2).

ЛИТЕРАТУРА

1. *Дурнев В. Г., Зеткина А. И.* Об уравнениях в свободных моноидах с ограничениями на решения // Актуальные проблемы прикладной математики, информатики и механики: сб. трудов Междунар. науч. конф. Воронеж, 13–15 декабря 2021 г. Воронеж: Изд-во «Научно-исследовательские публикации», 2022. С. 1571–1575.

2. *Дурнев В. Г., Зеткина А. И.* Алгоритмические проблемы для уравнений в свободных группах и полугруппах с ограничениями на решения // Всерос. науч. конф. «Математические основы информатики и информационно-коммуникационных систем». Тверь, 3–8 декабря 2021 г. Тверь: ТвГУ, 2021. С. 25–41.
3. *Перязев Н. А.* Позитивные теории свободных моноидов // Алгебра и логика. 1993. Т. 32. № 2. С. 148–159.
4. *Косовский Н. К.* Некоторые свойства решений уравнений в свободной полугруппе // Записки науч. семинаров Ленингр. отд. Матем. ин-та им. В. А. Стеклова АН СССР. 1972. Т. 32. С. 21–28.
5. *Дурнев В. Г.* Неразрешимость позитивной $\forall\exists^3$-теории свободной полугруппы // Сиб. матем. журн. 1995. Т. 36. № 5. С. 1067–1080.
6. *Хмелевский Ю. И.* Уравнения в свободной полугруппе. М.: Наука, 1971. 218 с.
7. *Матиясевич Ю. В.* Связь систем уравнений в словах и длинах с 10-й проблемой Гильберта // Исследования по конструктивной математике и математической логике. Записки науч. семинаров Ленингр. отд. Матем. ин-та им. В. А. Стеклова АН СССР. 1968. Т. 8. С. 132–143.
8. *Косовский Н. К.* О множествах, представимых в виде решений уравнений в словах и длинах // Вторая Всесоюз. конф. по матем. логике. Тезисы кратких сообщений. М., 1972. С. 23.
9. *Косовский Н. К.* О решении систем, состоящих одновременно из уравнений в словах и неравенств в длинах слов // Записки науч. семинаров Ленингр. отд. Матем. ин-та им. В. А. Стеклова АН СССР. 1973. Т. 33. С. 24–29.
10. *Дурнев В. Г.* Об уравнениях на свободных полугруппах и группах // Матем. заметки. 1974. Т. 16. № 5. С. 717–724.
11. *Buchi J. R. and Senger S.* Definability in the existential theory of concatenation // Z. Math. Log. und Grundl. Math. 1988. V. 34. No. 4. P. 337–342.
12. *Маканин Г. С.* Проблема разрешимости уравнений в свободной полугруппе // ДАН СССР. 1977. Т. 233. № 2. С. 287–290.
13. *Маканин Г. С.* Проблема разрешимости уравнений в свободной полугруппе // Матем. сб. 1977. Т. 103 (145). № 2 (6). С. 147–236.
14. *Маканин Г. С.* Уравнения в свободных группах // Изв. АН СССР. Сер. матем. 1982. Т. 46. № 6. С. 1199–1274.
15. *Дурнев В. Г.* Позитивная теория свободной полугруппы // ДАН СССР. 1973. Т. 211. № 4. С. 772–774.
16. *Дурнев В. Г.* О позитивных формулах на свободных полугруппах // Сиб. матем. журн. 1974. Т. 25. № 5. С. 1131–1137.
17. *Важенин Ю. М., Розенблат Б. В.* Разрешимость позитивной теории свободной счетнопорожденной полугруппы // Матем. сб. 1981. Т. 116. № 1. С. 120–127.
18. *Маканин Г. С.* Разрешимость универсальной и позитивной теорий свободной группы // Изв. АН СССР. Сер. матем. 1984. № 4. С. 735–749.
19. *Мерзляков Ю. И.* Позитивные формулы на свободных группах // Алгебра и логика. 1966. Т. 5. № 4. С. 25–42.
20. Коуровская тетрадь. 11-е изд., доп. / ред. В. Д. Мазуров и др. Новосибирск, 1990. 126 с.
21. *Малхасян А. Ш.* О разрешимости в подгруппах уравнений в свободной группе // Прикладная математика. 1986. Вып. 2. С. 42–47.
22. *Schulz K. U.* Makanin's algorithm for word equations — two improvements and a generalization // LNCS. 1990. V. 572. P. 85–150.

23. *Diekert V., Gutierrez C., and Hagenah C.* The existential theory of equations with rational constraints in free groups is PSPACE-complete // LNCS. 2001. V. 2010. P. 170–182.
24. *Diekert V., Gutierrez C., and Hagenah C.* The existential theory of equations with rational constraints in free groups is PSPACE-complete // Information and Computation. 2005. V. 202. P. 105–140.
25. *Матиясевич Ю. В.* Диофантовость перечислимых множеств // ДАН СССР. 1970. Т. 130. № 3. С. 495–498.
26. *Robinson J.* Existential definability in arithmetic // Trans. Amer. Math. Soc. 1952. V. 72. P. 437–439.

REFERENCES

1. *Durnev V. G. and Zetkina A. I.* Ob uravneniyakh v svobodnykh monoidakh s ogranicheniyami na resheniya [On equations in free monoids with restrictions on solutions]. Aktual'nye Problemy Prikladnoy Matematiki, Informatiki i Mekhaniki, Voronezh, 2022, pp. 1571–1575. (in Russian)
2. *Durnev V. G. and Zetkina A. I.* Algoritmicheskie problemy dlya uravneniy v svobodnykh gruppakh i polugruppakh s ogranicheniyami na resheniya [Algorithmic problems for equations in free groups and semigroups with constraints on solutions]. Vseros. nauch. konf. "Matematicheskie Osnovy Informatiki i Informatsionno-Kommunikatsionnykh Sistem", Tver, TvSU, 2021, pp. 25–41. (in Russian)
3. *Peryazev N. A.* Pozitivnye teorii svobodnykh monoidov [Positive theories of free monoids]. Algebra i Logika, 1993, vol. 32, no. 2, pp. 148–159. (in Russian)
4. *Kosovskiy N. K.* Nekotorye svoystva resheniy uravneniy v svobodnoy polugruppe [Some properties of solutions of equations in a free semigroup]. Zapiski Nauch. Seminarov Leningr. Otd. Mat. In-ta im. V. A. Steklova, 1972, vol. 32, pp. 21–28. (in Russian)
5. *Durnev V. G.* Nerazreshimost' pozitivnoy $\forall\exists^3$-teorii svobodnoy polugruppy [Undecidability of the positive $\forall\exists^3$-theory of a free semigroup]. Sib. Matem. Zhurn., 1995, vol. 36, no. 5, pp. 1067–1080. (in Russian)
6. *Khmelevskiy Yu. I.* Uravneniya v svobodnoy polugruppe [Equations in a free semigroup.]. Moscow, Nauka Publ., 1971. 218 p. (in Russian)
7. *Matiyasevich Yu. V.* Svyaz' sistem uravneniy v slovakh i dlinakh s 10-oy problemoy Gil'berta [Connection of systems of equations in words and lengths with Hilbert's 10th problem]. Zapiski Nauch. Seminarov Leningr. Otd. Mat. In-ta im. V. A. Steklova, 1968, vol. 8, pp. 132–143. (in Russian)
8. *Kosovskiy N. K.* O mnozhestvakh, predstavimykh v vide resheniy uravneniy v slovakh i dlinakh [On sets representable as solutions of equations in words and lengths]. Vtoraya Vsesoyuz. Konf. po Matem. Logike. Moscow, 1972, p. 23. (in Russian)
9. *Kosovskiy N. K.* O reshenii sistem, sostoyashchikh odnovremenno iz uravneniy v slovakh i neravenstv v dlinakh slov [On the solution of systems consisting simultaneously of equations in words and inequalities in word lengths]. Zapiski Nauch. Seminarov Leningr. Otd. Mat. In-ta im. V. A. Steklova, 1973, vol. 33, pp. 24–29. (in Russian)
10. *Durnev V. G.* Ob uravneniyakh na svobodnykh polugruppakh i gruppakh [On equations on free semigroups and groups]. Matem. Zametki, 1974, vol. 16, no. 5, pp. 717–724. (in Russian)
11. *Buchi J. R. and Senger S.* Definability in the existential theory of concatenation. Z. Math. Log. und Grundl. Math., 1988, vol. 34, no. 4, pp. 337–342.
12. *Makanin G. S.* Problema razreshimosti uravneniy v svobodnoy polugruppe [The problem of solvability of equations in a free semigroup]. DAN USSR, 1977, vol. 233, no. 2, pp. 287–290. (in Russian)

13. *Makanin G. S.* Problema razreshimosti uravneniy v svobodnoy polugruppe [The problem of solvability of equations in a free semigroup]. Matem. Sbor., 1977, vol. 103 (145), no. 2 (6), pp. 147–236. (in Russian)
14. *Makanin G. S.* Uravneniya v svobodnykh gruppakh [Equations in free groups]. Izv. AN USSR. Ser. Matem., 1982, vol. 46, no. 6, pp. 1199–1274. (in Russian)
15. *Durnev V. G.* Pozitivnaya teoriya svobodnoy polugruppy [Positive theory of a free semigroup]. DAN USSR, 1973, vol. 211, no. 4, pp. 772–774. (in Russian)
16. *Durnev V. G.* O pozitivnykh formulakh na svobodnykh polugruppakh [On positive formulas on free semigroups]. Sib. Matem. Zhurn., 1974, vol. 25, no. 5, pp. 1131–1137. (in Russian)
17. *Vazhenin Yu. M. and Rozenblat B. V.* Razreshimost' pozitivnoy teorii svobodnoy schetnoporozhdennoy polugruppy [Decidability of the positive theory of a free countably generated semigroup]. Matem. Sbornik, 1981, vol. 116, no. 1, pp. 120–127. (in Russian)
18. *Makanin G. S.* Razreshimost' universal'noy i pozitivnoy teoriy svobodnoy gruppy [Decidability of the universal and positive theories of a free group]. Izv. AN USSR. Ser. Matem., 1984, no. 4, pp. 735–749. (in Russian)
19. *Merzlyakov Yu. I.* Pozitivnye formuly na svobodnykh gruppakh [Positive formulas on free groups]. Algebra i Logika, 1966, vol. 5, no. 4, pp. 25–42. (in Russian)
20. Kourovskaya tetrad' [Kourovka Notebook]. V. D. Mazurov et al. (eds.). Novosibirsk, 1990. 126 p. (in Russian)
21. *Malkhasyan A. Sh.* O razreshimosti v podgruppakh uravneniy v svobodnoy gruppe [On the solvability in subgroups of equations in a free group]. Prikladnaya Matematika, 1986, iss. 2, pp. 42–47. (in Russian)
22. *Schulz K. U.* Makanin's algorithm for word equations — two improvements and a generalization. LNCS, 1990, vol. 572, pp. 85–150.
23. *Diekert V., Gutierrez C., and Hagenah C.* The existential theory of equations with rational constraints in free groups is PSPACE-complete. LNCS, 2001, vol. 2010, pp. 170–182.
24. *Diekert V., Gutierrez C., and Hagenah C.* The existential theory of equations with rational constraints in free groups is PSPACE-complete. Information and Computation, 2005, vol. 202, pp. 105–140.
25. *Matiyasevich Yu. V.* Diofantovost' perechislimykh mnozhestv [Diophantine property of enumerable sets]. DAN USSR, 1970, vol. 130, no. 3, pp. 495–498. (in Russian)
26. *Robinson J.* Existential definability in arithmetic. Trans. Amer. Math. Soc., 1952, vol. 72, pp. 437–439.

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.

УДК 519.719.2 DOI 10.17223/20710410/59/3

ТРОИЧНАЯ ЛЕММА О РАЗВЛЕТВЛЕНИИ И ЕЁ ПРИЛОЖЕНИЕ К АНАЛИЗУ СТОЙКОСТИ ОДНОЙ КОДОВОЙ СХЕМЫ ПОДПИСИ

К. Д. Царегородцев

АО «НПК "Криптонит"», г. Москва, Россия

E-mail: k.tsaregorodtsev@kryptonite.ru

Работа посвящена обобщению леммы о развлетвлении на случай, когда хеш-функция возвращает набор тритов, и приложению леммы к альтернативному доказательству стойкости в модели SUF-CMA одной кодовой схемы подписи, основанной на протоколе идентификации Штерна.

Ключевые слова: *лемма о развлетвлении, электронная подпись, доказуемая стойкость.*

TERNARY FORKING LEMMA AND ITS APPLICATION TO THE ANALYSIS OF ONE CODE-BASED SIGNATURE

K. D. Tsaregorodtsev

JSC "NPK "Kryptonite", Moscow, Russia

The paper is devoted to the generalization of the so-called "forking lemma" to the case when the hash function returns a tuple of trits (trit is a variable that can take one out of the three values 0, 1, 2). It can be stated as follows. Let $\mathcal{A}(par, b)$ be an algorithm that, when run on randomly chosen $par \leftarrow^{\mathrm{R}}$ Params and $b \leftarrow^{\mathrm{R}} \{0,1,2\}^{\delta}$, successfully stops and returns the correct answer x with probability ϵ. Then there exists an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and returns a triple (x_1, x_2, x_3), where $x_i \leftarrow \mathcal{A}(par, b^i)$, $i = 1, 2, 3$, with the additional condition that there exists a position j such that $\{b_j^1, b_j^2, b_j^3\} = \{0, 1, 2\}$ (i.e., all trits in this position are different); the success probability of $\mathcal{B}$ can be bounded from above as follows: $p_{\mathcal{B}} \geqslant \varepsilon^3 - \varepsilon\,(17/27)^{\delta/2}$, and the running time of $\mathcal{B}$ does not exceed $4\,t_{\mathcal{A}}$, where $t_{\mathcal{A}}$ is the time complexity of $\mathcal{A}$. The lemma is then applied to the analysis of code-based signature based on Stern identification protocol in the (standard) SUF-CMA model (with the outer hash function modelled as a programmable random oracle). First, we show that the SUF-CMA model can be reduced to the NMA model (in which the adversary makes no sign queries, only random oracle queries), thanks to the zero-knowledge property of the original Stern identification scheme and the programmability of an oracle. We then show that in the NMA model we can restrict attention only to the case, where the adversary makes a single random oracle query. The b value from the forking lemma acts as the random oracle's answer to the adversarial query. Using the lemma, we are able to fork the process of forging a signature. Having three valid signatures for the same message, we can to extract a secret key (or to find a collision of an inner hash function). Hence, we can bound from above the probability of successful forgery in terms of the probability of successful execution of collision finding and syndrome decoding algorithms.

Keywords: *forking lemma, digital signature, provable security.*

Введение

Лемма о развлетвлении (forking lemma) [1] часто используется для доказательства стойкости схем подписей, полученных из схем идентификации (сигма-протоколов) [2] на основе преобразования Фиата — Шамира [3]. Основой для её применения является следующее соображение. Базовая схема идентификации, как правило, обладает свойством корректности (soundness) [4]. Неформально, данное свойство гласит, что если противник может с большой вероятностью корректно пройти аутентификацию на особым образом связанных запросах, то фактически он должен знать секретное значение (более формально: можно запустить противника дважды, получить два ответа и извлечь из них секретное значение). При этом в стандартной версии леммы [1, 5] противнику, по сути, нужно успешно пройти аутентификацию на любых двух неравных запросах. Этого оказывается достаточно для протоколов, в которых вероятность пройти раунд протокола без знания секрета составляет $1/2$ (см. также свойство специальной корректности (special soundness), например, в [2]).

Для рассматриваемой в работе [6] кодовой схемы подписи основой является протокол идентификации Штерна [7], вероятность пройти раунд протокола без знания секрета в котором равна $2/3$. В этих условиях для извлечения секрета необходимо, чтобы противник трижды успешно прошел аутентификацию, причем требование попарного неравенства запросов необходимо усилить следующим образом: должно найтись такое число j, что все триты в j-й координате запросов попарно различны (трит — величина, которая может принимать одно из трёх значений 0, 1, 2). В описанных условиях стандартная лемма о развлетвлении неприменима. В настоящей работе рассматривается обобщение леммы о развлетвлении на троичный случай.

В п. 1 приводятся краткое описание рассматриваемой схемы подписи, модели SUF-CMA и NMA для изучения стойкости схем подписи, а также используемые обозначения. Пункт 2 посвящён формулировке и доказательству троичной леммы о развлетвлении. В п. 3 доказанная лемма применяется к схеме подписи для доказательства стойкости в модели SUF-CMA; показано, что оценка из этой работы асимптотически (при стремлении числа раундов $\delta \to \infty$) превосходит оценку, полученную в работе [6], но для конкретных значимых на практике параметров оказывается более слабой.

1. Определения и используемые обозначения

1.1. Используемые обозначения

Определение 1. Схемой подписи будем называть тройку вероятностных алгоритмов [8, разд. 13]:

1) Алгоритм генерации пары ключей $(\mathrm{pk}, \mathrm{sk}) \leftarrow^{\mathrm{R}} \mathrm{KGen}()$.
2) Алгоритм генерации подписи для сообщения m:

$$\tau \leftarrow^{\mathrm{R}} \mathrm{Sign}_{\mathrm{sk}}(m).$$

3) Алгоритм проверки подписи τ для сообщения m:

$$res \leftarrow \mathrm{Verify}_{\mathrm{pk}}(m, \tau) \in \{0, 1\}.$$

К схеме подписи предъявляется (синтаксическое) требование корректности: для любого сообщения m и любой пары ключей $(\mathrm{pk}, \mathrm{sk}) \leftarrow^{\mathrm{R}} \mathrm{KGen}()$ выполняется равенство

$$\mathrm{Verify}_{\mathrm{pk}}(m, \mathrm{Sign}_{\mathrm{sk}}(m)) = 1.$$

Определение 2. Код (линейный двоичный код) — линейное k-мерное подпространство $\mathcal{C}$ векторного пространства $\mathbb{F}_2^n$.

Определение 3. Проверочная матрица H кода $\mathcal{C}$ — матрица полного ранга размера $(n-k)\times n$, такая, что $Hx=0$ для любого $x\in\mathcal{C}$.

Определение 4. Задача синдромного декодирования [9] формулируется следующим образом. Противнику даются параметры (H, y, ω):

— $H\in \mathrm{Mat}_{n-k,n}(\mathbb{F}_2)$: проверочная матрица двоичного кода;
— $y\in\{0,1\}^{n-k}$: ненулевой синдром;
— $\omega>0$: число, вес вектора ошибок.

Его задачей является нахождение вектора $s\in\{0,1\}^n$, такого, что $\mathrm{wt}(s)=\omega$ и $Hs=y$.

Введём величину $\mathrm{InSec}^{\mathrm{SD}}(t)$ — максимальную вероятность успешного решения противником задачи SD (при заданных n, k и ω), где максимум берется по всем противникам, временные ресурсы которых ограничены t тактами вычислений.

Определение 5. Задача поиска коллизии формулируется следующим образом [8, разд. 6.1]. В начале эксперимента выбирается случайная функция $h\leftarrow^{\mathrm{R}}\mathrm{Hash}$ из некоторого семейства хеш-функций $\mathrm{Hash}=\{h\}$, $h\colon\{0,1\}^*\to\{0,1\}^\ell$, её описание предоставляется противнику (можно предполагать, что семейство хеш-функций параметризовано некоторым ключом-строкой s и противнику предоставляется конкретный индекс функции). Задачей противника является нахождение векторов $x', x''\in\{0,1\}^*$, $x'\neq x''$, со свойством $h(x')=h(x'')$.

Введём величину $\mathrm{InSec}^{\mathrm{Coll}}(t)$ — максимальную вероятность успешного решения противником задачи Coll (для заданного семейства Hash), где максимум берется по всем противникам, временные ресурсы которых ограничены t тактами вычислений. Во временные ресурсы также часто включают размер программы противника.

Будем использовать следующие обозначения:

δ — число раундов в схеме подписи;
H — проверочная матрица кода, используемого в схеме подписи;
h — «внутренняя» хеш-функция в схеме подписи;
f — «внешняя» хеш-функция в схеме подписи;
$\mathbb{B}$ — множество $\{0,1,2\}^\delta$; хеш-функция f принимает значения из $\mathbb{B}$;
n — длина используемого в схеме подписи кода;
S_n — группа подстановок на множестве $\{1,\ldots,n\}$;
$\sigma(x)$ — вектор $(x_{\sigma(1)},\ldots,x_{\sigma(n)})$, $\sigma\in S_n$;
$\mathbb{I}[A]$ — индикатор события A;
$\langle U\mid V\rangle$ — скалярное произведение векторов U и V над полем вещественных чисел $\mathbb{R}$;
$x\leftarrow^{\mathrm{R}}X$ — выбор случайного равновероятного элемента x из конечного множества X; если X — вероятностный алгоритм, то присвоить переменной x случайный выход алгоритма X;
$x\leftarrow y$ — присвоить переменной x значение переменной y;
Λ — неинициализированное значение массива (пустая строка).

1.2. Схема подписи на основе схемы идентификации Штерна

В работе [6] описана схемы подписи на основе схемы идентификации Штерна. Приведём её краткое описание и некоторые свойства.

Алгоритм генерации ключей. Дана проверочная матрица H некоторого кода. Алгоритм KGen() задаётся следующим образом: необходимо выбрать случайный вектор $s\in\{0,1\}^n$, такой, что $\mathrm{wt}(s)=\omega$, затем вычислить вектор $y\in\{0,1\}^{n-k}$ как $y=Hs$.

Вектор y является открытым ключом проверки подписи, вектор s — секретным ключом подписи.

Алгоритм подписи. Даны: ключ подписи $\mathrm{sk} = s$, подписываемое сообщение m. Для вычисления подписи необходимо сделать следующие шаги:

1) Для всех раундов $i = 1, \ldots, \delta$:
   - выбрать $u_i \leftarrow^{\mathrm{R}} \{0,1\}^n$, $\sigma_i \leftarrow^{\mathrm{R}} S_n$;
   - вычислить $c_{i0} \leftarrow h(\sigma_i \| Hu_i)$, $c_{i1} \leftarrow h(\sigma_i(u_i))$, $c_{i2} \leftarrow h(\sigma_i(u_i \oplus s))$, $C_i = (c_{i0} \| c_{i1} \| c_{i2})$.
2) Вычислить *challenge*: $b = f(m \| C_1 \| \ldots \| C_\delta) \in \mathbb{B}$.
3) Вычислить *response*: для всех раундов $i = 1, \ldots, \delta$:
   - если $b_i = 0$, положить $R_i \leftarrow \sigma_i \| u_i$;
   - если $b_i = 1$, положить $R_i \leftarrow \sigma_i \| (u_i \oplus s)$;
   - если $b_i = 2$, положить $R_i \leftarrow \sigma_i(u_i) \| \sigma_i(s)$.
4) Сформировать результат-подпись:

$$\zeta = (C, R) = (C_1 \| \ldots \| C_\delta \| R_1 \| \ldots \| R_\delta).$$

Алгоритм проверки подписи. Даны: открытый ключ y, сообщение m, подпись $\zeta = (C, R) = (C_1 \| \ldots \| C_\delta \| R_1 \| \ldots \| R_\delta)$. Для проверки подписи необходимо сделать следующие шаги:

1) Вычислить $b = f(m \| C_1 \| \ldots \| C_\delta)$.
2) Для всех раундов $i = 1, \ldots, \delta$ проверить:
   - если $b_i = 0$, то $c_{i0} = h\left(R_{i0} \| HR_{i1}\right)$, $c_{i1} = h(R_{i0}(R_{i1}))$;
   - если $b_i = 1$, то $c_{i0} = h\left(R_{i0} \| HR_{i1} \oplus y\right)$, $c_{i2} = h(R_{i0}(R_{i1}))$;
   - если $b_i = 2$, то $c_{i1} = h(R_{i0})$, $c_{i2} = h\left(R_{i0} \oplus R_{i1}\right)$, $\mathrm{wt}(R_{i1}) = \omega$.
3) Если все проверки успешны, то вернуть 1 (иначе 0).

Свойства схемы подписи. Обозначим некоторые свойства представленной схемы подписи:

— Симуляция раунда подписи без знания секрета [6, 7]: существует алгоритм Sim, который по заданному $b \in \{0,1,2\}$ моделирует такие случайные величины (C'_i, R'_i), что их распределение совпадает с величинами (C_i, R_i), построенными в соответствии с «честным» алгоритмом подписи. Это свойство следует из свойства нулевого разглашения протокола идентификации Штерна.

— Подделка подписи без знания секрета: противник может подделать подпись с вероятностью не менее чем $(2/3)^\delta$.

— Извлечение секрета (см. работу [6], а также доказательство теоремы 3): если противник может построить три подделки подписи (C^i, R^i), $i = 1, 2, 3$, такие, что соответствующие им векторы тритов b^i имеют позицию j, в которой все три вектора попарно различны (т. е. $\{b^1_j, b^2_j, b^3_j\} = \{0,1,2\}$), то существует алгоритм Extract, который с вероятностью, стремящейся к 1 при стремлении числа раундов δ к бесконечности, извлекает секретный ключ s схемы подписи либо находит коллизию внутренней хеш-функции h.

Замечание 1. Опишем подробнее свойство симуляции. Для симуляции подписи генерируется общий для всех шагов i, $1 \leqslant i \leqslant \delta$, случайный равновероятный вектор s' из множества $\mathbb{F}_2^n$ с дополнительным ограничением на вес. На каждом из шагов $i = 1, \ldots, \delta$ необходимо:

1) сгенерировать $u_i \leftarrow^{\mathrm{R}} \mathbb{F}_2^n$, $\sigma_i \leftarrow^{\mathrm{R}} S_n$;

2) в зависимости от b_i сгенерировать следующие величины:
   — если $b_i = 0$, положить

$$c_{i0} \leftarrow h(\sigma_i \| Hu_i), \quad c_{i1} \leftarrow h(\sigma_i(u_i)), \quad c_{i2} \leftarrow h(\sigma_i(u_i \oplus s')), \quad R_i \leftarrow \sigma_i \| u_i;$$

   — если $b_i = 1$, положить

$$c_{i0} = h(\sigma_i \| Hu_i \oplus y), c_{i1} = h(\sigma_i(u_i) \oplus s'), c_{i2} = h(\sigma_i(u_i)), \quad R_i \leftarrow \sigma_i \| u_i;$$

   — если $b_i = 2$, положить

$$c_{i0} = h(\sigma_i \| H(u_i \oplus s')), c_{i1} = h(\sigma_i(u_i)), c_{i2} = h(\sigma_i(u_i \oplus s')), \quad R_i \leftarrow \sigma_i(u_i) \| \sigma_i(s').$$

Заметим также, что при генерации c_{ij} в каждом из C_i есть подстрока вида $h(X)$, где X выбирается случайно равновероятно из множества $\{0,1\}^n$.

1.3. Модели SUF-CMA и NMA

Рассмотрим следующие две стандартные модели, используемые для изучения стойкости схем подписи. Противнику предоставляется доступ к оракулу подписи Sign, его задачей является подделка подписи для нового (ранее не запрашиваемого) сообщения. В случае модели со случайным оракулом (ROM, Random Oracle Model, см. [10, 11] и [8, разд. 6.5]) противнику также даётся доступ к случайному оракулу f (случайной функции, моделирующей поведение реальной хеш-функции). Приведём псевдокод эксперимента, используемого в моделях SUF-CMA и NMA:

$\mathrm{Exp}^{\text{SUF-CMA}}(\mathcal{A})$	$\mathrm{Sign}(m)$
$(\mathrm{sk}, \mathrm{pk}) \leftarrow^{\mathrm{R}} \mathrm{KGen}()$	$\tau \leftarrow^{\mathrm{R}} \mathrm{Sign}_{\mathrm{sk}}(m)$
$sent = \varnothing$	$sent \leftarrow sent \cup \{(m, \tau)\}$
$(m, \tau) \leftarrow^{\mathrm{R}} \mathcal{A}^{\mathrm{Sign},f}(\mathrm{pk})$	**return** τ
if $((m, \tau) \notin sent)$	
return $\mathrm{Verify}_{\mathrm{pk}}(m, \tau)$	
else	
return 0	
fi	

Определение 6. Уровнем нестойкости схемы подписи в модели SUF-CMA (Strong UnForgeability under Chosen Message Attack) со случайным оракулом будем называть число

$$\mathrm{InSec}^{\text{SUF-CMA}}(t, q_f, q_s) = \max_{\mathcal{A} \in A(t, q_f, q_s)} \mathsf{P}[\mathrm{Exp}^{\text{SUF-CMA}}(\mathcal{A}) \to 1],$$

где за $A(t, q_f, q_s)$ обозначено множество алгоритмов, работающих не более t тактов и делающих не более q_f запросов к случайному оракулу и q_s запросов к оракулу подписи.

Определение 7. Нестойкостью схемы подписи в модели NMA (No Message Attack) со случайным оракулом будем называть следующее число:

$$\mathrm{InSec}^{\mathrm{NMA}}(t, q_f) = \mathrm{InSec}^{\text{SUF-CMA}}(t, q_f, 0).$$

Другими словами, противник в модели NMA не делает ни одного запроса к оракулу подписи, и его задачей является подделка подписи для любого сообщения.

2. Троичная лемма о развлетвлении

Сформулируем и докажем основной результат работы.

Лемма 1. Пусть $\mathcal{A}(par, b)$ — алгоритм, который на случайно выбранных входах $par \leftarrow^{\mathrm{R}} \mathrm{Params}$ и $b \leftarrow^{\mathrm{R}} \mathbb{B}$ успешно завершает работу с некоторым выходом x с вероятностью ε. Тогда мы можем построить алгоритм $\mathcal{B}$, использующий алгоритм $\mathcal{A}$ как подпроцедуру, который возвращает тройку (x_1, x_2, x_3), где x_i есть результат работы $\mathcal{A}(par, b^i)$, с дополнительным условием, что в (b^1, b^2, b^3) найдётся такой номер i, для которого все триты в этой позиции различны ($\{b_i^1, b_i^2, b_i^3\} = \{0, 1, 2\}$), причём вероятность успешной работы алгоритма $\mathcal{B}$ оценивается снизу как

$$p_{\mathcal{B}} \geqslant \varepsilon^3 - \varepsilon\,(17/27)^{\delta/2}\,,$$

время работы алгоритма $\mathcal{B}$ оценивается сверху как $4\,t_{\mathcal{A}}$, где $t_{\mathcal{A}}$ — время работы алгоритма $\mathcal{A}$.

Доказательство. Построим алгоритм $\mathcal{B}$ следующим образом:

— В начале эксперимента алгоритм $\mathcal{B}$ генерирует параметры $par \leftarrow^{\mathrm{R}} \mathrm{Params}$ и три случайных независимых равновероятных элемента $b^i \leftarrow^{\mathrm{R}} \mathbb{B}$, $i = 1, 2, 3$.

— Алгоритм $\mathcal{B}$ трижды запускает алгоритм $\mathcal{A}$ на входах (par, b^i) и получает результаты работы x_i, $i = 1, 2, 3$.

— Если все три запуска завершились успешно, а также найдётся такой номер i, для которого все триты b_i^1, b_i^2, b_i^3 различны, то алгоритм $\mathcal{B}$ возвращает тройку (x_1, x_2, x_3) в качестве ответа.

Алгоритм $\mathcal{B}$ трижды запускает алгоритм $\mathcal{A}$ и генерирует параметры par, длина которых не может превышать $t_{\mathcal{A}}$, что даёт оценку сверху на время работы $t_{\mathcal{B}} \leqslant 4\,t_{\mathcal{A}}$.

Найдём вероятность успешного завершения работы алгоритма $\mathcal{B}$. Введём события A — все три запуска алгоритма $\mathcal{A}$ успешны; B — в векторах тритов (b^1, b^2, b^3) найдётся позиция i, в которой все триты различны (в таком случае событие $\overline{B}$ заключается в том, что для каждой позиции i хотя бы два из трёх тритов совпадают). В таком случае вероятность $p_{\mathcal{B}}$ можно оценить следующим образом:

$$p_{\mathcal{B}} = \mathsf{P}[A \cap B] = \mathsf{P}[A] - \mathsf{P}[A \cap \overline{B}].$$

Далее мы введём случайную величину X, выразим вероятность каждого из событий A и $A \cap \overline{B}$ через математические ожидания функций от X и применим к полученной оценке неравенство Йенсена. Определим индикатор $\mathbb{I}(par, b)$ и случайную величину X на пространстве Params следующим образом:

$$\mathbb{I}(par, b) = \mathbb{I}[\mathcal{A}(par, b) \text{ завершается успешно}],$$
$$X(par) = \mathsf{P}[b \in \mathbb{B}\colon \mathcal{A}(par, b) \text{ завершается успешно}].$$

В таком случае имеем (по определению математического ожидания и величины ε как вероятности успешного завершения алгоритма $\mathcal{A}$):

$$\mathbb{E}[X] = \sum_{par} \mathsf{P}[par] \cdot X(par) = \sum_{par,b} \mathsf{P}[par] \cdot \mathsf{P}[b] \cdot \mathbb{I}(par, b) = \varepsilon.$$

Заметим, что вероятности событий A и $A \cap \overline{B}$ выражаются через введённые величины следующим образом:

$$\mathsf{P}[A] = \sum_{par} \sum_{(b^1,b^2,b^3)\in\mathbb{B}^3} \mathsf{P}[par] \cdot \mathsf{P}[b^1,b^2,b^3] \cdot \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2)\,\mathbb{I}(par,b^3) =$$

$$= \sum_{par} \mathsf{P}[par] \left(\sum_{b\in\mathbb{B}} \mathsf{P}[b] \cdot \mathbb{I}(par,b) \right)^3 = \mathbb{E}[X^3];$$

$$\mathsf{P}[A \cap \overline{B}] = \sum_{par} \sum_{(b^1,b^2,b^3)\in\mathbb{B}'} \mathsf{P}[par] \cdot \mathsf{P}[b^1,b^2,b^3] \cdot \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2)\,\mathbb{I}(par,b^3) =$$

$$= \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|^3} \sum_{b^1,b^2\in\mathbb{B}^2} \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2)\,S(b^1,b^2).$$

Здесь $\mathbb{B}'$ — множество таких троек $(b^1,b^2,b^3) \in \mathbb{B}^3$, что для каждой позиции принимается не более двух значений трита из трёх возможных (в частности, $|\mathbb{B}'| = 21^\delta$), а $S(b^1,b^2)$ — величина, равная количеству таких векторов b^3, что для тройки (b^1,b^2,b^3) в каждой позиции $i = 1,\ldots,\delta$ встречаются не более двух тритов из трёх возможных (группировка суммы по (b^1,b^2)).

Оценим сверху вероятность $\mathsf{P}[A\cap\overline{B}]$ с помощью неравенства Коши — Буняковского. Для этого введём векторы $U(b^1,b^2) = \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2) \in \{0,1\}$ и $V(b^1,b^2) = S(b^1,b^2)$, а также скалярное произведение векторов $\langle U \mid V\rangle = \sum\limits_{b^1,b^2\in\mathbb{B}^2} \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2)\,S(b^1,b^2)$. Тогда

$$\sum_{b^1,b^2\in\mathbb{B}^2} \mathbb{I}(par,b^1)\,\mathbb{I}(par,b^2)\,S(b^1,b^2) = \langle U \mid V\rangle \leqslant \sqrt{\langle U \mid U\rangle} \cdot \sqrt{\langle V \mid V\rangle}.$$

Рассмотрим отдельно каждое из скалярных произведений. Для выражения $\langle U \mid U\rangle$ выполнено следующее (используем то, что $U(b^1,b^2) \in \{0,1\}$):

$$\langle U \mid U\rangle = \sum_{b^1,b^2} \left(U(b^1,b^2)\right)^2 = \sum_{b^1,b^2} U(b^1,b^2) = \sum_{b^1,b^2} \mathbb{I}(a,b^1)\,\mathbb{I}(a,b^2) = \left(\sum_{b} \mathbb{I}(a,b) \right)^2.$$

Для величины $\langle V \mid V\rangle$ найдём точное значение:

$$\langle V \mid V\rangle = \sum_{b^1,b^2} S(b^1,b^2)^2 = \sum_{t=0}^{\delta} \binom{\delta}{t} 3^\delta \cdot 2^t (3^{\delta-t}\cdot 2^t)^2.$$

Дадим пояснения:

— если в (b^1,b^2) различны t позиций, то $S(b^1,b^2) = 3^{\delta-t}\cdot 2^t$ (есть два варианта зафиксировать трит в b^3, в котором b^1 и b^2 не совпали, и три варианта зафиксировать трит, в котором они совпали);

— всего имеется $\binom{\delta}{t} 3^\delta \cdot 2^t$ способов выбрать два вектора из тритов так, чтобы у них было n различных позиций:

1) первый вектор выбирается любым из возможных 3^δ способов,
2) второй вектор выбирается отличным от первого в t фиксированных позициях, что даёт сомножитель 2^t.

Упрощая последнюю сумму, получим

$$\langle V \mid V\rangle = 3^{3\delta} \sum_{t=0}^{\delta} \binom{\delta}{t} \left(\frac{2^3}{3^2}\right)^t = 3^{3\delta} \left(1+\frac{8}{9}\right)^\delta = 51^\delta.$$

Таким образом, имеем

$$\mathsf{P}[A \cap \overline{B}] = \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|^3} \sum_{b^1, b^2 \in \mathbb{B}^2} \mathbb{I}(par, b^1)\, \mathbb{I}(par, b^2)\, S(b^1, b^2) = \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|^3} \langle U \mid V \rangle \leqslant$$

$$\leqslant \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|^3} \sqrt{\langle U \mid U \rangle}\, \sqrt{\langle V \mid V \rangle} \leqslant \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|^3} \sqrt{\left(\sum_b \mathbb{I}(a, b) \right)^2}\, \sqrt{51^\delta} =$$

$$= \frac{(51)^{\delta/2}}{3^{2\delta}} \sum_{par} \mathsf{P}[par] \frac{1}{|\mathbb{B}|} \sum_b \mathbb{I}(a, b) = \mathbb{E}[X] \left(\frac{17}{27} \right)^{\delta/2}.$$

Следовательно, вероятность успешного завершения алгоритма $\mathcal{B}$ ограничена снизу следующим образом:

$$p_{\mathcal{B}} = \mathsf{P}[A \cap B] = \mathsf{P}[A] - \mathsf{P}[A \cap \overline{B}] \geqslant \mathbb{E}[X^3] - \mathbb{E}[X]\, (17/27)^{\delta/2}.$$

Функция $\phi(x) = x^3 - \alpha x$, $\alpha = (17/27)^{\delta/2}$, является выпуклой при $x \geqslant 0$, и для неё выполняется неравенство Йенсена $\phi(\mathbb{E}[X]) \leqslant \mathbb{E}[\phi(X)]$, а значит,

$$p_{\mathcal{B}} \geqslant (\mathbb{E}[X])^3 - \mathbb{E}[X]\, (17/27)^{\delta/2} = \varepsilon^3 - \varepsilon\, (17/27)^{\delta/2}.$$

Лемма 1 доказана. ■

3. Приложение леммы к анализу схемы подписи

Применим лемму 1 к анализу стойкости схемы подписи на основе схемы идентификации Штерна. Доказательство состоит из нескольких шагов:

1) В модели программируемого случайного оракула (подробнее см. [11]) запросы противника на получение подписи сообщения можно моделировать (это свойство по сути следует из свойства нулевого разглашения протокола идентификации и возможности задать ответы случайного оракула на выбранных входах). Таким образом, возможно получить сведение модели SUF-CMA к модели NMA.
2) Модель NMA с q_f запросами к случайному оракулу может быть сведена к случаю одного запроса. Такая модель проще поддаётся анализу.
3) Если противник в случае одного запроса к случайному оракулу с высокой вероятностью успешно завершает атаку, то его можно перезапустить несколько раз (см. лемму 1) и с высокой вероятностью получить несколько «существенно различных» подделок подписи. В нашем случае это означает, что, имея три различные подделки подписи, можно либо найти коллизию внутренней хеш-функции h, либо решить задачу синдромного декодирования. Поскольку обе задачи на данный момент считаются сложными, отсюда можно заключить, что схема подписи является стойкой в исходной модели SUF-CMA.

Первые два этапа уже рассматривались ранее в [6], здесь мы приведём альтернативное доказательство этих фактов.

3.1. Сведение модели SUF-CMA к модели NMA

Основная идея доказательства заключается в следующем: если бы противник знал заранее, каким будет бит b_i для каждого раунда схемы подписи (т. е. знал бы выход внешней хеш-функции $f(\cdot)$), то он мог бы самостоятельно смоделировать подписание сообщения, не зная секретного ключа (свойство нулевого разглашения схемы идентификации Штерна). В модели случайного оракула мы делаем именно это: сначала

заранее выбираем значение выхода b, генерируем корректную подпись (C, R) для выбранного b, а затем перепрограммируем случайный оракул так, чтобы на сформированном входе $(m\|C)$ оракул выдавал бы значение b: $f(m\|C) \leftarrow b$.

Теорема 1. Выполняется неравенство

$$\mathrm{InSec}^{\text{SUF-CMA}}(t, q_f, q_s) \leqslant \mathrm{InSec}^{\text{SUF-CMA}}(t + T_{\text{SIG}} \cdot q_s, q_f) + \frac{(2q_f + q_s)q_s}{2^{\delta\lambda+1}},$$

где используются следующие обозначения:

— q_f: число запросов к случайному оракулу F;
— q_s: число запросов к Sign;
— δ: число раундов в схеме подписи;
— λ: min-энтропия величины $h(X)$, X выбирается случайно равновероятно из множества $\{0,1\}^n$;
— T_{SIG}: время вычисления подписи (количество элементарных операций, необходимых для вычисления).

Доказательство. Пусть $\mathcal{A}$ — противник для схемы подписи в модели SUF-CMA. По нему мы построим противника $\mathcal{B}$ в модели NMA для схемы подписи. Поскольку в модели NMA отсутствует оракул подписи Sign, противник $\mathcal{B}$ должен моделировать оракул Sign самостоятельно. Для этого противник $\mathcal{B}$ будет вести таблицу запросов к случайному оракулу $F[\cdot]$.

При запросе противника $\mathcal{A}$ к оракулу подписи на входе m:

1) $\mathcal{B}$ генерирует элемент $b \leftarrow^{\text{R}} \mathbb{B}$;
2) $\mathcal{B}$ генерирует векторы u_i и перестановки σ_i в соответствии со значением b_i и вычисляет C_i и R_i. Это можно сделать в силу свойства нулевого разглашения схемы идентификации Штерна (см. [7] и разд. 1.2), причём распределение сгенерированных (без знания секрета) элементов (C_i, R_i) статистически неотличимо от распределения элементов (C_i, R_i), сгенерированных по секретному значению;
3) если значение $m\|C$ уже запрашивалось ранее (т. е. $F[m\|C] \neq \Lambda$), то $\mathcal{B}$ прерывает эксперимент (выдаёт ошибку). Если $F[m\|C] = \Lambda$, то задать значение $F[m\|C] \leftarrow b$ (программирование случайного оракула f).

При запросе x противника $\mathcal{A}$ к случайному оракулу F:

— если $F[x] \neq \Lambda$, то возвращается $F[x]$;
— в противном случае $\mathcal{B}$ запрашивает значение на входе x у своего случайного оракула, его ответ b записывает в таблицу $F[x] \leftarrow b$ и возвращает вектор b противнику $\mathcal{A}$.

Выпишем псевдокод противника $\mathcal{B}$ (генерация (R_i, C_i) описана в замечании 1):

$\mathcal{B}^{\mathcal{F}}(\mathcal{A})$

$F \leftarrow [\,]$
$(C, R) \leftarrow^{\mathrm{R}} \mathcal{A}^{\mathrm{SimSign},F}$
return (C, R)

$F(x)$

if $F[x] = \Lambda$
  $F[x] \leftarrow^{\mathrm{R}} \mathcal{F}(x)$
fi
return $F[x]$

SimSign(m)

$b \leftarrow^{\mathrm{R}} \{0,1,2\}^{\delta}$
for $(0 \leqslant i < \delta)$
  generate R_i, C_i
endfor
$x \leftarrow m \| C_0 \| \ldots \| C_{\delta-1}$
if $(F[x] \neq \Lambda)$
  $flag \leftarrow \texttt{true}$
  return $\perp$
else
  $F[x] \leftarrow b$
  return (C, R)
fi

Как подчёркивалось ранее (см. замечание 1), распределение величин C_i и R_i такое же, как и в случае знания секрета. До тех пор, пока не случилась коллизия (строка $flag \leftarrow \texttt{true}$), распределение ответов в «настоящей» модели SUF-CMA не отличается от распределения ответов в модели NMA, где оракул Sign заменяется на оракул SimSign. При этом для корректной симуляции необходимо затратить порядка $T_{\mathrm{SIG}} \cdot q_s$ элементарных операций (симуляция подписи q_s раз).

Найдём теперь вероятность коллизии. Ключи массива F формируются следующим образом: либо они были запрошены через оракул $F(x)$ у случайного оракула $\mathcal{F}$, либо сформированы при симуляции подписи в SimSign. Ключи при формировании подписи в интерфейсе SimSign имеют вид $m\|C_1\| \ldots \|C_\delta$. В каждый из C_i входит подстрока вида $h(X)$, где X выбирается случайно независимо равновероятно из множества $\{0,1\}^n$ (см. замечание 1). Для того чтобы случилась коллизия, необходимо, чтобы все δ раундовых значений C_i совпали как подстроки с подстроками запрашиваемых ранее у оракула $\mathcal{F}$ запросов.

Обозначим $y \leftarrow C_1\| \ldots \|C_\delta$. Пусть среди ключей массива F на очередном шаге уже содержатся q некоторых значений $x_1, \ldots, x_q$, тогда вероятность коллизии не превышает $q \cdot \mathsf{P}[y = x]$ для некоторого $x \in \{x_1, \ldots, x_q\}$. Эту вероятность (в силу независимости подстрок вида $h(X)$ в строках C_i) можно оценить сверху следующим образом:

$$\mathsf{P}[y = x] \leqslant \prod_{i=1}^{\delta} \left(\mathsf{P}[h(X) = x']\right) = \left(\sum_c \mathsf{P}[h(X) = c]\,\mathsf{P}[x' = c]\right)^{\delta} \leqslant \left(\sum_c 2^{-\lambda}\,\mathsf{P}[x' = c]\right)^{\delta} = 2^{-\delta\lambda},$$

где λ — min-энтропия величины $h(X)$, $X \leftarrow^{\mathrm{R}} \{0,1\}^n$.

В таком случае вероятность коллизии ключей F можно оценить как

$$\mathsf{P}[F\text{-coll}] \leqslant (q_f + (q_f + 1) + \ldots + (q_f + q_s - 1))\, 2^{-\delta\lambda} \leqslant \frac{(2q_f + q_s)q_s}{2^{\delta\lambda+1}},$$

поскольку при первом запросе к оракулу SimSign в массиве $F[\cdot]$ содержится не более q_f ключей, при втором — не более $q_f + 1$ ключей и так далее.

Таким образом, по лемме 1 из работы [12] имеем

$$\mathrm{InSec}^{\text{SUF-CMA}}(t, q_f, q_s) \leqslant \mathrm{InSec}^{\text{SUF-CMA}}(t + T_{\mathrm{SIG}} \cdot q_s, q_f) + \frac{(2q_f + q_s)q_s}{2^{\delta\lambda+1}}.$$

Теорема 1 доказана. ■

Замечание 2. Заметим, что в доказательстве теоремы 1 мы не использовали условия (не)равенства различных значений m. Это соответствует тому факту, что в модели SUF-CMA противнику для успеха необходимо подделать новую (не запрошенную ранее) подпись под некоторым сообщением m (при этом, вообще говоря, не требуется, чтобы сообщение никогда ранее не запрашивалось у оракула подписи).

3.2. Сведение случая q запросов к одному запросу

Теорема 2. Выполняется следующее неравенство:

$$\mathrm{InSec}^{\mathrm{NMA}}(t, q_f) \leqslant q_f \,\mathrm{InSec}^{\mathrm{NMA}}(t + \delta \cdot q_f, 1).$$

Доказательство. Без ограничения общности можем считать, что противник $\mathcal{A}$ не повторяет запросы к оракулу. Перед началом эксперимента противник $\mathcal{B}$ выбирает $l \leftarrow^{\mathrm{R}} \{1, \ldots, q_f\}$ — номер запроса, который он перенаправит «настоящему» случайному оракулу. На i-й запрос m_i к оракулу хеширования от противника $\mathcal{A}$ противник $\mathcal{B}$:

— при $i \neq l$ возвращает значение $b \leftarrow^{\mathrm{R}} \mathbb{B}$;
— при $i = l$ возвращает ответ оракула $\mathcal{F}(m_i)$.

В конце эксперимента $\mathcal{B}$ выдаёт подделку (m, τ), полученную от $\mathcal{A}$. Найдём вероятность успешной подделки подписи противником $\mathcal{B}$. Если $m \in \{m_1, \ldots, m_{q_f}\}$, то $m = m_l$ с вероятностью q_f^{-1}, поскольку распределения всех $\mathcal{F}(m_i)$ одинаковы и индекс l выбран случайно независимо. Если $m \notin \{m_1, \ldots, m_{q_f}\}$, то вероятность успеха противника $\mathcal{A}$ равна вероятности успеха противника $\mathcal{B}$, поскольку для корректности подписи необходимо, чтобы хеш-значение b, задействованное в подписи, совпало с $f(m)$, не запрашиваемым ранее. Отсюда следует утверждение теоремы. ■

3.3. Применение троичной леммы о разветвлении для схемы подписи

Теорема 3. Выполнены следующие неравенства:

$$\mathrm{InSec}^{\mathrm{SUF\text{-}CMA}}(t, q_f, q_s) \leqslant q_f \,\mathrm{InSec}^{\mathrm{NMA}}(t + T_{\mathrm{SIG}} \cdot q_s + \delta \cdot q_f, 1) + \frac{(2q_f + q_s)\, q_s}{2^{\delta\lambda+1}}; \tag{1}$$

$$\varepsilon_t^3 - \varepsilon_t \,(17/27)^{\delta/2} \leqslant \mathrm{InSec}^{\mathrm{SD}}(4t) + \mathrm{InSec}^{\mathrm{Coll}}(4t) + 3^{1-\delta}, \tag{2}$$

где $\varepsilon_t = \mathrm{InSec}^{\mathrm{NMA}}(t, 1)$; остальные обозначения те же, что в теореме 1.

Доказательство. Из теоремы 1 следует

$$\mathrm{InSec}^{\mathrm{SUF\text{-}CMA}}(t, q_f, q_s) \leqslant \mathrm{InSec}^{\mathrm{NMA}}(t + T_{\mathrm{SIG}} \cdot q_s, q_f),$$

из теоремы 2 следует оценка (1).

Найдём ограничение на величину ε_t. Рассмотрим алгоритм $\mathcal{A}(par, b)$, где $b \in \mathbb{B}$ — ответ случайного оракула $\mathcal{F}$ на единственный запрос от $\mathcal{A}$; par — открытый ключ $par = \mathrm{pk} = Hs$ рассматриваемой криптосистемы (зафиксирован в эксперименте). Пусть ρ — случайная лента противника $\mathcal{A}$. Выходом противника x является пара (m, τ) — подделка подписи сообщения. Согласно теореме 1, найдётся противник $\mathcal{B}$, который выдаёт три подделки (m_0, τ_0), (m_1, τ_1), (m_2, τ_2), $\tau_i = (C^i, R^i)$, полученные на некоторых значениях b^0, b^1, b^2, в которых найдётся позиция j, такая, что (без ограничения общности)

$$b_j^0 = 0, \quad b_j^1 = 1, \quad b_j^2 = 2.$$

При этом для сообщений m_i выполняется равенство $\mathcal{F}(m_i\|C^i) = b^i$ (в противном случае подпись τ_i неверна). Следовательно, сообщение $m_i\|C^i$ с большой вероятностью запрашивалось в качестве единственного запроса к оракулу (в противном случае каждая подпись верна с вероятностью не более чем $3^{-\delta}$). Но при этом запрос к оракулу $\mathcal{F}$ (значение $m\|C_1\|\ldots\|C_\delta$) зависит только от ленты противника ρ и открытого ключа pk (т. е. от параметров par), а следовательно, не меняется от запуска к запуску. Таким образом, с вероятностью не менее чем $1 - 3^{1-\delta}$ выполнено равенство $m_0\|C^0 = m_1\|C^1 = m_2\|C^2$. Если равенство не выполнено, то эксперимент прерывается.

Рассмотрим соответствующие позиции j значения подписей (C_j^0, R_j^0), (C_j^1, R_j^1), (C_j^2, R_j^2). По определению схемы подписи имеем

— $R_j^0 = (\sigma\|u)\colon h(\sigma\|Hu) = C_j^0, h(\sigma(u)) = C_j^1$;
— $R_j^1 = (\pi\|v)\colon h(\pi\|Hv \oplus y) = C_j^0, h(\pi(v)) = C_j^2$;
— $R_j^2 = (w_1\|w_2)\colon h(w_1) = C_j^1, h(w_1 \oplus w_2) = C_j^2, \mathrm{wt}(w_2) = \omega$.

Если при этом была построена коллизия для внутренней хеш-функции h, то эксперимент прерывается. Вероятность этого события может быть оценена сверху величиной $\mathrm{InSec}^{\mathrm{Coll}}(t_{\mathcal{B}})$.

В случае отсутствия коллизий $\mathcal{B}$ восстанавливает вектор $s = \pi^{-1}(w_2)$. Из условий отсутствия коллизий получим

$$H\pi^{-1}(w_2) = H(v \oplus \pi^{-1}(w_1)) = H(v \oplus u) = Hv \oplus Hv \oplus y = y,$$

причём $\mathrm{wt}(\pi^{-1}(w_2)) = \mathrm{wt}(w_2) = \omega$.

Вероятность восстановления вектора s (секретного ключа) может быть оценена сверху величиной $\mathrm{InSec}^{\mathrm{SD}}(t_{\mathcal{B}})$. Отсюда следует, что выполнено неравенство

$$\varepsilon_t^3 - \varepsilon_t\,(17/27)^{\delta/2} \leqslant \mathrm{InSec}^{\mathrm{SD}}(4t) + \mathrm{InSec}^{\mathrm{Coll}}(4t) + 3^{1-\delta},$$

что и требовалось доказать. ■

3.4. Практическая значимость оценки

Асимптотическое поведение оценки

При большом числе раундов $\delta \geqslant \Delta$ слагаемое вида $\varepsilon_t\,(17/27)^{\delta/2}$ вносит всё меньший вклад, и в пределе неравенство (2) переходит в асимптотическое неравенство вида

$$\mathrm{InSec}^{\mathrm{NMA}}(t, 1) \leqslant \sqrt[3]{\mathrm{InSec}^{\mathrm{SD}}(4t) + \mathrm{InSec}^{\mathrm{Coll}}(4t) + 3^{1-\delta}}.$$

Совместно с оценкой (1) получаем следующую оценку сверху на величину $\mathrm{InSec}^{\text{SUF-CMA}}$:

$$\mathrm{InSec}^{\text{SUF-CMA}}(t, q_f, q_s) \leqslant q_f \sqrt[3]{\mathrm{InSec}^{\mathrm{SD}}(4T) + \mathrm{InSec}^{\mathrm{Coll}}(4T) + 3^{1-\delta}} + \frac{(2q_f + q_s)\,q_s}{2^{\delta\lambda+1}},$$

где $T = t + T_{\mathrm{SIG}}\cdot q_s + \delta\cdot q_f$, что меньше оценки, полученной в [6], примерно в $14\cdot\sqrt[3]{\delta^2/4}$ раз (время работы противника, рассматриваемого в [6], при $q_f, q_s \ll t$, примерно равно $\delta^2 t$, также в оценке [6] присутствует сомножитель $\sqrt[3]{1920/(1-e^{-1}} \approx 14$).

Сравнение оценок при конкретных значениях параметров

К сожалению, для интересных на практике значений числа раундов δ оценка для $\mathrm{InSec}^{\mathrm{NMA}}(t, 1)$, полученная в настоящей работе, намного хуже полученной в [6].

Примем следующие значения параметров задач SD и Coll:

$$n = 2896,\ k = 1448,\ \omega = 318,\ \ell = 512.$$

Согласно работе [13] и программному коду [14], оптимальный алгоритм для заданных (n, k, ω) решает задачу SD с вероятностью 1 за $T_{\mathrm{SD}} = 2^{323}$ битовых операций.

Пусть

$$\mathrm{InSec}^{\mathrm{SD}}(t) = t/T_{\mathrm{SD}}, \quad \mathrm{InSec}^{\mathrm{Coll}}(t) = t^2/2^{\ell}.$$

Примем (следуя работе [6]) следующие значения параметров: $t = 2^{70}$ элементарных операций, $\ell = 512$. В таблице в первом столбце указано число раундов в схеме подписи δ, во втором и третьем — двоичный логарифм от оценки сверху для величины $\mathrm{InSec}^{\mathrm{NMA}}(t, 1)$ (для второго столбца использована оценка [6, теорема 1], для третьего — оценка (2)). Заметим, что оценка (2) становится лучше при $\delta = 414$ раундах схемы подписи.

δ	Прошлая оценка	Текущая оценка
100	−58,5	−16,69
137	−71,16	−22,86
200	−70,43	−33,37
300	−69,65	−50,06
500	−68,67	−83,21

Заключение

В работе доказана троичная версия леммы о развлетвлении. Полученная лемма применена для анализа стойкости схемы подписи, основанной на схеме идентификации Штерна, в модели SUF-CMA. Направлением дальнейших исследований может быть получение более сильных оценок для практически значимых результатов при малом числе раундов, а также обобщение доказательства на модель квантового доступа к случайному оракулу QROM.

Автор благодарит Александру Бабуеву и Викторию Высоцкую за полезные обсуждения в ходе работы, а также рецензента, замечания которого помогли улучшить статью.

ЛИТЕРАТУРА

1. *Pointcheval D. and Stern J.* Security proofs for signature schemes // EUROCRYPT'96. LNCS. 1996. V. 1070. P. 387–398.
2. *Damgård I.* On Σ-protocols. Lecture Notes. University of Aarhus, Department of Computer Science, 2002.
3. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems // LNCS. 1987. V. 263. P. 186–194.
4. *Черемушкин А. В.* Криптографические протоколы: основные свойства и уязвимости // Прикладная дискретная математика. Приложение. 2009. № 2. С. 115–150.
5. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma // Proc. 13th ACM Conf. CCM'06. 2006. P. 390–399.
6. *Vysotskaya V. V. and Chizhov I. V.* The security of the code-based signature scheme based on the Stern identification protocol // Прикладная дискретная математика. 2022. Т. 57. С. 67–90.
7. *Stern J.* A new identification scheme based on syndrome decoding // LNCS. 1993. V. 773. P. 13–21.
8. *Katz J. and Lindell Y.* Introduction to Modern Cryptography: 3d Ed. Chapman & Hall/CRC Cryptography and Network Security Series, 2021. 628 pp.

9. *Berlekamp E., McEliece R., and Van Tilborg H.* On the inherent intractability of certain coding problems // IEEE Trans. Inform. Theory. 1978. V. 24. No. 3. P. 384–386.
10. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols // Proc. 1st ACM Conf. CCS'93. 1993. P. 62–73.
11. *Fischlin M., Lehmann A., Ristenpart T., et al.* Random oracles with (out) programmability // LNCS. 2010. V. 6477. P. 303–320.
12. *Shoup V.* Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive. 2004. Paper 2004/332.
13. *Esser A. and Bellini E.* Syndrome Decoding Estimator. Cryptology ePrint Archive. 2021. Paper 2021/1243.
14. `https://github.com/Crypto-TII/syndrome_decoding_estimator` — Syndrome Decoding Estimator. 2021.

REFERENCES

1. *Pointcheval D. and Stern J.* Security proofs for signature schemes. EUROCRYPT'96, LNCS, 1996, vol. 1070, pp. 387–398.
2. *Damgård I.* On Σ-protocols. Lecture Notes, University of Aarhus, Department of Computer Science, 2002.
3. *Fiat A. and Shamir A.* How to prove yourself: Practical solutions to identification and signature problems. LNCS, 1987, vol. 263, pp. 186–194.
4. *Cheremushkin A. V.* Kriptograficheskie protokoly: osnovnye svoystva i uyazvimosti [Cryptographic protocols: main properties and vulnerabilities]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2009, no. 2, pp. 115–150. (in Russian)
5. *Bellare M. and Neven G.* Multi-signatures in the plain public-key model and a general forking lemma. Proc. 13th ACM Conf. CCM'06, 2006, pp. 390–399.
6. *Vysotskaya V. V. and Chizhov I. V.* The security of the code-based signature scheme based on the Stern identification protocol. Prikladnaya Diskretnaya Matematika, 2022, vol. 57, pp. 67–90.
7. *Stern, J.* A new identification scheme based on syndrome decoding. LNCS, 1993, vol. 773, pp. 13–21.
8. *Katz J. and Lindell Y.* Introduction to Modern Cryptography. 3d Ed. Chapman & Hall/CRC Cryptography and Network Security Series, 2021. 628 p.
9. *Berlekamp E., McEliece R., and Van Tilborg H.* On the inherent intractability of certain coding problems. IEEE Trans. Inform. Theory, 1978, vol. 24, no. 3, pp. 384–386.
10. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols. Proc. 1st ACM Conf. CCS'93, 1993, pp. 62–73.
11. *Fischlin M., Lehmann A., Ristenpart T., et al.* Random oracles with (out) programmability. LNCS, 2010, vol. 6477, pp. 303–320.
12. *Shoup V.* Sequences of Games: a Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, 2004, Paper 2004/332.
13. *Esser A. and Bellini E.* Syndrome Decoding Estimator. Cryptology ePrint Archive, 2021, Paper 2021/1243.
14. `https://github.com/Crypto-TII/syndrome_decoding_estimator` — Syndrome Decoding Estimator, 2021.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.175.3 DOI 10.17223/20710410/59/4

АСИМПТОТИКА ЧИСЛА ПОМЕЧЕННЫХ ПЛАНАРНЫХ ТЕТРАЦИКЛИЧЕСКИХ И ПЕНТАЦИКЛИЧЕСКИХ ГРАФОВ

В. А. Воблый

Всероссийский институт научной и технической информации РАН, г. Москва, Россия

E-mail: vitvobl@yandex.ru

Связный граф с цикломатическим числом равным k называется k-циклическим графом. Получена формула для числа помеченных непланарных пентациклических графов с заданным числом вершин, а также найдена асимптотика числа помеченных связных планарных тетрациклических и пентациклических графов с n вершинами при $n \to \infty$. Доказано, что при равномерном распределении вероятностей на множестве рассматриваемых графов вероятность того, что помеченный тетрациклический граф является планарным, асимптотически равна 1089/1105, а вероятность того, что помеченный пентациклический граф является планарным, асимптотически равна 1591/1675.

Ключевые слова: *помеченный граф, планарный граф, тетрациклический граф, пентациклический граф, блок, перечисление, асимптотика, вероятность.*

AN ASYMPTOTICS FOR THE NUMBER OF LABELLED PLANAR TETRACYCLIC AND PENTACYCLIC GRAPHS

V. A. Voblyi

Russian Institut for Scientific and Technical Information, Moscow, Russia

A connected graph with a cyclomatic number k is said to be a k-cyclic graph. We obtain the formula for the number of labelled non-planar pentacyclic graphs with a given number of vertices, and find the asymptotics of the number of labelled connected planar tetracyclic and pentacyclic graphs with n vertices as $n \to \infty$. We prove that under a uniform probability distribution on the set of graphs under consideration, the probability that the labelled tetracyclic graph is planar is asymptotically equal to 1089/1105, and the probability that the labeled pentacyclic graph is planar is asymptotically equal to 1591/1675.

Keywords: *labelled graph, planar graph, tetracyclic graph, pentacyclic graph, block, enumeration, asymptotics, probability.*

Введение

Планарные графы применяются при проектировании СБИС [1, 2], в теории кодирования [3], в физике при нахождении статистической суммы [4] и в компьютерном зрении [5]. Эти и другие применения планарных графов обуславливают актуальность их перечисления.

О. Гименез и М. Ной асимптотически перечислили помеченные планарные графы по числу вершин [6]. Е. Бендер нашёл асимптотику числа помеченных 2-связных планарных графов с n вершинами при $n \to \infty$ [7]. Однако в общем случае асимптотика числа помеченных планарных n-вершинных k-циклических графов при $n \to \infty$ неизвестна. В [8] получена явная формула для числа помеченных связных непланарных тетрациклических графов с заданным числом вершин. В [9] найдена явная формула для числа помеченных связных непланарных пентациклических блоков с заданным числом вершин, а также получена соответствующая асимптотика.

В данной работе получена формула для числа помеченных связных непланарных пентациклических графов с заданным числом вершин, а также найдена асимптотика числа помеченных связных планарных n-вершинных тетрациклических и пентациклических графов при $n \to \infty$.

1. Перечисление графов

Рассматриваются неориентированные простые связные графы.

Определение 1. Для связного графа *точкой сочленения* называется его вершина, после удаления которой вместе с инцидентными ей рёбрами граф становится несвязным. *Блок* — это связный граф без точек сочленения, а также максимальный связный нетривиальный подграф, не имеющий точек сочленения [10, с. 41].

Определение 2. *Цикломатическим числом* связного графа называется увеличенная на единицу разность между числом рёбер и вершин графа, *k-циклический граф* — это граф с цикломатическим числом k.

Определение 3. Граф называется *планарным*, если его можно уложить на плоскости без пересечения рёбер [10, с. 127].

Определение 4. *Кактусом* называется связный граф, в котором нет рёбер, лежащих более чем на одном простом цикле [11, с. 93]. Все блоки кактуса — рёбра или простые циклы.

Определение 5. Класс графов называется *блочно-устойчивым*, если граф принадлежит этому классу тогда и только тогда, когда каждый блок графа принадлежит этому классу [12].

Пусть $P(n,k)$ ($\bar{P}(n,k)$) — число помеченных связных планарных (непланарных) k-циклических графов с n вершинами.

Теорема 1. Число $\bar{P}(n,5)$ помеченных связных непланарных пентациклических графов с n вершинами при $n \geqslant 6$ равно

$$\bar{P}(n,5) = (n-1)![z^{-1}]e^{nz}\Big(\frac{nz^7(2+z)}{48(1-z)^{11}} + \frac{z^5(12+47z+9z^2-8z^3)}{24(1-z)^{13}}\Big)z^{-n}. \qquad (1)$$

Доказательство. Обозначим через $B(n,k)$ число помеченных k-циклических блоков с n вершинами; $B_k(z)$ — экспоненциальную производящую функцию для $B(n,k)$; $C(n,k)$ — число помеченных связных k-циклических графов с n вершинами.

Известна [13, 14] формула

$$C(n,k) = \frac{(n-1)!}{nk!}[z^{-1}]e^{nz}Y_k(n1!B_1'(z), n2!B_2'(z), \ldots, nk!B_k'(z))z^{-n}, \qquad (2)$$

где $[z^{-1}]$ — оператор формального вычета [15, с. 25], а $Y_k(x_1,\ldots,x_k)$ — многочлены разбиений (многочлены Белла). Для этих многочленов известно выражение [16, с. 173]

$$Y_k(x_1,\ldots,x_k) = \sum_{\pi(k)} \frac{k!}{m_1!\ldots m_k!}\Big(\frac{x_1}{1!}\Big)^{m_1}\ldots\Big(\frac{x_k}{k!}\Big)^{m_k},$$

где суммирование проводится по всем разбиениям $\pi(k)$ числа k, то есть по всем неотрицательным решениям $(m_1, m_2, \cdots, m_k)$ уравнения $m_1 + 2m_2 + \ldots + km_k = k$, $m_i \geqslant 0$, $i = 1, \ldots, k$.

Формула (2) верна не только для всего класса связных графов, но и для блочно-устойчивого его подкласса [13]. Известно, что класс планарных графов является блочно-устойчивым [12].

Так как [15, с. 246]

$$Y_5(x_1, x_2, x_3, x_4, x_5) = x_1^5 + 10x_1^3x_2 + 15x_1x_2^2 + 10x_1^2x_3 + 10x_2x_3 + 5x_1x_4 + x_5$$

и $x_i = ni!B_i'(z)$, имеем

$$C(n,5) = \frac{(n-1)!}{120n}[z^{-1}]e^{nz}\Big(n^5(B_1'(z))^5 + 20n^4(B_1'(z))^3B_2'(z) + 60n^3B_1'(z)(B_2'(z))^2 + \\ +60n^3(B_1'(z))^2B_3'(z) + 120n^2B_2'(z)B_3'(z) + 120n^2B_1'(z)B_4'(z) + 120nB_5'(z)\Big)z^{-n}. \tag{3}$$

Обозначим через $\bar{B}(n,k)$ число помеченных непланарных k-циклических блоков с n вершинами, а через $\bar{B}_k(z)$ — экспоненциальную производящую функцию для $\bar{B}(n,k)$.

По теореме Понтрягина — Куратовского граф планарен только тогда, когда он не содержит подграфов, гомеоморфных полному графу K_5 или полному двудольному графу $K_{3,3}$. Так как граф K_5 является 6-циклическим, а граф $K_{3,3}$ — 4-циклическим, все униццклические, бициклические и трициклические блоки не могут содержать таких подграфов и все эти блоки являются планарными графами. Поэтому в выражение для $\bar{P}(n,5)$ войдут только слагаемые из разложения (3), содержащие производящие функции $\bar{B}_4(z)$ и $\bar{B}_5(z)$ для непланарных блоков:

$$\bar{P}(n,5) = \frac{(n-1)!}{120n}[z^{-1}]e^{nz}\Big(120n^2B_1'(z)\bar{B}_4'(z) + 120n\bar{B}_5'(z)\Big)z^{-n}. \tag{4}$$

Так как уницикличесий блок — это простой цикл, имеем

$$B(n,1) = (n-1)!/2, \quad B_1(z) = \sum_{n=3}^{\infty}\frac{1}{2}(n-1)!\frac{z^n}{n!}, \quad B_1'(z) = \frac{z^2}{2(1-z)}.$$

В [8] получена формула $\bar{B}(n,4) = \dfrac{n!}{72}\dbinom{n+2}{8}$, из которой следуют выражения

$$\bar{B}_4(z) = \frac{z^6}{72(1-z)^9}, \quad \bar{B}_4'(z) = \frac{2z^5 + z^6}{24(1-z)^{10}},$$

а также найдено значение

$$\bar{P}(n,4) = \frac{n!}{72}\sum_{k=6}^{n}\binom{k+2}{8}\frac{kn^{n-k-1}}{(n-k)!}. \tag{5}$$

В [9] доказана формула

$$\bar{B}(n,5) = \frac{n!}{380160}\binom{n+1}{7}(10n^4 + 118n^3 + 72n^2 - 1232n - 1968),$$

из которой следует

$$\bar{B}_5(z) = \frac{z^6(2+5z-2z^2)}{24(1-z)^{12}}, \quad \bar{B}_5'(z) = \frac{z^5(12+47z+9z^2-8z^3)}{24(1-z)^{13}}.$$

Подставляя выражения для $\bar{B}_4(z)$ и $\bar{B}_5(z)$ в (4), получим формулу (1). ■

В таблице представлены числа $\bar{P}(n,4)$ и $\bar{P}(n,5)$, вычисленные с помощью формул (5), (1) и пакета программ Maple.

n	6	7	8	9	10	11	12
$\bar{P}(n,4)$	10	1050	73920	4483080	256032000	14353651620	807516864000
$\bar{P}(n,5)$	60	8610	781200	58688280	4034520000	266400523620	17353002522240

2. Асимптотика и вероятность

Теорема 2. Для числа $P(n,4)$ помеченных связных планарных тетрациклических графов с n вершинами при $n \to \infty$ верна асимптотика

$$P(n,4) \sim \frac{121 n^{n+4}}{13440}\,. \tag{6}$$

Доказательство. Очевидно, имеем $P(n,4) = C(n,4) - \bar{P}(n,4)$.

Пусть $f(n,n+k)$ — число помеченных графов с n вершинами и $(n+k)$ рёбрами. Е. Райт нашёл асимптотику при $n \to \infty$ и $k = O(n^{1/2})$ [17]:

$$\begin{gathered} f(n,n+k) \sim \rho_k n^{n+(3k-1)/2}, \quad \rho_k = \frac{\sqrt{\pi}\sigma_k}{2^{(3k-1)/2}\Gamma((3k/2)+1)}, \\ \sigma_0 = \frac{1}{4}, \quad \sigma_1 = \frac{5}{16}, \quad \sigma_2 = \frac{15}{16}, \quad \sigma_{k+1} = \frac{3}{2}(k+1)\sigma_k + \sum_{s=1}^{k-1}\sigma_s\sigma_{k-s}, \ k \geqslant 2. \end{gathered} \tag{7}$$

С помощью формулы для гамма-функции $\Gamma\left(n+\frac{1}{2}\right) = \frac{(2n-1)!!}{2^n}\sqrt{\pi}$ при $n \to \infty$ найдём

$$\begin{gathered} f(n,n+3) \sim \rho_3 n^{n+4}, \quad \rho_3 = \frac{\sqrt{\pi}\sigma_3}{2^4\Gamma(11/2)} = \frac{\sqrt{\pi}\sigma_3}{16\frac{9!!}{2^5}\sqrt{\pi}} = \frac{2\sigma_3}{9\cdot 7\cdot 5\cdot 3} = \frac{2\sigma_3}{945}, \\ \sigma_3 = \frac{9}{2}\sigma_2 + \sigma_1^2 = \frac{9}{2}\cdot\frac{15}{16} + \frac{25}{256} = \frac{1105}{256}, \quad \rho_3 = \frac{2\cdot 1105}{945\cdot 256} = \frac{221}{24192}, \\ C(n,4) = f(n,n+3) \sim \frac{221}{24192} n^{n+4}. \end{gathered} \tag{8}$$

Используем следующую лемму:

Лемма 1 [18]. Обозначим

$$A_n(m,q) = [z^{-1}]\frac{p(z,q)e^{nz}z^{-n}}{(1-z)^m}, \quad p(z,q) = \sum_{i=0}^{q} c_i z^i,$$

тогда при фиксированных m, q и $n \to \infty$ верна асимптотика

$$A_n(m,q) \sim \frac{\sqrt{\pi}p(1,q))n^{n+m/2}}{n!2^{m/2}\Gamma((m+1)/2)}.$$

Из выражения (2) для тетрациклических графов с помощью леммы 1 получим

$$\bar{P}(n,4)=\frac{(n-1)!}{24n}[z^{-1}]e^{nz}24n\bar{B}_4'(z)z^{-n}=\frac{n!}{n}[z^{-1}]e^{nz}\frac{2z^5+z^6}{24(1-z)^{10}}z^{-n}\sim\frac{n!}{24n}\frac{\sqrt{\pi}3n^{n+5}}{n!2^5\Gamma(11/2)}=\frac{n^{n+4}}{7560},$$

$$P(n,4)=C(n,4)-\bar{P}(n,4)\sim\frac{221}{24192}n^{n+4}-\frac{n^{n+4}}{7560}=\frac{121}{13440}n^{n+4}.$$

Теорема 2 доказана. ■

Зададим на множестве помеченных тетрациклических графов с n вершинами равномерное распределение вероятностей.

Следствие 1. Вероятность $P_4(n)$ того, что помеченный связный тетрациклический граф является планарным, асимптотически равна $1089/1105\approx 0{,}9855$.

Доказательство. С помощью формул (6) и (8) при $n\to\infty$ найдём

$$P_4(n)=\frac{P(n,4)}{C(n,4)}\sim\frac{121n^{n+4}24192}{13440n^{n+4}221}=\frac{1089}{1105}\approx 0{,}9855.$$

Следствие 1 доказано. ■

Теорема 3. Для числа $P(n,5)$ помеченных связных планарных пентациклических графов с n вершинами при $n\to\infty$ верна асимптотика

$$P(n,5)\sim\sqrt{\frac{\pi}{2}}\frac{1591n^{n+11/2}}{1474560}. \tag{9}$$

Доказательство. Справедливо равенство $P(n,5)=C(n,5)-\bar{P}(n,5)$.
С помощью асимптотики (7) при $n\to\infty$ найдём

$$C(n,5)=f(n,n+4)\sim\rho_4 n^{n+11/2},\quad \rho_4=\frac{\sqrt{\pi}\sigma_4}{2^{11/2}\Gamma(7)},$$

$$\sigma_4=6\sigma_3+\sum_{s=1}^{2}\sigma_s\sigma_{3-s}=6\sigma_3+2\sigma_1\sigma_2=6\,\frac{1105}{256}+2\,\frac{5}{16}\,\frac{15}{16}=\frac{1695}{64}, \tag{10}$$

$$\rho_4=\frac{\sqrt{\pi}1695}{32\sqrt{2}\cdot 720\cdot 64}=\sqrt{\frac{\pi}{2}}\frac{339}{294912},\quad C(n,5)\sim\sqrt{\frac{\pi}{2}}\frac{339}{294912}n^{n+11/2}\ .$$

С помощью формулы (1) и леммы 1 получим

$$\bar{P}(n,5)=(n-1)![z^{-1}]e^{nz}\Big(\frac{nz^7(2+z)}{48(1-z)^{11}}+\frac{z^5(12+47z+9z^2-8z^3)}{24(1-z)^{13}}\Big)z^{-n}\sim$$

$$\sim n!\frac{\sqrt{\pi}3n^{n+11/2}}{n!482^{11/2}\Gamma(6)}+\frac{n!}{n}\cdot\frac{\sqrt{\pi}60n^{n+11/2}}{242^{13/2}\Gamma(7)}=\sqrt{\frac{\pi}{2}}\Big(\frac{1}{61440}+\frac{1}{18432}\Big)n^{n+11/2}=\sqrt{\frac{\pi}{2}}\frac{13n^{n+11/2}}{184320},$$

$$P(n,5)=C(n,5)-\bar{P}(n,5)\sim\sqrt{\frac{\pi}{2}}\frac{339}{294912}n^{n+11/2}-\sqrt{\frac{\pi}{2}}\frac{13n^{n+11/2}}{184320}=\sqrt{\frac{\pi}{2}}\frac{1591n^{n+11/2}}{1474560}\ .$$

Теорема 3 доказана. ■

Зададим на множестве помеченных пентациклических графов с n вершинами равномерное распределение вероятностей.

Следствие 2. Вероятность $P_5(n)$ того, что помеченный связный пентациклический граф является планарным, асимптотически равна $1591/1695\approx 0{,}9386$.

Доказательство. С помощью формул (9) и (10) при $n \to \infty$ найдём

$$P_5(n) = \frac{P(n,5)}{C(n,5)} \sim \sqrt{\frac{\pi}{2}} \frac{1591 n^{n+11/2}}{1474560} \frac{294912}{339 n^{n+11/2}\sqrt{\pi/2}} = \frac{1591}{1695} \approx 0{,}9386.$$

Следствие 2 доказано. ■

В [6] доказано, что число помеченных связных планарных графов с n вершинами при $n \to \infty$ асимптотически равно $cn^{-7/2}\gamma^n n!$, где c, γ — константы. Кроме того, известно, что число помеченных связных графов с n вершинами асимптотически равно числу помеченных не обязательно связных графов с n вершинами, то есть $2^{n(n-1)/2}$ [11]. Из этого следует, что почти все помеченные связные графы с n вершинами являются непланарными. Для фиксированного цикломатического числа k поведение числа помеченных планарных графов с n вершинами при $n \to \infty$ другое.

При равномерном распределении вероятностей на множестве помеченных связных n-вершинных k-циклических графов вероятность их планарности при $n \to \infty$ зависит от k и не равна нулю. Пусть $Ca(n,k)$ — число помеченных k-циклических кактусов с n вершинами. В [19] при фиксированном k и $n \to \infty$ получена асимптотика

$$Ca(n,k) \sim \frac{\sqrt{\pi}}{2^{3k/2} k! \Gamma\big((k+1)/2\big)} n^{n+(3k-4)/2}.$$

Так как кактусы являются планарными графами, справедливы неравенства

$$Ca(n,k) \leqslant P(n,k) \leqslant C(n,k), \ \frac{\sqrt{\pi}}{2^{3k/2} k! \Gamma\big((k+1)/2\big)} n^{n+(3k-4)/2} \leqslant P(n,k) \leqslant \rho_k n^{n+(3k-4)/2}.$$

Гипотеза. При фиксированном k и $n \to \infty$ верна асимптотика

$$P(n,k) \sim c_k n^{n+(3k-4)/2},$$

где c_k — константа, зависящая от k.

В частности, $c_1 = \sqrt{\dfrac{\pi}{8}}$, $c_2 = \dfrac{5}{24}$, $c_3 = \sqrt{\dfrac{\pi}{2}}\dfrac{5}{128}$, $c_4 = \dfrac{121}{13440}$, $c_5 = \sqrt{\dfrac{\pi}{2}}\dfrac{1591}{1474560}$. Таким образом, гипотеза верна для $k \leqslant 5$.

В заключение отметим, что Е. Ф. Дмитриев [20] другим способом перечислял помеченные планарные тетрациклические и пентациклические графы, но не опубликовал доказательства своих результатов.

Автор благодарит рецензента за полезные замечания.

ЛИТЕРАТУРА

1. *Рихтер М. Р.* Алгоритм трассировки при проектировании СБИС // Научно-технические ведомости СПбГПУ. 2011. Вып. 5. С. 111–118.
2. *Aggarwal A., Klawe M., and Shor P.* Multilayer grid embedding for VLSI // Algorirhmica. 1991. No. 6. P. 129–151.
3. *Haymaker K. and O'Pella J.* Locally recoverable codes from planar graphs // J. Algebra Comb. Discrete Appl. 2020. V. 7. No. 1. P. 35–53.
4. *Карандашев Я. М., Мальсагов М. Ю.* Полиномиальный алгоритм точного вычисления статистической суммы для модели бинарных спинов на планарных графах // Тр. НИИСИ РАН. 2017. Т. 7. № 1. С. 18–24.

5. *Schmidt F. R., Toppe E., and Cremers D.* Efficient planar graphs cuts with applications in computer vision // IEEE Computer Society Conf. Computer Vision and Pattern Recognition. Miami, Florida, 2009. P. 1–6.
6. *Gimenez O. and Noy M.* Asymptotic enumeration and limit laws of planar graphs // J. Amer. Math. Soc. 2009. V. 92. No. 2. P. 169–210.
7. *Bender E. A., Gao Z., and Wormand N. C.* The number of labeled 2-connected planar graphs // Electron. J. Combinatorics. 2002. No. 9. Article R43.
8. *Воблый В. А., Мелешко А. К.* Перечисление помеченных непланарных тетрациклических графов // Материалы XVIII Междунар. семинара «Комбинаторные конфигурации и их приложения». Кировоград, 2016. С. 33–36.
9. *Воблый В. А.* Перечисление помеченных непланарных пентациклических блоков // Итоги науки и техн. Соврем. матем. и её прилож. Темат. обзоры. 2021. Т. 193. С. 28–32.
10. *Харари Ф.* Теория графов. М.: Мир, 1973. 300 с.
11. *Харари Ф., Палмер Э.* Перечисление графов. М.: Мир, 1977. 324 с.
12. *McDiarmid C. and Scott A.* Random graphs from a block stable class // Europ. J. Combin. 2016. V. 58. P. 96–106.
13. *Воблый В. А.* Об одном подходе к перечислению помеченных связных графов: обзор результатов // Итоги науки и техн. Совр. матем. и её прилож. Темат. обзоры. 2020. Т. 188. С. 106–118.
14. *Воблый В. А.* Перечисление помеченных эйлеровых пентациклических графов // Прикладная дискретная математика. 2020. № 50. С. 87–92.
15. *Гульден Я., Джексон Д.* Перечислительная комбинаторика. М.: Наука, 1990. 504 с.
16. *Риордан Дж.* Комбинаторные тождества. М.: Наука, 1982. 256 с.
17. *Wright E. M.* The number of connected sparsely edged graphs // J. Graph Theory. 1977. V. 1. No. 4. P. 317–330.
18. *Воблый В. А.* Асимптотическое перечисление помеченных последовательно-параллельных тетрациклических графов // Итоги науки и техн. Соврем. матем. и её прилож. Темат. обзоры. 2020. Т. 187. С. 31–35.
19. *Воблый В. А.* О перечислении помеченных связных графов с заданными числами вершин и ребер // Дискрет. анализ и исслед. операций. 2016. Т. 23. № 2. С. 5–20.
20. *Дмитриев Е. Ф.* Перечисление отмеченных графов с данными структурными свойствами. Автореферат дис. ... канд. физ.-мат. наук. Институт математики АН БССР, Минск, 1986.

REFERENCES

1. *Rikhter M. R.* Algoritm trassirovki pri proektirovanii SBIS [Tracing algorithm for VLSI design]. Nauchno-Tekhnicheskie Vedomosti SPbGPU, 2011, iss. 5, pp. 111–118. (in Russian)
2. *Aggarwal A., Klawe M., and Shor P.* Multilayer grid embedding for VLSI. Algorirhmica, 1991, no. 6, pp. 129–151.
3. *Haymaker K. and O'Pella J.* Locally recoverable codes from planar graphs. J. Algebra Comb. Discrete Appl., 2020, vol. 7, no. 1, pp. 35–53.
4. *Karandashev Ya. M. and Mal'sagov M. Yu.* Polinomial'nyy algoritm tochnogo vychisleniya statisticheskoy summy dlya modeli binarnykh spinov na planarnykh grafakh [Polynomial algorithm for exact computation of the partition function for the binary spin model on planar graphs]. Proc. NIISI RAS, 2017, vol. 7, no. 1, pp. 18–24. (in Russian)
5. *Schmidt F. R., Toppe E., and Cremers D.* Efficient planar graphs cuts with applications in computer vision. IEEE Computer Society Conf. Computer Vision and Pattern Recognition, Miami, Florida, 2009, pp. 1–6.

6. *Gimenez O. and Noy M.* Asymptotic enumeration and limit laws of planar graphs. J. Amer. Math. Soc., 2009, vol. 92, no. 2, pp. 169–210.
7. *Bender E. A., Gao Z., and Wormand N. C.* The number of labeled 2-connected planar graphs. Electron. J. Combinatorics, 2002, no. 9, article R43.
8. *Voblyy V. A. and Meleshko A. K.* Perechislenie pomechennykh neplanarnykh tetratsiklicheskikh grafov [Enumeration of labeled non-planar tetracyclic graphs]. Materialy XVIII Mezhd. Seminara "Kombinatornye konfiguratsii i ikh prilozheniya", Kirovograd, 2016, pp. 33–36. (in Russian)
9. *Voblyy V. A.* Perechislenie pomechennykh neplanarnykh pentatsiklicheskikh blokov [Enumeration of labeled nonplanar pentacyclic blocks]. Itogi Nauki i Tekhn. Sovrem. Matem. i ee Prilozh. Temat. Obzory, 2021, vol. 193, pp. 28–32. (in Russian)
10. *Harary F.* Graph Theory. CRC Press, 1994. 288 p.
11. *Harary F. and Palmer E. M.* Graphical Enumeration. N.Y., London, Academic Press, 1973. 286 p.
12. *McDiarmid C. and Scott A.* Random graphs from a block stable class. Europ. J. Combin., 2016, vol. 58, pp. 96–106.
13. *Voblyy V. A.* Ob odnom podkhode k perechisleniyu pomechennykh svyaznykh grafov: obzor rezul'tatov [On an approach to enumeration of labeled connected graphs: A review]. Itogi Nauki i Tekhn. Sovrem. Matem. i ee Prilozh. Temat. Obzory, 2020, vol. 188, pp. 106–118. (in Russian)
14. *Voblyy V. A.* Perechislenie pomechennykh eylerovykh pentatsiklicheskikh grafov [Enumeration of labeled Eulerian pentacyclic graphs]. Prikladnaya Diskretnaya Matematika, 2020, no. 50, pp. 87–92. (in Russian)
15. *Goulden I. P. and Jackson D. M.* Combinatorial Enumeration. Dover, 2004. 608 p.
16. *Riordan D.* Combinatorial Identities. N.Y., Wiley, 1968. 256 p. (in Russian)
17. *Wright E. M.* The number of connected sparsely edged graphs. J. Graph Theory, 1977, vol. 1, no. 4, pp. 317–330.
18. *Voblyy V. A.* Asimptoticheskoe perechislenie pomechennykh posledovatel'no-parallel'nykh tetratsiklicheskikh grafov [Asymptotical enumeration of labeled series-parallel tetracyclic graphs]. Itogi Nauki i Tekhn. Sovrem. Matem. i ee Prilozh. Temat. Obzory, 2020, vol. 187, pp. 31–35. (in Russian)
19. *Voblyy V. A.* Enumeration of labeled connected graphs with given order and number of edges. J. Appl. Industr. Math., 2016, vol. 10, no. 2, pp. 302–310.
20. *Dmitriev E. F.* Perechislenie otmechennykh grafov s dannymi strukturnymi svoystvami [Enumeration of marked graphs with given structural properties]. PhD thesis. Institute of Mathematics of the Academy of Sciences of the BSSR, Minsk, 1986. (in Russian)

УДК 519.1, 004.05 DOI 10.17223/20710410/59/5

АТТРАКТОРЫ И ЦИКЛИЧЕСКИЕ СОСТОЯНИЯ В КОНЕЧНЫХ ДИНАМИЧЕСКИХ СИСТЕМАХ ОРИЕНТАЦИЙ ПОЛНЫХ ГРАФОВ

А. В. Жаркова

Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, г. Саратов, Россия

E-mail: ZharkovaAV3@gmail.com

Рассматривается конечная динамическая система, состояниями которой являются все возможные ориентации данного полного графа, а эволюционная функция задаётся следующим образом: динамическим образом орграфа является орграф, полученный из исходного путём переориентации всех дуг, входящих в стоки, других отличий между исходным орграфом и его образом нет. Характеризуются циклические состояния системы (принадлежащие аттракторам), приводится таблица с количеством циклических состояний и состояний, не являющихся циклическими, в системах ориентаций полных графов с количеством вершин от 1 до 8 включительно. Описывается формирование аттракторов системы, их вид, длина, приводится таблица с соответствующим количеством аттракторов в системах ориентаций полных графов с количеством вершин от 1 до 8 включительно.

Ключевые слова: *аттрактор, граф, информационная безопасность, кибербезопасность, конечная динамическая система, ориентированный граф, отказоустойчивость, полный граф, циклическое состояние, эволюционная функция.*

ATTRACTORS AND CYCLIC STATES IN FINITE DYNAMIC SYSTEMS OF COMPLETE GRAPHS ORIENTATIONS

A. V. Zharkova

Saratov State University, Saratov, Russia

Graph models occupy an important place in information security tasks. Finite dynamic systems of complete graphs orientations are considered. States of a dynamic system (Γ_{K_n}, α), $n \geqslant 1$, are all possible orientations of a given complete graph K_n, and the evolutionary function transforms the complete graph orientation by reversing all arcs that go into sinks, and there are no other differences between the given and the next digraphs. The cyclic (belonging to attractors) states of the system are characterized, namely, state belongs to an attractor, if and only if it hasn't a sink or its indegrees vector is $(n-1, n-2, \ldots, 0)$. A table is given with the number of cyclic and non-cyclic states in the systems of complete graphs orientations with the number of vertices from 1 to 8 inclusive. The formation of attractors of the system, their type and length are described, namely, there are attractors of length 1, each of which is formed by state without sink, and attractors of length n, each of which is formed by such states $\overrightarrow{G} \in \Gamma_{K_n}$, in which indegrees vector is $(n-1, n-2, \ldots, 0)$, wherein each such attractor represents a circuit, in which each next state is obtained from the previous one as follows: if $\overrightarrow{G}$ has vector of indegrees of its vertices in the order of their enumeration $(d^-(v_1), d^-(v_2), \ldots, d^-(v_n))$, then $\alpha(\overrightarrow{G}) \in \Gamma_{K_n}$ has vector of indegrees of

its vertices in the order of their enumeration $(d^-(v_1)+1, d^-(v_2)+1, \ldots, d^-(v_n)+1)$, where the addition is calculated modulo n, and only these attractors. Note that in the considered finite dynamic systems the number of attractors of length n is equal to $(n-1)!$ and the number belonging to the attractors states is equal to $n!$. A table is given with the corresponding number of attractors in the systems of complete graphs orientations with the number of vertices from 1 to 8 inclusive.

Keywords: *attractor, complete graph, cybersecurity, cyclic state, directed graph, evolutionary function, fault-tolerance, finite dynamic system, graph, information security.*

Введение

Графовые модели занимают важное место в задачах, связанных с информационной безопасностью. В вопросах кибербезопасности с помощью графовых моделей можно, например, выявлять связи между сущностями системы, группировать их, оценивать их поведение, выявлять различные аномалии. В задачах, связанных с отказоустойчивостью компьютерных сетей, отказы процессоров интерпретируются как удаление соответствующих вершин, а отказы сетевых каналов — как удаление дуг. При изучении модельных графов можно применять идеи и методы теории конечных динамических систем [1–3]. В модели [1] в качестве механизма восстановления работоспособности сети предлагается так называемая SER-динамика бесконтурных связных ориентированных графов. В настоящей работе полные графы изучаются с точки зрения динамического подхода к кибербезопасности и отказоустойчивости графовых систем.

1. Основные определения и постановка задачи

Под *ориентированным графом* (или, для краткости, *орграфом*) понимается пара $\overrightarrow{G} = (V, \beta)$, где V — конечное непустое множество (*вершины* орграфа), а $\beta \subseteq V \times V$ — отношение на множестве V (пара $(u, v) \in \beta$ называется *дугой* орграфа с *началом* u и *концом* v). Отношение β называют *отношением смежности*. Отсутствие *петель* (дуг с совпадающими началом и концом) в орграфе $\overrightarrow{G} = (V, \beta)$ означает антирефлексивность его отношения смежности β. *Неориентированным графом* (или, для краткости, *графом*) называется пара $G = (V, \beta)$, где β — симметричное и антирефлексивное отношение на множестве вершин V. Дуги неориентированного графа называют *рёбрами*. Орграф $\overrightarrow{G} = (V, \beta)$ называется *направленным графом* (или *диграфом*), если отношение α антисимметрично. *Степенью исхода* вершины $v \in V$ называется число $d^+(v)$ дуг орграфа $\overrightarrow{G} = (V, \beta)$, имеющих своим началом v. *Степень захода* вершины v — это количество $d^-(v)$ дуг, имеющих v своим концом. Орграф называется *функциональным*, если $d^+(v) = 1$ для любой его вершины v. Граф $G = (V, \beta)$ называется *полным*, если любые две его вершины соединены ребром. Полный граф с n вершинами обозначается символом K_n. Вершины u и v графа G называются *связанными*, если в G существует проходящий через них путь. Отношение связанности является эквивалентностью на множестве вершин графа. Классы этого отношения называются *компонентами связности* (или просто *компонентами*) графа. Маршрут, в котором никакая дуга не встречается более одного раза, называется *путём*. Путь, каждая вершина которого принадлежит не более чем двум его дугам, является *простым*. Простой циклический путь в орграфе называется *контуром*. Говорят, что вершина v *достижима* из вершины u, если в орграфе существует путь из u в v. Вершина орграфа, недостижимая из других его вершин, называется *источником*, а вершина, из которой не достижима никакая другая вершина, — *стоком* [4].

Под *конечной динамической системой* понимается пара $(S,\ \delta)$, где S — конечное непустое множество, элементы которого называются *состояниями системы*; $\delta : S \to S$ — отображение множества состояний в себя, называемое *эволюционной функцией системы*. Каждой конечной динамической системе сопоставляется карта, представляющая собой функциональный орграф с множеством вершин S и дугами, проведёнными из каждой вершины $s \in S$ в вершину $\delta(s)$. Компоненты связности орграфа, задающего динамическую систему, называются её *бассейнами*. Каждый бассейн представляет собой контур с входящими в него деревьями. Контуры, в свою очередь, называются предельными циклами, или *аттракторами*. Состояние, принадлежащее аттрактору, называется *циклическим*.

Основными проблемами теории конечных динамических систем являются задачи отыскания эволюционных параметров системы без проведения динамики. К числу таких характеристик относятся принадлежность состояния аттрактору, описание аттракторов системы. Автором описаны свойства принадлежности состояний аттракторам, сами аттракторы конечных динамических систем ориентаций некоторых типов графов (например, [5, 6]). В данной работе приводится критерий принадлежности состояний аттракторам в конечных динамических системах ориентаций полных графов, описывается формирование аттракторов в данных системах, их вид, длина.

2. Описание конечной динамической системы (Γ_{K_n}, α)

Пусть дан полный граф $G = K_n$, $n \geqslant 1$, $m = n(n-1)/2$ — число рёбер. Пометим вершины и придадим рёбрам произвольную ориентацию, тем самым получив направленный граф $\overrightarrow{G} = (V, \beta)$, где отношение смежности β антирефлексивно и антисимметрично. Применим к полученному орграфу эволюционную функцию α, которая одновременно переориентирует все дуги, входящие в стоки, а остальные дуги оставляет без изменения, в результате получим орграф $\alpha(\overrightarrow{G})$. Если проделать указанные действия со всеми возможными ориентациями данного графа, то получим карту конечной динамической системы, состоящую из одного или нескольких бассейнов.

Таким образом, будем рассматривать конечную динамическую систему (Γ_{K_n}, α), $n \geqslant 1$, где через Γ_{K_n} обозначим множество всех возможных ориентаций данного полного графа K_n, $|\Gamma_{K_n}| = 2^m$, а эволюционная функция α задаётся следующим образом: если дан некоторый орграф $\overrightarrow{G} \in \Gamma_{K_n}$, то его динамическим образом $\alpha(\overrightarrow{G})$ является орграф, полученный из $\overrightarrow{G}$ одновременной переориентацией всех дуг, входящих в стоки, других отличий между $\overrightarrow{G}$ и $\alpha(\overrightarrow{G})$ нет.

На рис. 1 представлен граф K_3 и карта конечной динамической системы (Γ_{K_3}, α).

В работе [1] рассматривается конечная динамическая система (Ω, α), где Ω — множество всех бесконтурных ориентаций данного связного графа, и замечается, что для полного графа существует $n!$ бесконтурных ориентаций, где $n!$ — количество перестановок его вершин, при этом система имеет $(n-1)!$ бассейнов, каждый из которых состоит исключительно из аттрактора длины n, то есть все состояния данной системы являются циклическими.

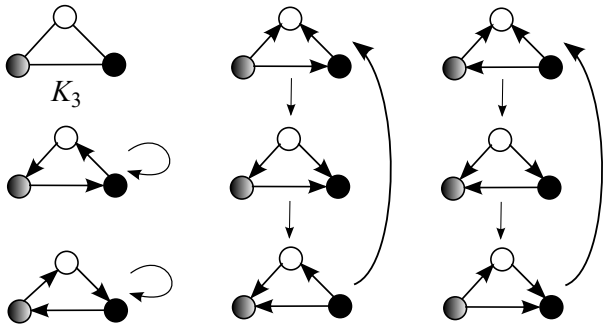


Рис. 1. Граф K_3 и карта конечной динамической системы (Γ_{K_3}, α)

3. Циклические состояния конечной динамической системы (Γ_{K_n}, α)

Определение 1. Под *вектором степеней захода* орграфа будем понимать вектор, компонентами которого являются расположенные в убывающем порядке степени захода всех его вершин.

Например, на рис. 1 расположенный сверху слева орграф имеет вектор степеней захода $(1, 1, 1)$.

Теорема 1. В конечной динамической системе (Γ_{K_n}, α), $n \geqslant 1$, состояние $\overrightarrow{G} \in \in \Gamma_{K_n}$ принадлежит аттрактору (является циклическим) тогда и только тогда, когда орграф $\overrightarrow{G}$

1) не имеет стока

или

2) имеет вектор степеней захода $(n-1, n-2, \ldots, 0)$.

Доказательство.

Необходимость. Пусть состояние $\overrightarrow{G} \in \Gamma_{K_n}$ конечной динамической системы (Γ_{K_n}, α), $n \geqslant 1$, принадлежит аттрактору, то есть является циклическим. Очевидно, что в орграфе $\overrightarrow{G}$ может быть не более одного стока. Рассмотрим циклические состояния системы в зависимости от наличия в них стока.

1) В орграфе $\overrightarrow{G}$ нет стока.

Согласно заданной эволюционной функции α, получаем, что $\alpha(\overrightarrow{G}) = \overrightarrow{G}$, тем самым состояние $\overrightarrow{G}$ образует аттрактор единичной длины и, действительно, является циклическим.

2) В орграфе $\overrightarrow{G}$ есть сток.

При $n = 2$ система (Γ_{K_2}, α) имеет два состояния, у каждого есть сток, и они образуют аттрактор длины 2, то есть являются циклическими, при этом оба состояния имеют вектор степеней захода $(1, 0)$.

Пусть $n \geqslant 3$ и стоком является вершина v_s, то есть $d^-(v_s) = n - 1$. Покажем, что орграф $\overrightarrow{G}$ имеет вектор степеней захода $(n-1, n-2, \ldots, 0)$.

Предположим, что это не так. Допустим, вершины v_k и v_l имеют одинаковую степень захода: $d^-(v_k) = d^-(v_l) = p$. Заметим, что $p \neq 0$ и $p \neq n-1$, так как иначе в орграфе $\overrightarrow{G}$ было бы несколько источников или стоков, что невозможно.

Применим эволюционную функцию к состоянию $\overrightarrow{G}$, получим $\alpha(\overrightarrow{G}) = \overrightarrow{G}'$, у которого $d^-(v'_s) = 0$ (для наглядности добавим $'$ к соответствующим вершинам), так как сток станет источником; $d^-(v'_k) = d^-(v'_l) = p+1$, так как v_k и v_l не являлись стоками и только единственные инцидентные им и вершине v_s дуги поменяли своё направление,

тем самым увеличив их степень захода. Заметим, что $p \neq n-2$, иначе в $\overrightarrow{G}'$ вершины v_k' и v_l' стали бы двумя стоками, что невозможно. Таким образом, $1 \leqslant p \leqslant n-3$.

Так как состояние $\overrightarrow{G}$ принадлежит аттрактору, то состояние $\overrightarrow{G}'$ также принадлежит этому же аттрактору, значит, в $\overrightarrow{G}'$ есть сток v_o', то есть $d^-(v_o') = n-1$, при этом в $\overrightarrow{G}$ $d^-(v_o) = n-2$. Получаем, что орграф $\overrightarrow{G}$ имеет вектор степеней захода $(n-1, n-2, \ldots, p, p, \ldots)$ и соответственно орграф $\overrightarrow{G}'$ имеет вектор степеней захода $(n-1, \ldots, p+1, p+1, \ldots, 0)$.

Продолжая аналогичные рассуждения и учитывая, что все состояния, получаемые при применении к текущему орграфу эволюционной функции, являются циклическими, дойдём до состояния, в котором у соответствующих вершин v_k и v_l степень захода равна $n-1$, то есть в нём окажется два стока, что невозможно. Получили противоречие. Таким образом, циклическое состояние системы, в орграфе которого есть сток, имеет вектор степеней захода $(n-1, n-2, \ldots, 0)$.

Достаточность.

1) Пусть в орграфе $\overrightarrow{G}$ нет стока, тогда аналогично п. 1 в доказательстве необходимости получаем, что данное состояние является циклическим.

2) Пусть орграф $\overrightarrow{G}$ имеет вектор степеней захода $(n-1, n-2, \ldots, 0)$, покажем, что данное состояние является циклическим.

При $n = 2$ система (Γ_{K_2}, α) имеет два состояния, каждое имеет вектор степеней захода $(1, 0)$, и они образуют аттрактор длины 2, то есть являются циклическими.

Пусть $n \geqslant 3$.

а) Пусть $\overrightarrow{G}_0 = \overrightarrow{G}$, а далее $\overrightarrow{G}_i = \alpha^i(\overrightarrow{G})$, $i \geqslant 1$.

Не теряя общности, пусть вектор степеней захода в орграфе $\overrightarrow{G_0}$ и вектор, составленный из степеней захода его вершин в порядке их нумерации, совпадают и равны $(n-1, n-2, \ldots, 0)$. Тогда у орграфа $\overrightarrow{G}_i$ вектор, составленный из степеней захода его вершин в порядке их нумерации, равен $(n-1+i, n-2+i, \ldots, 0+i)$, где сложение осуществляется по модулю n.

Получаем, что у орграфа $\overrightarrow{G}_n = \alpha^n(\overrightarrow{G}_0)$ вектор, составленный из степеней захода его вершин в порядке их нумерации, и вектор степеней захода совпадают и равны $(n-1, n-2, \ldots, 0)$, то есть равны соответствующим векторам орграфа $\overrightarrow{G} = \overrightarrow{G}_0$.

б) Покажем по построению, что $\overrightarrow{G}_0 = \overrightarrow{G_n}$, то есть образуется аттрактор.

Построим последовательно ориентацию графа, у которой вектор, составленный из степеней захода его вершин в порядке их нумерации, и вектор степеней захода есть $(n-1, n-2, \ldots, 0)$.

У вершины v_n степень захода равна 0: $d^-(v_n) = 0$, то есть она является источником, все рёбра ориентируем из неё.

У вершины v_{n-1} степень захода равна 1: $d^-(v_{n-1}) = 1$, то есть она достижима только из одной вершины, а именно из v_n, все остальные рёбра ориентируем из неё.

У вершины v_{n-2} степень захода равна 2: $d^-(v_{n-2}) = 2$, то есть она достижима только из двух вершин, а именно из v_n и v_{n-1}, все остальные рёбра ориентируем из неё.

Продолжая аналогично, доходим до вершины v_1, у которой степень захода равна $n-1$: $d^-(v_1) = n-1$, то есть она является стоком, и на этом шаге все рёбра уже ориентированы в данную вершину.

Таким образом, получаем единственную ориентацию графа, у которой вектор, составленный из степеней захода его вершин в порядке их нумерации, и вектор степеней захода есть $(n-1, n-2, \ldots, 0)$, то есть $\overrightarrow{G}_0 = \overrightarrow{G}_n$.

Таким образом, орграф $\overrightarrow{G}$ с вектором степеней захода $(n-1, n-2, \ldots, 0)$ принадлежит аттрактору, то есть является циклическим. ■

Например, в конечной динамической системе (Γ_{K_3}, α) (см. рис. 1) все восемь состояний системы являются циклическими, при этом два состояния не имеют стока и шесть состояний имеют вектор степеней захода $(2, 1, 0)$.

В табл. 1 приведены данные по количеству принадлежащих и не принадлежащих аттракторам состояний в конечных динамических системах (Γ_{K_n}, α) для $1 \leqslant n \leqslant 8$, полученные с помощью вычислительных экспериментов.

Таблица 1

Количество циклических состояний (ЦС) и состояний, не являющихся циклическими (НЦС), в конечных динамических системах (Γ_{K_n}, α)

n	Количество состояний	Количество ЦС	%	Количество НЦС	%
1	1	1	100	0	0
2	2	2	100	0	0
3	8	8	100	0	0
4	64	56	87,5	8	12,5
5	1024	824	$\approx 80{,}5$	200	$\approx 19{,}5$
6	32768	27344	$\approx 83{,}4$	5424	$\approx 16{,}6$
7	2097152	1872816	$\approx 89{,}3$	224336	$\approx 10{,}7$
8	268435456	251698560	$\approx 93{,}8$	16736896	$\approx 6{,}2$

Можно заметить, что количество циклических состояний в конечных динамических системах (Γ_{K_n}, α) составляет абсолютное большинство по сравнению с состояниями, не являющимися циклическими.

4. Аттракторы конечной динамической системы (Γ_{K_n}, α)

Теорема 2. В конечной динамической системе (Γ_{K_n}, α), $n \geqslant 1$, существуют следующие аттракторы:

1) длины 1, каждый из которых образован состоянием $\overrightarrow{G} \in \Gamma_{K_n}$, у которого нет стока;
2) длины n, каждый из которых состоит из состояний $\overrightarrow{G} \in \Gamma_{K_n}$, у которых вектор степеней захода есть $(n-1, n-2, \ldots, 0)$, при этом аттрактор представляет собой контур, в котором каждое следующее состояние получается из предыдущего таким образом: если $(d^-(v_1), d^-(v_2), \ldots, d^-(v_n))$ — вектор, составленный из степеней захода вершин в порядке их нумерации для $\overrightarrow{G}$, то для $\alpha(\overrightarrow{G}) \in \Gamma_{K_n}$ соответствующий вектор равен $(d^-(v_1)+1, d^-(v_2)+1, \ldots, d^-(v_n)+1)$, где сложение осуществляется по модулю n,

и только они.

Доказательство. Рассмотрим конечную динамическую систему (Γ_{K_n}, α), $n \geqslant 1$. Состояниями системы являются все возможные ориентации полного графа K_n. Очевидно, что в данных орграфах может быть не более одного стока. Рассмотрим состояния системы в зависимости от наличия в них стока.

1) У состояния нет стока.

Применим к такому состоянию $\overrightarrow{G}$ эволюционную функцию α, получим $\alpha(\overrightarrow{G}) = \overrightarrow{G}$, тем самым состояние $\overrightarrow{G}$ образует аттрактор единичной длины.

2) У состояния есть сток.

Согласно теореме 1, принадлежащие аттракторам состояния имеют вектор степеней захода $(n-1, n-2, \ldots, 0)$.

Рассмотрим произвольное циклическое состояние $\overrightarrow{G}$, перенумеруем его вершины таким образом, чтобы его вектор степеней захода и вектор, составленный из степеней захода его вершин в порядке нумерации, совпадали.

Пусть $\overrightarrow{G}_0 = \overrightarrow{G}$, а далее $\overrightarrow{G}_i = \alpha^i(\overrightarrow{G})$, $i \geqslant 1$. Орграф $\overrightarrow{G}_i$ имеет вектор, составленный из степеней захода его вершин в порядке их нумерации, $(n-1+i, n-2+i, \ldots, 0+i)$, где сложение осуществляется по модулю n, а также, согласно теореме 1, имеет вектор степеней захода $(n-1, n-2, \ldots, 0)$. Аналогично доказательству теоремы 1 получаем, что $\overrightarrow{G}_n = \overrightarrow{G}_0$, то есть образуется аттрактор длины n, который состоит из таких состояний $\overrightarrow{G} \in \Gamma_{K_n}$, у которых вектор степеней захода есть $(n-1, n-2, \ldots, 0)$, при этом аттрактор представляет собой контур, в котором каждое следующее состояние получается из предыдущего так: если $\overrightarrow{G}$ имеет вектор, составленный из степеней захода вершин в порядке нумерации, равный $(d^-(v_1), d^-(v_2), \ldots, d^-(v_n))$, то для $\alpha(\overrightarrow{G}) \in \Gamma_{K_n}$ вектор, составленный из степеней захода вершин в порядке нумерации, равен $(d^-(v_1)+1, d^-(v_2)+1, \ldots, d^-(v_n)+1)$. ■

Например, в конечной динамической системе (Γ_{K_3}, α) на рис. 1 есть два аттрактора длины 1, каждый из которых образован состоянием, у которого нет стока, и два аттрактора длины 3, каждый из которых состоит из состояний с вектором степеней захода $(2, 1, 0)$ и соответствующими векторами, составленными из степеней захода его вершин в порядке нумерации, например, для состояний расположенного посередине аттрактора имеем $(2, 1, 0) \to (0, 2, 1) \to (1, 0, 2) \to (2, 1, 0)$.

Заметим, что в конечной динамической системе (Γ_{K_n}, α), $n \geqslant 1$, количество аттракторов длины n равно $(n-1)!$, а количество состояний, принадлежащих данным аттракторам, равно $n!$.

Например, карта конечной динамической системы (Γ_{K_6}, α), $|\Gamma_{K_6}| = 32768$, состоит из 26 744 бассейнов, при этом 27 344 состояния принадлежат аттракторам (что составляет $\approx 83\,\%$ от общего числа состояний), которые образуют 26 624 аттрактора длины 1 и 120 аттракторов длины 6.

В табл. 2 приведены данные по количеству аттракторов в конечных динамических системах (Γ_{K_n}, α) для $1 \leqslant n \leqslant 8$, полученные с помощью вычислительных экспериментов.

Таблица 2

Количество аттракторов в конечных динамических системах (Γ_{K_n}, α)

n	Количество аттракторов (бассейнов)	Длины 1	Длины n
1	1	1	1
2	1	0	1
3	4	2	2
4	38	32	6
5	728	704	24
6	26744	26624	120
7	1868496	1867776	720
8	251663280	251658240	5040

Заключение

В работе приведён критерий принадлежности состояний аттракторам рассматриваемой конечной динамической системы (Γ_{K_n}, α), $n \geqslant 1$, всех возможных ориентаций графа K_n, описано формирование аттракторов в данных системах, их вид, длина, что является полезным для задач, связанных с информационной безопасностью, в том числе для построения отказоустойчивых графовых систем.

ЛИТЕРАТУРА

1. *Barbosa V. C.* An Atlas of Edge-Reversal Dynamics. London: Chapman & Hall/CRC, 2001. 372 p.
2. *Colon-Reyes O., Laubenbacher R., and Pareigis B.* Boolean monomial dynamical systems // Ann. Combinatorics. 2004. V. 8. P. 425–439.
3. *Салий В. Н.* Об одном классе конечных динамических систем // Вестник Томского госуниверситета. Приложение. 2005. № 14. С. 23–26.
4. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, Физматлит, 1997. 368 с.
5. *Власова А. В.* Аттракторы динамических систем, ассоциированных с циклами // Прикладная дискретная математика. 2011. № 2 (12). С. 90–95.
6. *Жаркова А. В.* Аттракторы в конечных динамических системах двоичных векторов, ассоциированных с ориентациями пальм // Прикладная дискретная математика. 2014. № 3 (25). С. 58–67.

REFERENCES

1. *Barbosa V. C.* An Atlas of Edge-Reversal Dynamics. London, Chapman & Hall/CRC, 2001. 372 p.
2. *Colon-Reyes O., Laubenbacher R., and Pareigis B.* Boolean monomial dynamical systems. Ann. Combinatorics, 2004, vol. 8, pp. 425–439.
3. *Salii V. N.* Ob odnom klasse konechnykh dinamicheskikh sistem [On a class of finite dynamic systems]. Vestnik TSU. Prilozheniye, 2005. no. 14, pp. 23–26. (in Russian)
4. *Bogomolov A. M. and Salii V. N.* Algebraicheskiye osnovy teorii diskretnykh sistem [Algebraic Foundations of the Theory of Discrete Systems]. Moscow, Nauka, Fizmatlit, 1997. 368 p. (in Russian)
5. *Vlasova A. V.* Attraktory dinamicheskikh sistem, assotsiirovannykh s tsiklami [Attractors of dynamic systems associated with cycles]. Prikladnaya Diskretnaya Matematika, 2011, no. 2 (12), pp. 90–95. (in Russian)
6. *Zharkova A. V.* Attraktory v konechnykh dinamicheskikh sistemakh dvoichnykh vektorov, assotsiirovannykh s oriyentatsiyami pal'm [Attractors in finite dynamic systems of binary vectors associated with palms orientations]. Prikladnaya Diskretnaya Matematika, 2014, no. 3 (25), pp. 58–67. (in Russian)

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.726 DOI 10.17223/20710410/59/6

АЛГОРИТМ ВАЛИДАЦИИ ОГРАНИЧЕННО-ДЕТЕРМИНИРОВАННОГО ПОВЕДЕНИЯ ПЕРЕДАТЧИКА В КАНАЛЕ ЧАСТИЧНОГО СТИРАНИЯ

И. Б. Казаков

Московский физико-технический институт, г. Долгопрудный, Россия

E-mail: i_b_kazakov@mail.ru

Понятия структуры частичного стирания и канала частичного стирания были введены в предшествующих работах автора. Также в данных работах представлена формальная модель взаимодействия приемника и передатчика. Введено понятие корректного протокола, т. е. понятие согласования поведения приемника с поведением передатчика. Найдено накладываемое на поведение передатчика необходимое и достаточное условие того, что существует согласованное с ним поведение приемника. В настоящей работе представлен алгоритм проверки указанного условия и оценка его сложности.

Ключевые слова: *скрытые каналы, структура частичного стирания, канал частичного стирания, протокол передачи информации, алгоритм проверки.*

A VALIDATION ALGORITHM OF THE TRANSMITTER'S BOUNDEDLY DETERMINISTIC BEHAVIOUR IN A PARTIAL ERASURE CHANNEL

I. B. Kazakov

Moscow Institute of Physics and Technology, Dolgoprudny, Russia

In the previous papers, the notions of a partial erasure structure and a partial erasure channel have been introduced. A partial erasure structure is a triplet consisting of an alphabet A, a family of partitions of the alphabet and a set of probabilities assigned to the partitions. A partial erasure channel functions as follows. Alice sends Bob a symbol $a \in A$, but Bob receives only partial information about the symbol. Bob only knows which partition has been choosen and which class of the partition the symbol belongs to. The set of pairs consisting of a partition and a class of the partition, i.e., the signals Bob can receive, is denoted as Bob's alphabet B. There is a natural correspondence between the characters sent by Alice and received by Bob. For symbols $a \in A$ and $b \in B$, $a \mapsto b$ is assumed to hold if there exists a partition T in the partial erasure structure such that $b = (\pi_T(a), T)$. The correspondence is also generalized to the words. We assume that $a_1 \dots a_n \mapsto b_1 \dots b_n$ holds, if $a_i \mapsto b_i$ holds for all $i = 1 \dots n$. Suppose that Alice has an input tape from which she reads symbols of some finite alphabet S. Bob has an output tape, on which he can print symbols of alphabet S and also specially reserved (not in S) erasure symbol "*". Thus, a problem

of transmitting information via the descibed channel (i.e., a problem of construction of Alice's and Bob's behaviour) arises. In the previous papers of the author, a formal model of interaction between a transmitter (Alice) and a receiver (Bob), called a protocol, is presented. A protocol is a pair of functions (F, G), where $F : S^* \to A^*$ is the Alice behaviour function and $G : B^* \to S \cup \{*, \Lambda\}$ is the Bob behaviour function. Suppose that Alice has just read another symbol $s \in S$ from the input tape, having previously read the word $\hat{s} = s_1, \ldots, s_m$. Then we assume that during the following $|F(\hat{s}s)|$ steps Alice sends the word $F(\hat{s}s)$ symbol by symbol via the channel of partial erasure. Bob receives the symbol $b \in B$ on each step. After receiving the symbol, he has to decide what to print on the output tape. He can print a symbol of the alphabet S, or the erasure symbol "*", or nothing. Bob's decision is determined by the function G. The protocol (F, G) is said to be valid, if Bob prints on the output tape the same content what originally is on the input tape, while, perhaps, with the replacement of the significant symbols (i.e., the symbols of the alphabet S) to the erasure symbol "*". Among all Alice's behavior functions, a class of correct functions is defined. A function F is said to be correct, if $F(\Lambda)F(s_1^1)F(s_1^1 s_2^1)\ldots F(s_1^1\ldots s_n^1) \mapsto \beta_1$, $F(\Lambda)F(s_1^2)F(s_1^2 s_2^2)\ldots F(s_1^2\ldots s_m^2) \mapsto \beta_2$ and β_1 is a prefix to β_2, where $\beta_1, \beta_2 \in B^*$. The question is investigated, for which functions of Alice's behavior F there exists a function of Bob's behavior G such that the pair (F, G) is a correct protocol. The theorem is proved that for this it is necessary and sufficient that F belongs to the class of correct functions. This paper presents an algorithm for verifying that Alice's behavior function F belongs to the class of correct functions. The complexity of the algorithm is $O(L|Q|^3|S|^4)$, where $|Q|$ is the number of states of the automaton representing the function F, L is the maximum length of the word Alice throws out, $|S|$ is the number of symbols of the alphabet S.

Keywords: *covert channels, partial erasure structure, partial erasure channel, information transmission protocol, check algorithm.*

Введение

В работе представлен алгоритм, который по формальному описанию поведения передатчика (традиционно называемого Алисой) в канале частичного стирания проверяет выполнение критериального условия существования согласованного с ним поведения приемника (традиционно называемого Бобом). Понятие канала частичного стирания, формальное описание поведений Алисы и Боба, а также постановка задачи проверки указанного условия представлены в предшествующих работах автора [1–3].

Изложим сведения, обосновывающие актуальность решения поставленной задачи. Предмет, исследуемый в работе, имеет отношение к скрытым каналам. Скрытый канал — это любой коммуникационный канал, передающий информацию не предназначенным для этого способом. Понятие скрытого канала впервые в открытой литературе введено в работе [4]. Современный краткий обзор, посвященный скрытым каналам, можно найти, например, в [5]. Особую важность имеют сетевые скрытые каналы, обзор и классификация техник построения которых имеется в [6].

Настоящая работа связана со скрытым каналом блужданий по плоскости. Функционирование этого канала может быть описано следующим образом: передающий участник (Алиса) определённым образом движется по плоскости, а принимающий (Боб) считывает из наблюдаемых перемещений передающего участника закодированную им информацию. Скрытый канал блужданий по плоскости может быть реализован в online-шутерах, то есть в многопользовательских играх, где есть так называемое

«игровое поле», по которому возможны перемещения. Задача построения скрытых каналов через online-шутеры ранее исследовалась, например, в [7].

В работе [2] рассматривается предположение о том, что Боб получает лишь часть информации о движении Алисы. Имеется в виду, что он «видит» Алису, т. е. определяет её местоположение не во все моменты игрового времени, а только в некоторые из них. Таким образом, если две траектории Алисы совпадают в обозначенные «моменты видимости», то с точки зрения Боба данные траектории являются неразличимыми. Следовательно, на множестве траекторий Алисы, начинающихся в фиксированной точке и имеющих фиксированную длину N, определено 2^N отношений эквивалентности, каждое из которых соответствует определённому выбору множества «моментов видимости». Каждому из 2^N возможных выборов «моментов видимости» также приписана вероятность того, что Боб «увидит» Алису именно в выбранные моменты.

Далее произведено абстрагирование от траекторий и от множеств моментов видимости. В результате получены понятия структуры частичного стирания и канала частичного стирания.

Структура частичного стирания — это тройка, состоящая из алфавита A, семейства определённых на данном алфавите разбиений и набора вероятностей, приписанных разбиениям. Канал частичного стирания функционирует следующим образом. Алиса отправляет Бобу символ $a \in A$, а Боб получает только часть информации: какое было выбрано разбиение и какому классу выбранного разбиения принадлежал отправленный символ. Множество пар, состоящих из разбиения и класса по данному разбиению, т. е. множество возможных ответов, которые может получить Боб, обозначается как алфавит Боба B.

Легко видеть, что ранее упомянутое множество траекторий Алисы вместе с соответствующими отношениями эквивалентности является частным случаем структуры частичного стирания. Алфавитом A является множество траекторий. Каждое из 2^N отношений эквивалентности, заданных на траекториях, определяет разбиение алфавита, т. е. имеется множество разбиений, содержащее 2^N элементов. Каждому разбиению приписана соответствующая вероятность.

Таким образом, возникает задача передачи информации по описанному каналу, т. е. задача построения соответствующей пары поведений Алисы и Боба. Полагаем, что у Алисы имеется входная лента, с которой она читает символы некоторого конечного алфавита S. У Боба, в свою очередь, имеется выходная лента, на которую он может печатать символы алфавита S, а также специально зарезервированный (не входящий в S) символ стирания «*». Общей целью как Алисы, так и Боба является отпечатывание на выходной ленте того же содержания, которое изначально имеется на входной, при этом, может быть, с заменой значащих символов (т. е. символов алфавита S) на символ стирания «*». На множестве S определён порядок символов: $S = \{s_1, \ldots, s_d\}$. Также порядок определён и на множестве пар символов алфавита S, т. е. на множестве S^2. Данный порядок является лексикографическим, т. е. $(s_1, s_2) < (s'_1, s'_2)$, если $s_1 < s'_1$ или $s_1 = s'_1$ и $s_2 < s'_2$.

В работах [2, 3] рассмотрены формализованные схемы организации передачи информации от Алисы к Бобу, названные протоколами передачи информации в канале частичного стирания. В [2] описан частный случай протокола, называемый «схемой равномерного кодирования». Протоколу в общем случае посвящена работа [3]. В соответствии с произведённой в [3] формализацией поведение Алисы может быть описано как функция $F : S^* \to A^*$, а поведение Боба — как функция $G : B^* \to S \cup \{*\}$, где A^*, B^*, S^* — множества слов, т. е. конечных последовательностей символов ал-

фавитов A, B, S соответственно. Протокол — это пара функций (F, G). Функция F однозначно определяет детерминированную функцию $\hat{F}$ следующим образом: $\hat{F}(\hat{s}) = \hat{F}(s_1 \dots s_m) = F(\Lambda)F(s_1)F(s_1 s_2) \dots F(s_1 \dots s_m)$, где Λ — пустое слово.

Представим содержательную интерпретацию, относящуюся к функциям поведения F, G. Пусть Алиса только что прочитала с входной ленты очередной символ $s \in S$, предварительно уже считав слово $\hat{s} = s_1 \dots s_m$. Тогда считаем, что на протяжении последующих $|F(\hat{s}s)|$ тактов Алиса посимвольно отправляет по каналу частичного стирания слово $F(\hat{s}s)$. Что касается Боба, то на каждом такте он получает некий символ $b \in B$. Получив указанный символ, он должен принять решение о том, что печатать на выходной ленте: или какой-нибудь символ из $S \cup \{*\}$, или ничего не печатать. Полагаем, что если $G(\beta) \in S \cup \{*\}$, то Боб печатает символ $G(\beta)$. Иначе, т. е. если $G(\beta) = \Lambda$, Боб ничего не печатает.

В соответствии с общей целью Алисы и Боба в работе [3] введено понятие корректного протокола. Основным является вопрос о том, для каких именно функций поведения Алисы F существует функция поведения Боба G, такая, что пара (F, G) является корректным протоколом. Ответом является теорема, что для этого необходимо и достаточно, чтобы F принадлежала к классу правильных функций. Понятие правильной функции и доказательство теоремы представлены в [3]. Очевидно, представляет интерес решение задачи проверки произвольной функции $F : S^* \to A^*$ на принадлежность к классу правильных. Настоящая работа посвящена этому алгоритму и оценке его сложности.

Отметим, что функция $\hat{F}$ представима в виде (не обязательно конечного) автомата, у которого S является входным алфавитом, а множество слов A^* — выходным: автомат на каждом шаге принимает символ $s \in S$ и выдаёт слово $\alpha \in A^*$. Действительно, рассмотрим бесконечный автомат, состояния которого находятся во взаимно-однозначном соответствии с элементами множества S^*. Находясь в состоянии $\hat{s} \in S^*$ и приняв символ $s \in S$, автомат переходит в состояние $\hat{s}s$ и выдаёт слово $F(\hat{s}s)$. Подвергая указанный автомат преобразованию, отождествляющему его неразличимые состояния, получаем приведённый автомат, представляющий функцию $\hat{F}$.

Входными данными алгоритма считаются функции переходов и выходов описанного автомата. Алгоритм принимает входные данные и за конечное время выдаёт ответ вида «да/нет» о принадлежности функции поведения Алисы F к классу правильных.

Поскольку за конечное время алгоритм не может обработать бесконечный объём входных данных, то имеет смысл рассматривать только ограниченно-детерминированные функции, т. е. такие, автомат которых имеет конечное число состояний.

Зафиксируем функцию F, такую, что соответствующая ей функция $\hat{F}$ является ограниченно-детерминированной. Соответственно фиксированным окажется конечный автомат, представляющий функцию $\hat{F}$. Множество состояний автомата обозначим через Q. Автомат имеет функцию переходов $\varphi : Q \times S \to Q$ и выходную функцию $\psi : Q \times S \to A^*$. Начальное состояние обозначим q_Λ. В качестве $q_{\hat{s}}$ будем обозначать состояние, в которое автомат приходит, приняв на вход слово $\hat{s} = s_1 \dots s_m$, т. е. последовательно приняв символы $s_1, \dots, s_m$. Таким образом, для любого слова $\hat{s} \in S^*$ и любого символа $s \in S$ выполняется $\psi(q_{\hat{s}}, s) = F(\hat{s}s)$. Значение $F(\Lambda)$, отсутствующее в описании автомата, также фиксировано и хранится в памяти отдельно.

Представим структуру дальнейшего изложения. В разд. 1 изложено проверяемое условие правильности, в разд. 2 представлен алгоритм валидации и оценка его сложности. Доказательствам посвящён разд. 3. В заключении подведены итоги и намечены направления дальнейших исследований.

1. Условие правильности

Приведём формальное определение проверяемого условия. Необходимые вспомогательные понятия, а также определение правильной функции представлены в п. 1.1. В п. 1.2 данное определение преобразовано в равносильную форму, непосредственно проверяемую алгоритмом.

1.1. Определения

Повторим представленные в работах [2, 3] и упомянутые во введении понятия, относящиеся к функционированию канала частичного стирания. Для дальнейшего изложения также имеют значение пары слов из A^*, такие, что Боб всегда сможет отличить факт отправки Алисой одного слова из пары от факта отправки Алисой другого слова из этой пары.

Определение 1. Структура частичного стирания — тройка $(A, \mathfrak{T}, \mathfrak{P})$, состоящая из алфавита A, непустого множества определённых на нем разбиений (отношений эквивалентности) $\mathfrak{T}$, а также приписанных данным разбиениям весов p_T, сумма которых равна 1.

Определение 2. Два символа $a_1, a_2 \in A$ абсолютно различимы в структуре частичного стирания $(A, \mathfrak{T}, \mathfrak{P})$, если в ней не найдётся разбиения, такого, что a_1, a_2 принадлежат одному классу этого разбиения. Будем говорить, что слова α_1, α_2 имеют общую абсолютно различимую позицию в структуре частичного стирания $(A, \mathfrak{T}, \mathfrak{P})$, если найдётся такое i, что символы $\alpha_1[i], \alpha_2[i]$ абсолютно различимы в структуре частичного стирания $(A, \mathfrak{T}, \mathfrak{P})$.

Алиса отправляет Бобу символ $a \in A$. Боб получает лишь часть информации об отправленном символе: выбранное отношение эквивалентности T, а также класс $\pi_T(a)$ по данному отношению. Множество всех пар $(\pi_T(a), T)$ обозначается как алфавит B Боба. Таким образом, между отправляемыми Алисой и получаемыми Бобом символами естественным образом определено вероятностное соответствие $a \underset{p_T}{\mapsto} b$. Поскольку в настоящей работе не рассматриваются вероятностные аспекты исследуемой модели, будем использовать обозначение $a \mapsto b$. Данное соответствие обобщается также на слова.

Определение 3. Для символов $a \in A$, $b \in B$ полагаем $a \mapsto b$, если в структуре частичного стирания имеется разбиение T, такое, что $b = (\pi_T(a), T)$. Для слов $\alpha \in A^*$, $\beta \in B^*$ полагаем $\alpha \mapsto \beta$, если $|\alpha| = |\beta| = l$ и для всех $i = 1, \ldots, l$ выполнено $\alpha(i) \mapsto \beta(i)$.

Определение 4. Для функции F определим функцию $\hat{F} : S^* \to A^*$ по следующему правилу: $\hat{F}(\hat{s}) = F(\Lambda)F(s_1)F(s_1 s_2)\ldots F(s_1 \ldots s_m)$, где $\hat{s} = s_1 \ldots s_m$.

Определение 5. Слово β_1 — префикс слова β_2, если β_2 имеет вид $\beta_2 = \beta_1\beta_1'$. Аналогично, β_1 — суффикс слова β_2, если β_2 имеет вид $\beta_2 = \beta_1'\beta_1$.

Определение 6. Функция F называется правильной, если из выполнения условий $\hat{F}(\hat{s}_1) \mapsto \beta_1$, $\hat{F}(\hat{s}_2) \mapsto \beta_2$ и β_1 — префикс β_2 следует выполнение условия $|\hat{s}_1| \leqslant |\hat{s}_2|$.

1.2. Равносильный вид

Преобразуем условие правильности таким образом, чтобы входящие в его формулировку понятия относились только к структуре частичного стирания и к функции поведения Алисы F. Для этого введём вспомогательные понятия аномальных пар слов из S^* 1-го и 2-го рода.

Определение 7. Пара слов $(\hat{s}_1, \hat{s}_2) \in (S^*)^2$ называется одноходовой, если выполнено $|\hat{s}_1| = |\hat{s}_2| + 1$.

Определение 8. Пара слов $(\hat{s}_1, \hat{s}_2) \in (S^*)^2$ называется аномалией 1-го рода для функции F, если слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$ имеют общую абсолютно различимую позицию.

Определение 9. Пара слов $(\hat{s}_1, \hat{s}_2) \in (S^*)^2$ называется аномалией 2-го рода для функции F, если $(|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|$ и $|\hat{s}_2| > |\hat{s}_1|)$ или $(|\hat{F}(\hat{s}_2)| \leqslant |\hat{F}(\hat{s}_1)|$ и $|\hat{s}_1| < |\hat{s}_2|)$.

Теорема 1. Для того чтобы функция F была правильной, необходимо и достаточно несуществования одноходовой пары, которая является аномалией 2-го рода, но не является аномалией 1-го рода для функции F.

Доказательство.

Необходимость. Предположим обратное: $(\hat{s}_1, \hat{s}_2)$ — аномалия 2-го рода для функции F, не являющаяся аномалией 1-го рода для функции F, т. е. $|\hat{s}_1| = |\hat{s}_2| + 1$, $|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|$, а слова $\hat{F}(\hat{s}_1), \hat{F}(\hat{s}_2)$ не имеют общей абсолютно различимой позиции.

Обозначим: $\alpha_1 = \hat{F}(\hat{s}_1)$, $\alpha_2 = \hat{F}(\hat{s}_2)$, $k_1 = |\alpha_1|$, $k_2 = |\alpha_2|$. Слова α_1, α_2 не имеют общей абсолютно различимой позиции и $k_1 \leqslant k_2$.

Так как слова α_1, α_2 не имеют общей абсолютно различимой позиции, можно выбрать k_1 разбиений $T_1, \ldots, T_{k_1}$ из структуры частичного стирания, таких, что при всех $i = 1, \ldots, k_1$ символы $\alpha_1(i), \alpha_2(i)$ принадлежат одному классу разбиения T_i, т. е. выполняется $\pi_{T_i}(\alpha_1(i)) = \pi_{T_i}(\alpha_2(i))$. Определим $b_i = (T_i, \pi_{T_i}(\alpha_1(i)))$; таким образом, при всех $i = 1, \ldots, k_1$ выполнено $\alpha_1(i), \alpha_2(i) \mapsto b_i$.

Выберем произвольным образом оставшиеся $k_2 - k_1$ разбиений $T_{k_1+1}, \ldots, T_{k_2}$. Определим $b_i = (T_i, \pi_{T_i}(\alpha_2(i)))$ при $i = k_1 + 1, \ldots, k_2$. Таким образом, при $i = k_1 + 1, \ldots, k_2$ выполняется $\alpha_2(i) \mapsto b_i$.

Положим $\beta_1 = b_1 \ldots b_{k_1}$, $\beta_2 = b_1 \ldots b_{k_1} b_{k_1+1} \ldots b_{k_2}$. Тогда $\hat{F}(\hat{s}_1) = \alpha_1 \mapsto \beta_1$, $\hat{F}(\hat{s}_2) = \alpha_2 \mapsto \beta_2$, β_1 — префикс β_2. Согласно определению правильной функции, отсюда следует, что выполняется $|\hat{s}_2| + 1 = |\hat{s}_1| \leqslant |\hat{s}_2|$. Противоречие.

Достаточность.

1. Предположим обратное: F не является правильной функцией. Это означает существование слов $\hat{s}_1, \hat{s}_2 \in S^*$ и $\beta_1, \beta_2 \in B^*$, таких, что β_1 — префикс β_2, $\hat{F}(\hat{s}_1) \mapsto \beta_1$, $\hat{F}(\hat{s}_2) \mapsto \beta_2$, $|\hat{s}_1| > |\hat{s}_2|$.

2. Отсюда следует, что слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$ не имеют общей абсолютно различимой позиции, т. е. пара $(\hat{s}_1, \hat{s}_2)$ не является аномалией 1-го рода для функции F.

3. Поскольку $|\hat{s}_1| > |\hat{s}_2|$, то можно записать $\hat{s}_1 = \hat{s}'_1 \hat{s}''_1$, где $|\hat{s}'_1| = |\hat{s}_2| + 1$. Отметим, что так как $\hat{F}(\hat{s}'_1)$ — префикс $\hat{F}(\hat{s}_1)$, то одноходовая пара $(\hat{s}'_1, \hat{s}_2)$ также не является аномалией 1-го рода для функции F.

4. Так как $\hat{s}'_1$ — префикс $\hat{s}_1$, то $\hat{F}(\hat{s}'_1)$ — префикс $\hat{F}(\hat{s}_1)$.

5. Определим слово β'_1 как префикс слова β_1, такой, что $|\beta'_1| = |\hat{F}(\hat{s}'_1)|$. Очевидно, что $\hat{F}(\hat{s}'_1) \mapsto \beta'_1$.

6. Итак, β'_1 — префикс β_1 (п. 1), β_1 — префикс β_2 (п. 5), следовательно, β'_1 — префикс β_2.

7. Таким образом, $|\hat{F}(\hat{s}'_1)| = |\beta'_1| \leqslant |\beta_2| = |\hat{F}(\hat{s}_2)|$. Следовательно, одноходовая пара $(\hat{s}'_1, \hat{s}_2)$ является аномалией 2-го рода для функции F.

8. Сопоставляя выводы п. 3 и 7, получаем противоречие.

Теорема 1 доказана. ■

Указанное в теореме 1 равносильное условие непосредственно проверяется алгоритмом.

2. Алгоритм валидации

Входными данными алгоритма являются функция перехода φ и выходная функция ψ конечного автомата, представляющего функцию $\hat{F}$. Функции заданы в виде таблиц. Отдельно в памяти хранится слово $F(\Lambda)$, хотя фактически оно не используется алгоритмом.

Алгоритм состоит из подготовительной и основной части. Описание подготовительной части алгоритма, называемого процедурой предынициализации, представлено в п. 2.1, основной части — в п. 2.3. Основная часть алгоритма работает с так называемыми перебираемыми сущностями, которые предварительно вводятся в п. 2.2.

2.1. Процедура предынициализации

До основной работы алгоритма выполняется процедура, модифицирующая входные данные, т. е. изменяющая функцию F таким образом, чтобы во время основной работы оказались выполнены ограничения $F(\Lambda) = \Lambda$ и $F(\hat{s}) \neq \Lambda$ при всех $\hat{s} \neq \Lambda$.

В первую очередь хранящееся в памяти значение $F(\Lambda)$ заменяется на пустое слово Λ, то есть функция F заменяется на функцию F_0, определяемую следующим образом: $F_0(\Lambda) = \Lambda$; $F_0(\hat{s}) = F(\hat{s})$, если $\hat{s} \neq \Lambda$. Эта замена корректна, так как F_0 правильна тогда и только тогда, когда правильна F. Данная операция требует времени $O(1)$.

Во вторую очередь проверяем, выполнено ли для всех слов $\hat{s} \neq \Lambda$ условие $F(\hat{s}) \neq \Lambda$, т. е. $\psi(q, s) \neq \Lambda$ для всех состояний $q \in Q$ представляющего автомата и всех символов $s \in S$. Указанная проверка выполняется за время, пропорциональное времени перечисления всех возможных переходов между состояниями, т. е. за время $O(|Q||S|)$. Если условие не выполнено, то функция F не является правильной и алгоритм возвращает ответ «нет» и завершается. Если условие выполнено, то алгоритм переходит к выполнению основной работы.

Теорема 2. Пусть функция F_0 определена следующим образом: $F_0(\Lambda) = \Lambda$; $F_0(\hat{s}) = F(\hat{s})$, если $\hat{s} \neq \Lambda$. Тогда функция F_0 правильна тогда и только тогда, когда правильна функция F.

Доказательство. Зафиксируем пару слов $(\hat{s}_1, \hat{s}_2) \in (S^*)^2$. Слова $\hat{F}_0(\hat{s}_1)$ и $\hat{F}_0(\hat{s}_2)$ имеют общую абсолютно различимую позицию тогда и только тогда, когда её имеют слова $\hat{F}(\hat{s}_1) = F(\Lambda)\hat{F}_0(\hat{s}_1)$, $\hat{F}(\hat{s}_2) = F(\Lambda)\hat{F}_0(\hat{s}_2)$. Равносильны следующие условия: $|\hat{F}(\hat{s}_1)| \leqslant |\hat{F}(\hat{s}_2)|$ и $|\hat{F}_0(\hat{s}_1)| \leqslant |\hat{F}_0(\hat{s}_2)|$, поскольку $|\hat{F}(\hat{s}_1)| = |\hat{F}_0(\hat{s}_1)| + |F(\Lambda)|$ и $|\hat{F}(\hat{s}_1)| = |\hat{F}_0(\hat{s}_1)| + |F(\Lambda)|$

Таким образом, пара слов $(\hat{s}_1, \hat{s}_2)$ является аномалией 2-го рода, но не является аномалией 1-го рода для функции F_0 в том и только в том случае, когда она является таковой для функции F. Применяя теорему 1, получаем требуемое. ■

Теорема 3. Пусть для некоторого слова $\hat{s} \in S^*$ выполнено $\hat{s} \neq \Lambda$ и $\hat{F}(\hat{s}) = \Lambda$. Тогда функция F не является правильной.

Доказательство. Предположим обратное: пусть F — правильная функция.

Так как $\hat{s} \neq \Lambda$, то существуют слово $\hat{s}' \in S^*$ и символ $s \in S$, такие, что $\hat{s} = \hat{s}'s$.

Условие теоремы означает, что $\alpha = \hat{F}(\hat{s}) = \hat{F}(\hat{s}'s) = \hat{F}(\hat{s}')F(\hat{s}'s) = \hat{F}(\hat{s}')F(\hat{s}) = \hat{F}(\hat{s}')\Lambda = \hat{F}(\hat{s}')$.

Обозначим $k = |\alpha|$. Выберем из структуры частичного стирания k произвольных разбиений $T_1, \ldots, T_k$ и положим $b_i = (T_i, \pi_{T_i}(\alpha(i)))$, $\beta = b_1 \ldots b_k$. Тогда $\alpha(i) \mapsto b_i$ и, следовательно, $\hat{F}(\hat{s}'s) = \hat{F}(\hat{s}') = \alpha \mapsto \beta$.

Таким образом, $\hat{F}(\hat{s}'s), \hat{F}(\hat{s}') \mapsto \beta$. Так как β — префикс самого себя, то, применяя определение правильной функции, получаем $|\hat{s}'| + 1 = |\hat{s}'s| \leqslant |\hat{s}'|$. Противоречие. ■

2.2. Перебираемые сущности

Алгоритм оперирует с пятёрками вида $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$, состоящими из двух состояний $q_1, q_2 \in Q$ представляющего функцию $\hat{F}$ автомата, слова α из множества A^*, а также двух слов $\hat{s}_1, \hat{s}_2$ из множества S^*. Такая пятёрка называется перебираемой сущностью.

Определение 10. Перебираемая сущность — это пятёрка $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$, где $q_1, q_2 \in Q$; $\alpha \in A^*$; $\hat{s}_1, \hat{s}_2 \in S^*$.

Определение 11. Будем говорить, что перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ имеет размер k, если $|\hat{s}_2| = k$. Размер перебираемой сущности e будем обозначать как $\mathrm{size}(e)$.

На множестве перебираемых сущностей определено действие пар символов (s_1, s_2) из S. Под действием указанной пары перебираемая сущность может переходить в аномалию 1-го рода, или в аномалию 2-го рода, или в другую перебираемую сущность.

Определение 12. Перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ под действием пары символов (s_1, s_2) переходит в аномалию 1-го рода, если слова $\alpha\psi(q_1, s_1)$ и $\psi(q_2, s_2)$ имеют общую абсолютно различимую позицию.

Определение 13. Перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ под действием пары символов (s_1, s_2) переходит в аномалию 2-го рода, если $|\alpha\psi(q_1, s_1)| \leqslant |\psi(q_2, s_2)|$.

Определение 14. Перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ под действием пары символов (s_1, s_2) переходит в перебираемую сущность $(q_1', q_2', \alpha', \hat{s}_1', \hat{s}_2')$, если она под действием этой пары не переходит ни в аномалию 1-го рода, ни в аномалию 2-го рода, а также выполнены следующие условия: $q_1' = \varphi(q_1, s_1)$, $q_2' = \varphi(q_2, s_2)$, $\hat{s}_1' = \hat{s}_1 s_1$, $\hat{s}_2' = \hat{s}_2 s_2$, α' — суффикс $\alpha\psi(q_1, s_1)$, $|\alpha'| = |\alpha\psi(q_1, s_1)| - |\psi(q_2, s_2)|$.

Определение 15. Будем говорить, что перебираемые сущности $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ и $(q_1', q_2', \alpha', \hat{s}_1', \hat{s}_2')$ подобны, если $q_1 = q_1'$, $q_2 = q_2'$ и $\alpha = \alpha'$.

Установим свойства подобных сущностей:

Утверждение 1. Пусть под действием пары символов (s_1, s_2) перебираемые сущности e_1, e_2 переходят в перебираемые сущности e_1', e_2' соответственно. Если сущности e_1, e_2 подобны, то подобны и сущности e_1', e_2'

Доказательство. Сущности e_1, e_2 подобны, то есть $e_1 = (q_1, q_2, \alpha, \hat{s}_{11}, \hat{s}_{12})$, $e_2 = (q_1, q_2, \alpha, \hat{s}_{21}, \hat{s}_{22})$. Под действием пары символов (s_1, s_2) сущности e_1, e_2 переходят в сущности $e_1' = (\varphi(q_1, s_1), \varphi(q_2, s_2), \alpha', \hat{s}_{11}s_1, \hat{s}_{12}s_2)$, $e_2' = (\varphi(q_1, s_1), \varphi(q_2, s_2), \alpha', \hat{s}_{11}s_1, \hat{s}_{12}s_2)$ соответственно, где α' — суффикс $\alpha\psi(q_1, s_1)$, и выполнено $|\alpha'| = |\alpha\psi(q_1, s_1)| - |\psi(q_2, s_2)|$. У сущностей e_1', e_2' совпадают первые три компоненты, т. е. они подобны. ■

Утверждение 2. Пусть e_1, e_2 — подобные перебираемые сущности, (s_1, s_2) — пара символов алфавита S. Тогда

1) Если под действием (s_1, s_2) сущность e_1 переходит в аномалию 1-го рода, то e_2 также переходит в аномалию 1-го рода.
2) Если под действием (s_1, s_2) сущность e_1 переходит в аномалию 2-го рода, то e_2 также переходит а аномалию 2-го рода.
3) Если под действием (s_1, s_2) сущность e_1 переходит в некую сущность e_1', то найдется сущность e_2', такая, что e_2 переходит в e_2' под действием (s_1, s_2).

Доказательство. Сущности e_1, e_2 подобны, то есть $e_1 = (q_1, q_2, \alpha, \hat{s}_{11}, \hat{s}_{12})$, $e_2 = (q_1, q_2, \alpha, \hat{s}_{21}, \hat{s}_{22})$.

1) Если под действием (s_1, s_2) сущность e_1 переходит в аномалию 1-го рода, то слова $\alpha\psi(q_1, s_1)$ и $\psi(q_2, s_2)$ имеют общую абсолютно различимую позицию. Тогда сущность e_2 также переходит в аномалию 1-го рода под действием (s_1, s_2).

2) Если под действием (s_1, s_2) сущность e_1 переходит в аномалию 2-го рода, то $|\alpha\psi(q_1, s_1)| \leqslant |\psi(q_2, s_2)|$. Тогда сущность e_2 также переходит в аномалию 2-го рода под действием (s_1, s_2).

3) Если под действием (s_1, s_2) сущность e_1 переходит в некую сущность e_1', то e_1 не переходит ни в аномалию 1-го рода, ни в аномалию 2-го рода под действием (s_1, s_2). Тогда слова $\alpha\psi(q_1, s_1)$ и $\psi(q_2, s_2)$ не имеют общей абсолютно различимой позиции и $|\alpha\psi(q_1, s_1)| > |\psi(q_2, s_2)|$. Следовательно, e_2 также не переходит ни в аномалию 1-го рода, ни в аномалию 2-го рода. Таким образом, найдётся сущность e_2', такая, что e_2 переходит в e_2' под действием (s_1, s_2).

Утверждение 2 доказано. ■

2.3. Описание основной работы алгоритма

Кроме входных данных, алгоритм хранит в памяти, во-первых, очередь перебираемых сущностей, которые предстоит обработать алгоритму. Во-вторых, в памяти хранится также множество перебираемых сущностей, которое далее называется множеством принятых к обработке. Тот факт, что конкретная перебираемая сущность в некоторый момент находится в этом множестве, означает, что к данному моменту алгоритм уже добавлял в очередь подобную ей перебираемую сущность.

Вызываемые алгоритмом 1 процедуры размещают в очереди перебираемые сущности. Разместить в очереди перебираемую сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ означает предварительно проверить, не находится ли подобная ей перебираемая сущность в множестве принятых к обработке. Если не находится, то сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ следует добавить в хвост очереди и в множество принятых к обработке. В ином случае ничего не следует делать.

Алгоритм 1. Основная часть алгоритма валидации

1: Выполнить процедуру инициализации:
2: **Для всех** символов $s \in S$, взятых в соответствующем порядке,
разместить в очереди инициальную перебираемую сущность $(q_s, q_\Lambda, \hat{F}(s), s, \Lambda)$.
3: **Пока** очередь не пуста и не выдано исключение:
брать с головы очереди перебираемую сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ и подавать её как аргумент на вход процедуры обработки:
4: **Для всех** пар символов $(s_1, s_2) \in S^2$, взятых в лексикографическом порядке:
5: **Если** перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ под действием пары (s_1, s_2) переходит в аномалию 1-го рода, **то**
6: пропустить выполнение последующих шагов цикла.
7: **Если** перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ под действием пары (s_1, s_2) переходит в аномалию 2-го рода, **то**
8: выдать исключение, т. е. завершить работу алгоритма и вернуть ответ «нет».
9: Для перебираемой сущности $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ найти, в какую другую перебираемую сущность $(q_1', q_2', \alpha', \hat{s}_1', \hat{s}_2')$ она переходит под действием пары символов (s_1, s_2). Такая сущность найдётся, так как не осуществляется перехода ни в аномалию 1-го, ни в аномалию 2-го рода.
10: Разместить в очереди перебираемую сущность $(q_1', q_2', \alpha', \hat{s}_1', \hat{s}_2')$.

Определение 16. Перебираемая сущность называется инициальной, если она имеет вид $(q_s, q_\Lambda, \hat{F}(s), s, \Lambda)$, где символ $s \in S$ рассматривается как слово длины 1.

Формально для алгоритма имеются три возможности: или он остановится в состоянии пустой очереди, или выдаст исключение, или будет работать бесконечно долго. Представляющий функцию F автомат имеет $|S|$ входных символов и $|Q|$ внутренних состояний. Значениями выходной функции являются слова из A^*. Обозначим максимальную длину слова из множества значений выходной функции через L.

Теорема 4. Возможность бесконечно долгой работы алгоритма исключена. Если F — правильная функция, то алгоритм остановится в состоянии пустой очереди, иначе он выдаст исключение. Сложность алгоритма составляет $O(L|Q|^3|S|^4)$.

3. Доказательство теоремы 4

В п. 3.1 определим вид полных перебираемых сущностей, которые могут быть поданы как аргумент в процедуру обработки. Далее в п. 3.2 установим такие свойства алгоритма, как «единственность обработки краткой сущности» и «упорядоченность по размеру». Условия выдачи алгоритмом исключения изучены в п. 3.3 и применены в п. 3.4 к аномалиям, в результате чего установлено, что исключение выдаётся тогда и только тогда, когда проверяемая функция не является правильной. В п. 3.5 показывается, что некоторые сущности не могут обрабатываться алгоритмом, тем самым обосновывается конечность времени его работы. Оценка сложности представлена в п. 3.6.

Отметим, что поскольку алгоритм вызывает процедуру предынициализации, то считаем, что на функцию F наложены соответствующие ограничения: $F(\Lambda) = \Lambda$, $F(\hat{s}) \neq \Lambda$ для всех $\hat{s} \neq \Lambda$.

3.1. Регулярные сущности

Определим вид перебираемых сущностей, которые могут быть поданы как аргумент в функцию обработки. Такие сущности назовём регулярными.

Определение 17. Будем говорить, что одноходовая пара слов $(\hat{s}_1, \hat{s}_2)$ из S^* не содержит префиксным образом аномалию 2-го рода для F, если никакая одноходовая пара $(\hat{s}'_1, \hat{s}'_2)$, такая, что $\hat{s}'_i$ — префикс $\hat{s}_i$, не является аномалией 2-го рода для F.

Определение 18. Перебираемая сущность $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$ называется регулярной, если выполнены следующие условия: $q_1 = q_{\hat{s}_1}$; $q_2 = q_{\hat{s}_2}$; $(\hat{s}_1, \hat{s}_2)$ — одноходовая пара, которая не является аномалией 1-го рода для функции F и не содержит префиксным образом аномалию 2-го рода для функции F; слово α — суффикс слова $\hat{F}(\hat{s}_1)$; $|\alpha| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|$.

Легко видеть, что понятия «инициальная сущность» и «регулярная сущность размера 0» тождественны.

Установим свойства регулярных сущностей, гарантирующие, что алгоритмом будут обрабатываться только регулярные сущности. Во-первых, инициальные сущности регулярны. Во-вторых, регулярная сущность, если не переходит в аномалию 1-го или 2-го рода, под действием любой пары символов (s_1, s_2) переходит в регулярную сущность. В-третьих, всякая неинициальная регулярная сущность имеет предшествующую регулярную сущность, т. е. ту, которая под действием некоторой пары (s_1, s_2) переходит в данную неинициальную регулярную сущность.

Утверждение 3. Пусть слова $\hat{s}'_1, \hat{s}'_2$ — префиксы слов $\hat{s}_1, \hat{s}_2$ соответственно и пара $(\hat{s}'_1, \hat{s}'_2)$ является аномалией 1-го рода для функции F. Тогда пара $(\hat{s}_1, \hat{s}_2)$ также является аномалией 1-го рода для функции F.

Доказательство. Так как $\hat{s}'_1$ — префикс $\hat{s}_1$, то $\hat{F}(\hat{s}'_1)$ — префикс $\hat{F}(\hat{s}_1)$. Аналогично, из того, что $\hat{s}'_2$ — префикс $\hat{s}_2$, следует, что $\hat{F}(\hat{s}'_2)$ — префикс $\hat{F}(\hat{s}_2)$.

Пусть слова $\hat{F}(\hat{s}'_1)$ и $\hat{F}(\hat{s}'_2)$ имеют общую абсолютно различимую позицию. Тогда данная позиция является также абсолютно различимой позицией слов $\hat{F}(\hat{s}'_1)$ и $\hat{F}(\hat{s}'_2)$. Следовательно, пара $(\hat{s}_1, \hat{s}_2)$ — аномалия 1-го рода для функции F. ■

Утверждение 4. Пусть слова $\hat{s}'_1, \hat{s}'_2$ — префиксы слов $\hat{s}_1, \hat{s}_2$ соответственно, пары слов $(\hat{s}'_1, \hat{s}'_2)$ и $(\hat{s}_1, \hat{s}_2)$ являются одноходовыми и пара $(\hat{s}_1, \hat{s}_2)$ не содержит префиксным образом аномалию 2-го рода для функции F. Тогда пара $(\hat{s}'_1, \hat{s}'_2)$ также не содержит префиксным образом аномалию 2-го рода для функции F.

Доказательство. Предположим обратное, т. е. что $(\hat{s}'_1, \hat{s}'_2)$ содержит префиксным образом аномалию 2-го рода для функции F. Тогда найдётся одноходовая пара слов $(\hat{s}''_1, \hat{s}''_2)$, являющаяся аномалией 2-го рода для функции F, такая, что $\hat{s}''_1$ — префикс $\hat{s}'_1$, $\hat{s}''_2$ — префикс $\hat{s}'_2$. Так как, в свою очередь, $\hat{s}'_1$ — префикс $\hat{s}_1$, то $\hat{s}''_1$ — префикс $\hat{s}_1$; аналогично, $\hat{s}''_2$ — префикс $\hat{s}_2$.

Из свойств пары $(\hat{s}''_1, \hat{s}''_2)$ следует, что пара $(\hat{s}_1, \hat{s}_2)$ содержит префиксным образом аномалию 2-го рода для функции F, что противоречит условию. ■

Утверждение 5. Пусть $(\hat{s}_1, \hat{s}_2)$ — одноходовая пара слов алфавита S, s_1, s_2 — символы алфавита S. Если пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 2-го рода для функции F, а пара $(\hat{s}_1, \hat{s}_2)$ не содержит префиксным образом аномалию 2-го рода для функции F, то пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ также не содержит префиксным образом аномалию 2-го рода для функции F.

Доказательство. Предположим обратное, т. е. что пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ содержит префиксным образом аномалию 2-го рода для функции F. Тогда найдётся одноходовая пара слов $(\hat{s}'_1, \hat{s}'_2)$, являющаяся аномалией 2-го рода для функции F, такая, что $\hat{s}'_1$ — префикс $\hat{s}_1 s_1$, а $\hat{s}'_2$ — префикс $\hat{s}_2 s_2$.

Предположим, что $\hat{s}'_2 = \hat{s}_2 s_2$. Тогда $|\hat{s}'_1| = |\hat{s}'_2| + 1 = |\hat{s}_2 s_2| + 1 = |\hat{s}_2| + 2 = |\hat{s}_1| + 1 = |\hat{s}_1 s_1|$. Так как $\hat{s}'_1$ — префикс $\hat{s}_1 s_1$, то отсюда следует $\hat{s}'_1 = \hat{s}_1 s_1$. Следовательно, пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ является аномалией 2-го рода для функции F, что противоречит условию. Таким образом, $\hat{s}'_2 \neq \hat{s}_2 s_2$.

Так как $\hat{s}'_2$ — префикс $\hat{s}_2 s_2$ и $\hat{s}'_2 \neq \hat{s}_2 s_2$, то $|\hat{s}'_2| < |\hat{s}_2 s_2|$. Отсюда следует $|\hat{s}'_1| = |\hat{s}'_2| + 1 < |\hat{s}_2 s_2| + 1 = |\hat{s}_2| + 2 = |\hat{s}_1| + 1 = |\hat{s}_1 s_1|$. Таким образом, $\hat{s}'_1 \neq \hat{s}_1 s_1$.

Так как $\hat{s}'_1$ — префикс $\hat{s}_1 s_1$ и $\hat{s}'_1 \neq \hat{s}_1 s_1$, то $\hat{s}'_1$ — префикс $\hat{s}_1$.

Так как $\hat{s}'_2$ — префикс $\hat{s}_2 s_2$ и $\hat{s}'_2 \neq \hat{s}_2 s_2$, то $\hat{s}'_2$ — префикс $\hat{s}_2$.

Из свойств пары $(\hat{s}'_1, \hat{s}'_2)$ следует, что пара $(\hat{s}_1, \hat{s}_2)$ содержит префиксным образом аномалию 2-го рода для функции F, что противоречит условию. ■

Утверждение 6. Инициальные перебираемые сущности, т. е. сущности вида $(q_s, q_\Lambda, F(s), s, \Lambda)$, $s \in S$, являются регулярными.

Доказательство.

1. Пара вида (s, Λ) не может быть аномалией 1-го рода для функции F, так как $\hat{F}(\Lambda) = F(\Lambda) = \Lambda$. Общих позиций у пустого слова Λ со словом $\hat{F}(s)$ нет, следовательно, не имеется и общих абсолютно различимых позиций.

2. Далее, $|\hat{F}(s)| = |F(\Lambda)F(s)| = |\Lambda F(s)| = |F(s)| > |\hat{F}(\Lambda)| = |F(\Lambda)| = 0$. Таким образом, пара (s, Λ) не является аномалией 2-го рода для функции F. Очевидно также, что данная пара не содержит префиксным образом аномалию 2-го рода для функции F, поскольку для пустого слова Λ не существует префиксов, отличных от самого пустого слова.

3. Слово $F(s)$ является суффиксом слова $\hat{F}(s) = F(\Lambda)F(s) = F(s)$. Запишем длину: $|F(s)| = |\hat{F}(s)| = |\hat{F}(s)| - 0 = |\hat{F}(s)| - |\hat{F}(\Lambda)|$.

Сопоставляя выводы п. 1–3, получаем, что перебираемая сущность $(q_s, q_\Lambda, F(s), s, \Lambda)$ регулярна. ■

Утверждение 7. Пусть перебираемая сущность $e_1 = (q_{\hat{s}_1}, q_{\hat{s}_2}, \alpha, \hat{s}_1, \hat{s}_2)$ регулярна и под действием пары символов (s_1, s_2) переходит в другую перебираемую сущность $e_2 = (q_{\hat{s}_1 s_1}, q_{\hat{s}_2 s_2}, \alpha', \hat{s}_1 s_1, \hat{s}_2 s_2)$. Тогда e_2 — регулярная сущность.

Доказательство. Выпишем некоторые условия регулярности сущности e_1. Во-первых, так как $(\hat{s}_1, \hat{s}_2)$ не является аномалией 1-го рода для функции F, слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$ не имеют общей абсолютно различимой позиции. Во-вторых, α — суффикс $\hat{F}(\hat{s}_1)$ и $|\alpha| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)| > 0$, поскольку $(\hat{s}_1, \hat{s}_2)$ не является аномалией 2-го рода для функции F. Следовательно, существует слово α_1, такое, что $|\alpha_1| = |\hat{F}(\hat{s}_2)|$ и $\hat{F}(\hat{s}_1) = \alpha_1\alpha$.

Пара $(\hat{s}_1, \hat{s}_2)$ является одноходовой. Следовательно, одноходовой является также пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$.

Проверим далее условия регулярности для сущности e_2.

1. $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не содержит префиксным образом аномалию для функции F:

— Под действием (s_1, s_2) сущность e_1 не переходит в аномалию 2-го рода для функции F, т. е. $|\alpha F(\hat{s}_1 s_1)| = |\alpha\psi(q_{\hat{s}_1}, s_1)| > |\psi(q_{\hat{s}_2}, s_2)| = |F(\hat{s}_2 s_2)|$. Следовательно, $|\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)| = |\alpha| > |F(\hat{s}_2 s_2)| - |F(\hat{s}_1 s_1)|$.

— Перенося слагаемые в неравенстве, получаем $|\hat{F}(\hat{s}_1 s_1)| = |\hat{F}(\hat{s}_1)| + |F(\hat{s}_1 s_1)| > |\hat{F}(\hat{s}_2)| + |F(\hat{s}_2 s_2)| = |\hat{F}(\hat{s}_2 s_2)|$. Таким образом, пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 2-го рода для функции F.

— Так как сущность e регулярна, пара $(\hat{s}_1, \hat{s}_2)$ не содержит префиксным образом аномалию 2-го рода для функции F. Кроме того, пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 2-го рода для функции F. В соответствии с утверждением 5 пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не содержит префиксным образом аномалию 2-го рода для функции F.

2. $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 1-го рода для функции F:

— Под действием пары символов (s_1, s_2) первая сущность не переходит в аномалию 1-го рода для функции F. Следовательно, слова $\alpha F(\hat{s}_1 s_1) = \alpha\psi(q_{\hat{s}_1}, s_1)$ и $F(\hat{s}_2 s_2) = \psi(q_{\hat{s}_2}, s_2)$ не имеют общей абсолютно различимой позиции.

— Предположим, что $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ является аномалией 1-го рода для функции F, т. е. слова $\hat{F}(\hat{s}_1 s_1) = \alpha_1\alpha F(\hat{s}_1 s_1)$ и $\hat{F}(\hat{s}_2 s_2) = \hat{F}(\hat{s}_2)F(\hat{s}_2 s_2)$ имеют общую абсолютно различимую позицию.

— Так как $|\alpha_1| = |\hat{F}(\hat{s}_2)|$, а слова $\alpha F(\hat{s}_1)$ и $F(\hat{s}_2 s_2)$ не имеют общей абсолютно различимой позиции, то общая абсолютно различимая позиция является общей абсолютно различимой позицией слов $\alpha_1, \hat{F}(\hat{s}_2)$.

— Следовательно, она же является абсолютно различимой позицией слов $\hat{F}(\hat{s}_1) = \alpha_1\alpha$, $\hat{F}(\hat{s}_2)$, что заведомо исключено.

3. Слово α' — суффикс $\hat{F}(\hat{s}_1 s_1)$ и $|\alpha'| = |\hat{F}(\hat{s}_1 s_1)| - |\hat{F}(\hat{s}_2 s_2)|$:

— По условию, сущность e_1 под действием (s_1, s_2) переходит в e_2. Следовательно, α' — суффикс $\alpha F(\hat{s}_1 s_1) = \alpha\psi(q_{\hat{s}_1}, s_1)$, $|\alpha'| = |\alpha F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)|$.

— α — суффикс $\hat{F}(\hat{s}_1)$. Следовательно, $\alpha F(\hat{s}_1 s_1)$ — суффикс $F(\hat{s}_1)F(\hat{s}_1 s_1) = \hat{F}(\hat{s}_1 s_1)$.

— α' — суффикс $\alpha F(\hat{s}_1 s_1)$, $\alpha F(\hat{s}_1 s_1)$ — суффикс $\hat{F}(\hat{s}_1 s_1)$. Следовательно, α' — суффикс $\hat{F}(\hat{s}_1 s_1)$.

— Подсчитаем длину: $|\alpha'| = |\alpha F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)| = |\alpha| + |F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)| + |F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)| = |\hat{F}(\hat{s}_1 s_1)| - |\hat{F}(\hat{s}_2 s_2)|$.

Утверждение 7 доказано. ■

Утверждение 8. Пусть e_1 — регулярная перебираемая сущность. Тогда найдутся регулярная сущность e_2 и пара символов (s_1, s_2) алфавита S, такие, что e_2 переходит в e_1 под действием (s_1, s_2).

Доказательство.

1. Сущность e_1 имеет вид $(q_{\hat{s}_1 s_1}, q_{\hat{s}_2 s_2}, \alpha', \hat{s}_1 s_1, \hat{s}_2 s_2)$, где s_1, s_2 — символы алфавита S.

2. Выпишем некоторые условия регулярности сущности e_1. Во-первых, слово α' является суффиксом слова $\hat{F}(\hat{s}_1 s_1)$ и $|\alpha'| = |\hat{F}(\hat{s}_1 s_1)| - |\hat{F}(\hat{s}_2 s_2)|$. Следовательно, найдётся слово α'_1, такое, что $|\alpha'_1| = |\hat{F}(\hat{s}_2 s_2)|$ и $\hat{F}(\hat{s}_1 s_1) = \alpha'_1 \alpha'$.

3. Во-вторых, пара слов $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 1-го рода для функции F, т. е. слова $\hat{F}(\hat{s}_1 s_1)$ и $\hat{F}(\hat{s}_2 s_2)$ не имеют общей абсолютно различимой позиции.

4. Пара слов $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ является одноходовой, т. е. $|\hat{s}_1 s_1| = |\hat{s}_2 s_2| + 1$. Следовательно, $|\hat{s}_1| = |\hat{s}_2| + 1$, т. е. $(\hat{s}_1, \hat{s}_2)$ также является одноходовой парой.

5. Так как $\hat{s}_1$ — префикс $\hat{s}_1 s_1$ и $\hat{s}_2$ — префикс $\hat{s}_2 s_2$, то в соответствии с утверждением 3 $(\hat{s}_1, \hat{s}_2)$ также не является аномалией 1-го рода для функции F.

6. Так как $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 2-го рода для функции F, то $|\hat{F}(\hat{s}_1 s_1)| > |\hat{F}(\hat{s}_2 s_2)|$. Следовательно, $|\alpha'| > 0$.

7. Так как пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не содержит префиксным образом аномалию 2-го рода для функции F, $\hat{s}_1$ — префикс $\hat{s}_1 s_1$, а $\hat{s}_2$ — префикс $\hat{s}_2 s_2$, то в соответствии с утверждением 4 пара $(\hat{s}_1, \hat{s}_2)$ также не содержит префиксным образом аномалию 2-го рода для функции F.

8. То, что $(\hat{s}_1, \hat{s}_2)$ — не аномалия 2-го рода для функции F, означает выполнение неравенства $|\hat{F}(\hat{s}_1)| > |\hat{F}(\hat{s}_2)|$. Следовательно, найдётся слово $\alpha \in A^*$, такое, что α — суффикс $\hat{F}(\hat{s}_1)$ и $|\alpha| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|$. Отсюда следует существование слова α_1, такого, что $|\alpha_1| = |\hat{F}(\hat{s}_2)|$ и $\hat{F}(\hat{s}_1) = \alpha_1 \alpha$.

9. Рассмотрим сущность $e_2 = (q_{\hat{s}_1}, q_{\hat{s}_2}, \alpha, \hat{s}_1, \hat{s}_2)$. Выводы п. 3, 4, 6 и 7 означают, что она регулярна. Докажем теперь, что под действием пары символов (s_1, s_2) она переходит в сущность e_1.

10. Перепишем тождества из п. 1 и 7: $\alpha'_1 \alpha' = \hat{F}(\hat{s}_1 s_1) = \hat{F}(\hat{s}_1) F(\hat{s}_1 s_1) = \alpha_1 \alpha F(\hat{s}_1 s_1)$. Таким образом, $|\alpha'| = |\alpha F(\hat{s}_1 s_1)| + |\alpha_1| - |\alpha'_1| = |\alpha F(\hat{s}_1 s_1)| + |\hat{F}(\hat{s}_2)| - |\hat{F}(\hat{s}_2 s_2)| = |\alpha F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)| < |\alpha F(\hat{s}_1 s_1)|$. Следовательно, α' — суффикс $\alpha F(\hat{s}_1 s_1)$.

11. Соотнося полученное в п. 9 выражение для $|\alpha'|$ с выводом п. 5, получаем $|\alpha\psi(q_{\hat{s}_1}, s_1)| = |\alpha F(\hat{s}_1 s_1)| > |F(\hat{s}_2 s_2)| = |\psi(q_{\hat{s}_2}, s_2)|$. Следовательно, сущность e_2 под действием (s_1, s_2) не переходит в аномалию 2-го рода.

12. Предположим, что e_2 под действием (s_1, s_2) переходит в аномалию 1-го рода, т. е. что слова $\alpha\psi(\hat{s}_1, s_1) = \alpha F(\hat{s}_1 s_1)$ и $\psi(q_{\hat{s}_2}, s_2) = F(\hat{s}_2 s_2)$ имеют общую абсолютно различимую позицию.

13. Так как $\hat{F}(\hat{s}_1 s_1) = \hat{F}(\hat{s}_1) F(\hat{s}_1 s_1) = \alpha_1 \alpha F(\hat{s}_1 s_1)$, $\hat{F}(\hat{s}_2 s_2) = \hat{F}(\hat{s}_2) F(\hat{s}_2 s_2)$ и $|\alpha_1| = |\hat{F}(\hat{s}_2)|$, то данная абсолютно различимая позиция является абсолютно различимой позицией слов $\hat{F}(\hat{s}_1 s_1)$ и $\hat{F}(\hat{s}_2 s_2)$, что противоречит выводу п. 2. Таким образом, e_2 не переходит в аномалию 1-го рода.

14. Соберём вместе выводы п. 9, 10 и 12: сущность e_2 под действием (s_1, s_2) не переходит ни в аномалию 1-го рода, ни в аномалию 2-го рода, α' является суффиксом $\alpha F(\hat{s}_1 s_1) = \alpha\psi(q_{\hat{s}_1}, s_1)$, $|\alpha'| = |\alpha F(\hat{s}_1 s_1)| - |F(\hat{s}_2 s_2)| = |\alpha\psi(q_{\hat{s}_1}, s_1)| - |\psi(q_{\hat{s}_2}, s_2)|$. Та-

ким образом, сущность e_2 под действием (s_1, s_2) переходит в сущность e_1, что и было заявлено в п. 8.

Утверждение 8 доказано. ■

3.2. Свойства обработки

Алгоритмом могут обрабатываться только регулярные сущности. Докажем, что каждая регулярная сущность может обрабатываться не более одного раза и что последовательность добавляемых в очередь сущностей упорядочена по размеру. Более того, по размеру упорядочена последовательность сущностей, подаваемых как аргумент в процедуру размещения в очереди.

Утверждение 9. Две подобные сущности не могут быть добавлены в очередь в разное время и, следовательно, не могут в разное время подаваться как аргумент в процедуру обработки.

Доказательство. Предположим обратное: пусть по ходу работы алгоритма две подобные сущности e_1 и e_2 добавлялись в очередь в разное время.

В очередь сущности попадают лишь в результате вызова процедуры размещения. Таким образом, сущности e_1 и e_2 подавались как аргумент на вход процедуры размещения в очереди и в результате её выполнений были добавлены в очередь. Для определённости предположим, что сущность e_1 размещалась ранее сущности e_2.

В соответствии с описанием процедуры размещения после завершения её работы на аргументе e_1 сущность e_1 находится в множестве принятых к обработке. Таким образом, она находится в множестве принятых к обработке во время работы процедуры размещения на аргументе e_2. Так как сущности e_1 и e_2 подобны, то e_2 не будет добавлена в очередь — противоречие. ■

Утверждение 10. В очередь алгоритма и в множество принятых к обработке могут попасть только регулярные сущности. Следовательно, все сущности, поступающие на вход процедуры обработки, регулярны.

Доказательство. После выполнения процедуры инициализации в очереди находятся инициальные сущности, которые регулярны согласно утверждению 6.

Всякая сущность, попадающая в очередь, является результатом действия пары (s_1, s_2) на сущность, которая была передана как аргумент в функцию обработки и, следовательно, сама попадала в очередь. Это действие, согласно утверждению 7, переводит регулярные сущности в регулярные. По индукции заключаем, что все попавшие в очередь сущности регулярны.

В множество принятых к обработке попадают те же сущности, что и в очередь. Следовательно, все они регулярны. ■

Утверждение 11. В каждый момент времени в очереди находятся или сущности одного и того же размера k, или сущности двух последовательных размеров k и $k+1$, причём очередь упорядочена по размеру, т. е. сущности размера $k+1$ находятся в очереди после сущностей размера k.

Доказательство. Во время выполнения процедуры инициализации и после её завершения находящиеся в очереди сущности имеют размер 0.

Пусть указанное в условии свойство выполнено перед обработкой очередной полной сущности с головы очереди. Докажем, что оно выполнено и после обработки данной сущности. По индукции заключаем, что это свойство выполняется всегда.

Действительно, пусть в очереди все сущности имеют размер k. Тогда, обрабатывая сущность с головы, процедура либо ничего не добавляет в очередь, либо добавляет в хвост очереди сущности размера $k+1$, что сохраняет выполнение свойства.

Если в очереди находятся сущности двух размеров k и $k+1$, то первой в очереди находится сущность размера k. Обрабатывая её, процедура, может быть, добавляет в хвост очереди сущности размера $k+1$, что также сохраняет выполнение свойства. ■

Утверждение 12. Пусть сущность e_1 размера k_1 и сущность e_2 размера k_2 попадают в очередь. Тогда если $k_1 < k_2$, то e_1 попадает в очередь ранее сущности e_2. И, следовательно, если обе сущности e_1 и e_2 подаются как аргумент в процедуру обработки, то сущность e_1 подаётся ранее сущности e_2.

Доказательство. Рассмотрим последовательность всех сущностей, которые попадают в очередь. Предположим, что в ней нарушен порядок по размеру. Не уменьшая общности, можно предположить, что порядок нарушен для двух соседних сущностей, т. е. имеются две сущности e_1 и e_2, такие, что $\mathrm{size}(e_1) > \mathrm{size}(e_2)$ и сущность e_2 была добавлена в очередь сразу вслед за сущностью e_1. Отметим, что сущности e_1 и e_2 не могут одновременно присутствовать в очереди, поскольку по утверждению 11 очередь упорядочена по размеру в каждый момент времени.

Предположим, что сущность e_2 добавляется в очередь во время выполнения процедуры инициализации. Тогда e_2 является инициальной сущностью и $\mathrm{size}(e_2) = 0$. Так как сущность e_1 добавлена в очередь ранее сущности e_2, то сущность e_1 также добавлена во время выполнения процедуры инициализации. Следовательно, $\mathrm{size}(e_1) = 0$, что противоречит условию $\mathrm{size}(e_1) > \mathrm{size}(e_2)$. Таким образом, e_2 добавляется в очередь не во время выполнения процедуры инициализации, а во время обработки некой сущности e. В момент завершения обработки сущности e сущность e_2 присутствует в очереди, и, следовательно, в данный момент сущность e_1 в очереди уже не присутствует.

Так как сущность e_2 была добавлена в очередь сразу вслед за сущностью e_1, то в данный момент сущность e_2 находится в голове очереди и в последовательности сущностей, которые попадают в очередь, сущность e непосредственно предшествует сущности e_2. Таким образом, $e = e_1$.

Так как e_2 добавляется в очередь во время обработки сущности e_1, то $\mathrm{size}(e_2) = \mathrm{size}(e_1) + 1$, что противоречит условию $\mathrm{size}(e_1) > \mathrm{size}(e_2)$. ■

Утверждение 13. Пусть сущность e_1 размера k_1 и сущность e_2 размера k_2 подаются как аргумент в процедуру размещения в очереди. Тогда если $k_1 < k_2$, то вызов процедуры размещения в очереди с аргументом e_1 предшествует вызову данной процедуры с аргументом e_2.

Доказательство. Поскольку $k_1 < k_2$, то $e_1 \neq e_2$. Следовательно, вызовы процедуры размещения в очереди с аргументами e_1 и e_2 не могут происходить одновременно.

Предположим обратное. Пусть вызов процедуры размещения в очереди с аргументом e_1 последовал за вызовом с аргументом e_2.

Предположим, что процедура размещения в очереди с аргументом e_2 была вызвана процедурой инициализации. Тогда $k_1 < 0 = \mathrm{size}(e_2) = k_2$, что невозможно, поскольку размер k_1 сущности e_1 не может быть отрицательным. Таким образом, процедура размещения в очереди с аргументом e_2 не была вызвана процедурой инициализации. В соответствии с предположением процедура инициализации не вызывала процедуру размещения в очереди также и с аргументом e_1.

Следовательно, найдутся сущности e'_1 и e'_2, такие, что вызовы процедуры размещения в очереди с аргументами e_1 и e_2 произошли во время выполнения процедуры обработки с аргументами e'_1 и e'_2 соответственно.

Таким образом, $\text{size}(e'_1) = \text{size}(e_1) - 1 = k_1 - 1$, $\text{size}(e'_2) = \text{size}(e_2) - 1 = k_2 - 1$. Отсюда следует $\text{size}(e'_1) < \text{size}(e'_2)$. В соответствии с утверждением 12 заключаем, что сущность e'_1 подается как аргумент в процедуру обработки ранее сущности e'_2.

Следовательно, вызов процедуры размещения в очереди с аргументом e_1, происходящий во время обработки e'_1, произошёл ранее вызова процедуры размещения в очереди с аргументом e_2, происходящего во время обработки e'_2, что противоречит предположению. ■

Утверждение 14. Пусть перебираемая сущность e размера k подаётся как аргумент в процедуру размещения в очереди. Тогда после выполнения данной процедуры в множестве принятых к обработке находится сущность e_1 размера не более k, подобная сущности e.

Доказательство. Предположим обратное: пусть после выполнения данной процедуры в множестве принятых к обработке нет сущности, подобной сущности e и имеющий размер не более k.

Предположим, что до выполнения процедуры в множестве принятых к обработке нет сущности, подобной сущности e. Тогда данная процедура добавляет в очередь и в множество принятых к обработке саму сущность e, и после её выполнения e находится в множестве принятых к обработке. Сущность e подобна самой себе и имеет размер не более k, что противоречит предположению. Таким образом, до выполнения процедуры размещения в очереди в множестве принятых к обработке лежит некая сущность e_1, подобная сущности e.

Сущность e_1 лежит в множестве принятых к обработке и после выполнения данной процедуры. Следовательно, $\text{size}(e_1) > \text{size}(e)$.

Так как в множество принятых к обработке попадают только сущности, добавляемые в очередь, а сущности, добавляемые в очередь, подавались как аргумент в процедуру размещения в очереди, то сущность e_1 подавалась как аргумент в процедуру размещения в очереди. Таким образом, вызов процедуры размещения в очереди с аргументом e_1 предшествует вызову с аргументом e. Применяя утверждение 13, получаем $\text{size}(e_1) \leqslant \text{size}(e)$ — противоречие. ■

3.3. Исключения

Установим свойства алгоритма, связанные с теми случаями, когда его работа оканчивается выдачей исключения.

Определение 19. Будем говорить, что алгоритм k-прерывен, если он выдаёт исключение при выполнении обработки некой перебираемой сущности размера k', где $k' \leqslant k$.

Замечание 1. Выражения «сущность подаётся как аргумент в процедуру обработки» и «сущность обрабатывается алгоритмом» различаются по смыслу. Если сущность обрабатывается, то она перед этим подаётся как аргумент. Однако обратное неверно, так как во время обработки поданной как аргумент сущности алгоритм может выдать исключение. В данном случае сущность не считается обработанной.

Утверждение 15. Пусть алгоритм k-прерывен. Тогда в процедуру обработки в качестве аргумента не может подаваться никакая сущность размера больше k.

Доказательство. Предположим обратное. Пусть e_1 — сущность размера не больше k, при обработке которой выдаётся исключение, а e_2 — сущность размера больше k, которая поступает как аргумент в процедуру обработки. Поскольку $\mathrm{size}(e_1) < \mathrm{size}(e_2)$, по утверждению 12 получаем, что сущность e_2 подавалась как аргумент в процедуру переработки позже, чем сущность e_1. Однако при обработке сущности e_1 алгоритм выдаёт исключение и завершает работу. Противоречие. ■

Утверждение 16. Пусть алгоритм не является k-прерывным и перебираемая сущность e размера не больше k попадает в очередь. Тогда сущность e обрабатывается алгоритмом.

Доказательство. Предположим обратное: сущность e не обрабатывается алгоритмом. Поскольку она попадает в очередь, то алгоритм выдаёт исключение.

Пусть алгоритм выдаёт исключение при обработке перебираемой сущности e_1. Поскольку сущность e не обрабатывается, то она остаётся не обработанной и в момент, непосредственно предшествующий подаче сущности e_1 как аргумента в процедуру обработки. Возможен в том числе и случай $e = e_1$.

В описанный момент сущность e_1 находится в голове очереди. Поскольку сущность e попадает в очередь, но не обрабатывается, то в данный момент она находится в очереди. Следовательно, $\mathrm{size}(e_1) \leqslant \mathrm{size}(e)$, поскольку очередь в каждый момент упорядочена по размеру сущностей (см. утверждение 11).

Таким образом, $\mathrm{size}(e_1) \leqslant \mathrm{size}(e) \leqslant k$ и алгоритм выдаёт исключение при обработке сущности e_1 размера не больше k. То есть алгоритм k-прерывен, что противоречит условию. ■

Утверждение 17. Пусть алгоритм не k-прерывен, e_1 — регулярная сущность размера k. Тогда найдётся регулярная сущность e_2 размера не больше k, подобная сущности e_1 и обрабатывающаяся алгоритмом.

Доказательство. Докажем по индукции.

Б а з а и н д у к ц и и : $k = 0$.

Пусть алгоритм не 0-прерывен, e_1 — регулярная сущность размера 0, т. е. инициальная.

Процедура инициализации подаёт на размещение в очереди все инициальные сущности, в том числе e_1. В соответствии с утверждением 14 после вызова процедуры размещения в очереди в множестве принятых к обработке лежит некая сущность e_2, подобная сущности e_1 и имеющая размер не более 0. Сущность не может иметь отрицательный размер, т. е. $\mathrm{size}(e_2) = 0$.

Так как e_2 попадает в множество принятых к обработке, то e_2 попадает и в очередь. В соответствии с утверждением 16 сущность e_2 обрабатывается алгоритмом, так как алгоритм не 0-прерывен и $\mathrm{size}(e_2) = 0$. Сущность e_2 является регулярной (см. утверждение 10).

Ш а г и н д у к ц и и. Пусть утверждение верно для некоторого k. Тогда оно верно и для $k + 1$.

Пусть алгоритм не $(k + 1)$-прерывен, e_1 — регулярная сущность размера $k + 1$.

По утверждению 8 найдутся регулярная сущность e_1' размера k и пара символов (s_1, s_2) алфавита S, такие, что e_1' под действием (s_1, s_2) переходит в e_1.

Так как алгоритм не $(k + 1)$-прерывен, то он и не k-прерывен. Применяя предположение индукции к регулярной сущности e_1' размера k, получаем, что найдётся регулярная сущность e_2' размера не больше k, подобная сущности e_1' и обрабатывающаяся алгоритмом.

В соответствии с утверждением 2, так как сущности e'_2 и e'_1 подобны и под действием (s_1, s_2) сущность e'_1 переходит в e_1, то найдётся сущность e_2, такая, что e'_2 под действием (s_1, s_2) переходит в e_2. Кроме того, $\mathrm{size}(e_2) \leqslant k+1$.

В соответствии с утверждением 1, так как сущность e'_1 подобна сущности e'_2 и под действием (s_1, s_2) сущности e'_1 и e'_2 переходят в e_1 и e_2 соответственно, то сущности e_1 и e_2 также являются подобными.

Так как e'_2 обрабатывается алгоритмом и под действием (s_1, s_2) сущность e'_2 переходит в e_2, то во время обработки e'_2 сущность e_2 подается как аргумент в процедуру размещения в очереди. В соответствии с утверждением 14 после вызова этой процедуры в множестве принятых к обработке лежит некая сущность e_3, подобная сущности e_2 и имеющая размер не больше размера сущности e_2. Сущность e_3 является регулярной (см. утверждение 10).

Сущность e_3 подобна e_2, сущность e_2 подобна e_1. Следовательно, сущность e_3 подобна сущности e_1.

Так как $\mathrm{size}(e_3) \leqslant \mathrm{size}(e_2) \leqslant k+1$ и алгоритм не $(k+1)$-прерывен, то в соответствии с утверждением 16 сущность e_3 обрабатывается алгоритмом. ■

3.4. Проверка правильности

Докажем, что алгоритм выдаёт исключение тогда и только тогда, когда F не является правильной функцией. В соответствии с теоремой 1 следует установить, что алгоритм выдаёт исключение тогда и только тогда, когда существует одноходовая аномалия 2-го рода для функции F, не являющаяся аномалией 1-го рода для функции F.

Утверждение 18. Пусть алгоритм выдаёт исключение. Тогда найдётся одноходовая пара слов алфавита S, являющаяся аномалией 2-го рода, но не являющаяся аномалией 1-го рода для функции F.

Доказательство. Пусть исключение выдаётся во время обработки регулярной (см. утверждение 10) сущности $e = (q_{\hat{s}_1}, q_{\hat{s}_2}, \alpha, \hat{s}_1, \hat{s}_2)$. Выпишем некоторые условия её регулярности: $(\hat{s}_1, \hat{s}_2)$ — одноходовая пара, не являющаяся аномалией 1-го или 2-го рода для функции F, слово α — суффикс слова $\hat{F}(\hat{s}_1)$, и $|\alpha| = |\hat{F}(\hat{s}_1)| - |\hat{F}(\hat{s}_2)|$. Тогда найдётся слово α_1, такое, что $\hat{F}(\hat{s}_1) = \alpha_1\alpha$ и $|\alpha_1| = |\hat{F}(\hat{s}_2)|$.

Так как при обработке сущности e алгоритм выдаёт исключение, то существует пара символов (s_1, s_2), под действием которой сущность e переходит в аномалию 2-го рода, но не переходит в аномалию 1-го рода. Значит, выполнено неравенство $|\alpha F(\hat{s}_1 s_1)| = |\alpha\psi(q_{\hat{s}_1}, s_1)| \leqslant |\psi(q_{\hat{s}_2}, s_2)| = |F(\hat{s}_2 s_2)|$ и, следовательно, $|\hat{F}(\hat{s}_1 s_1)| = |\hat{F}(\hat{s}_1)| + |F(\hat{s}_1 s_1)| = |\alpha_1| + |\alpha| + |F(\hat{s}_1 s_1)| = |\alpha_1| + |\alpha F(\hat{s}_1 s_1)| \leqslant |\alpha_1| + |F(\hat{s}_2 s_2)| = |\hat{F}(\hat{s}_2)| + |F(\hat{s}_2 s_2)| = |\hat{F}(\hat{s}_2 s_2)|$. Таким образом, пара $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ является аномалией 2-го рода для функции F.

Так как под действием (s_1, s_2) сущность e не переходит в аномалию 1-го рода, слова $\alpha F(\hat{s}_1 s_1)$ и $F(\hat{s}_2 s_2)$ не имеют общей абсолютно различимой позиции; так как $(\hat{s}_1, \hat{s}_2)$ не является аномалией 1-го рода для функции F, то и слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$ не имеют общей абсолютно различимой позиции. Поскольку α_1 — префикс $\hat{F}(\hat{s}_1)$, то слова α_1 и $\hat{F}(\hat{s}_2)$ также не имеют общей абсолютно различимой позиции.

Так как $|\alpha_1| = |\hat{F}(\hat{s}_2)|$, а пары слов $(\alpha_1, \hat{F}(\hat{s}_2))$ и $(\alpha F(\hat{s}_1 s_1), F(\hat{s}_2 s_2))$ не имеют общей абсолютно различимой позиции, то абсолютно различимой позиции нет также у слов $\hat{F}(\hat{s}_1 s_1) = \hat{F}(\hat{s}_1)F(\hat{s}_1 s_1) = \alpha_1\alpha F(\hat{s}_1 s_1)$ и $\hat{F}(\hat{s}_2 s_2) = \hat{F}(\hat{s}_2)F(\hat{s}_2 s_2)$. Таким образом, пара слов $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ не является аномалией 1-го рода для функции F.

Так как $(\hat{s}_1, \hat{s}_2)$ — одноходовая пара слов, то пара слов $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ также является одноходовой. Таким образом, $(\hat{s}_1 s_1, \hat{s}_2 s_2)$ — одноходовая пара слов, являющаяся аномалией 2-го рода, но не являющаяся аномалией 1-го рода для функции F. ■

Утверждение 19. Пусть одноходовая пара слов $(\hat{s}_1, \hat{s}_2)$ является аномалией 2-го рода для функции F, но не является аномалией 1-го рода для функции F, и $|\hat{s}_2| = k$. Тогда алгоритм выдаёт исключение и является $(k-1)$-прерывным.

Доказательство. Предположим обратное, т. е. что алгоритм не является $(k-1)$-прерывным.

Пусть $(\hat{s}'_1, \hat{s}'_2)$ — минимальная по длине $|\hat{s}'_2|$ одноходовая пара слов, являющаяся аномалией 2-го рода, но не являющаяся аномалией 1-го рода для функции F. Полагаем далее $|\hat{s}_2| = k'$. В соответствии с минимальностью длины выполнено $k' \leqslant k$.

Поскольку для любого символа $s \in S$ верно $|\hat{F}(s)| = |F(\Lambda)F(s)| = |\Lambda F(s)| = |F(s)| > |F(\Lambda)| = 0$, то (s, Λ) не может быть аномалией 2-го рода для функции F. Следовательно, $|\hat{s}'_2| > 0$, $|\hat{s}'_1| = |\hat{s}'_2| + 1 > 0$. Найдутся слова $\hat{s}'_1$ и $\hat{s}'_2$ и символы s_1 и s_2 алфавита S, такие, что $\hat{s}'_1 = \hat{s}''_1 s_1$, $\hat{s}'_2 = \hat{s}''_2 s_2$. Также выполнено $|\hat{s}''_2| = |\hat{s}'_2| - 1 = k' - 1 \leqslant \leqslant k - 1$. Пара $(\hat{s}''_1, \hat{s}''_2)$, очевидно, является одноходовой.

Так как $\hat{s}''_1$ — префикс $\hat{s}'_1$, $\hat{s}''_2$ — префикс $\hat{s}'_2$ и пара слов $(\hat{s}'_1, \hat{s}'_2)$ не является аномалией 1-го рода для функции F, то в соответствии с утверждением 3 пара слов $(\hat{s}''_1, \hat{s}''_2)$ не является аномалией 1-го рода для функции F.

Предположим, что пара $(\hat{s}''_1, \hat{s}''_2)$ содержит префиксным образом аномалию 2-го рода для функции F. Тогда существует одноходовая пара $(\hat{s}'''_1, \hat{s}'''_2)$, являющаяся аномалией 2-го рода для функции F, такая, что $\hat{s}'''_1$ — префикс $\hat{s}''_1$, $\hat{s}'''_2$ — префикс $\hat{s}''_2$. Тогда по утверждению 3 пара $(\hat{s}'''_1, \hat{s}'''_2)$ не является аномалией 1-го рода для функции F.

Таким образом, одноходовая пара $(\hat{s}'''_1, \hat{s}'''_2)$ является аномалией 2-го рода, но не является аномалией 1-го рода для функции F, и $|\hat{s}'''_2| \leqslant |\hat{s}''_2| = |\hat{s}_1| - 1$, что противоречит минимальности выбора пары $(\hat{s}'_1, \hat{s}'_2)$. Это означает, что предположение ложно, т. е. пара $(\hat{s}''_1, \hat{s}''_2)$ не содержит префиксным образом аномалию 2-го рода для функции F и, следовательно, не является аномалией 2-го рода для функции F.

Так как одноходовая пара $(\hat{s}''_1, \hat{s}''_2)$ не является аномалией 2-го рода для функции F, то $|\hat{F}(\hat{s}''_1)| > |\hat{F}(\hat{s}''_2)|$. Следовательно, найдутся слова α, α_1, такие, что $\hat{F}(\hat{s}''_1) = \alpha_1 \alpha$, $|\alpha_1| = |\hat{F}(\hat{s}''_2)|$, $|\alpha| = |\hat{F}(\hat{s}''_1)| - |\hat{F}(\hat{s}''_2)|$.

Рассмотрим сущность $e = (q_{\hat{s}''_1}, q_{\hat{s}''_2}, \alpha, \hat{s}''_1, \hat{s}''_2)$. Она регулярна и имеет размер не больше $k - 1$. Так как алгоритм не $(k-1)$-прерывен, то в соответствии с утверждением 17 найдётся регулярная сущность e_1 размера не более $k - 1$, подобная сущности e и обрабатывающаяся алгоритмом.

Так как $(\hat{s}'_1, \hat{s}'_2)$ является аномалией 2-го рода для функции F, то выполнено неравенство $|\hat{F}(\hat{s}'_1)| - |\hat{F}(\hat{s}'_2)| \leqslant 0$. Произведём расчет длин: $|\hat{F}(\hat{s}'_1)| - |\hat{F}(\hat{s}'_2)| = (|\alpha_1| + \\ + |\alpha F(\hat{s}'_1)|) - (|\hat{F}(\hat{s}''_2)| + |F(\hat{s}'_2)|) = |\alpha F(\hat{s}'_1)| - |F(\hat{s}'_2)|$. Таким образом, $|\alpha\psi(q_{\hat{s}'_1}, s_1)| = \\ = |\alpha F(\hat{s}''_1 s_1)| = |\alpha F(\hat{s}'_1)| \leqslant |F(\hat{s}'_2)| = |F(\hat{s}''_2 s_2)| = |\psi(q_{\hat{s}''_2}, s_2)|$. Следовательно, под действием (s_1, s_2) сущность e переходит в аномалию 2-го рода.

Предположим, что под действием (s_1, s_2) сущность e переходит в аномалию 1-го рода. Тогда слова $\alpha\psi(q_{\hat{s}''_1}, s_1) = \alpha F(\hat{s}''_1 s_1) = \alpha F(\hat{s}'_1)$, $\psi(q_{\hat{s}''_2}, s_2) = F(\hat{s}''_2 s_2) = F(\hat{s}'_2)$ имеют общую абсолютно различимую позицию.

Запишем тождества: $\hat{F}(\hat{s}'_1) = \hat{F}(\hat{s}''_1 s_1) = \hat{F}(\hat{s}''_1) F(\hat{s}''_1 s_1) = \alpha_1 \alpha F(\hat{s}'_1)$, $\hat{F}(\hat{s}'_2) = \\ = \hat{F}(\hat{s}''_2 s_2) = \hat{F}(\hat{s}''_2) F(\hat{s}''_2 s_2) = \hat{F}(\hat{s}''_2) F(\hat{s}'_2)$.

Из равенства $|\alpha_1| = |\hat{F}(\hat{s}''_2)|$ следует, что абсолютно различимая позиция слов $\alpha F(\hat{s}'_1)$ и $F(\hat{s}'_2)$ является также абсолютно различимой позицией слов $\hat{F}(\hat{s}'_1)$ и $\hat{F}(\hat{s}'_2)$. Таким

образом, пара слов $(\hat{s}'_1, \hat{s}'_2)$ является аномалией 1-го рода для функции F, что противоречит её выбору. Это означает, что под действием (s_1, s_2) сущность e не переходит в аномалию 1-го рода.

Итак, под действием (s_1, s_2) сущность e переходит в аномалию 2-го рода, но не переходит в аномалию 1-го рода, и сущность e_1 подобна сущности e. В соответствии с утверждением 1 под действием (s_1, s_2) сущность e_1 также переходит в аномалию 2-го рода, но не переходит в аномалию 1-го рода. Значит, алгоритм выдаёт исключение во время обработки сущности e_1. Поскольку сущность e_1 имеет размер не более $k-1$, алгоритм $(k-1)$-прерывен — противоречие. ■

3.5. Недостижимые сущности

Осталось установить, что алгоритм не может работать бесконечно долго. Докажем, что множество обрабатываемых сущностей является конечным, т. е. процедура обработки может вызываться только конечное число раз.

Определение 20. Регулярная перебираемая сущность вида $(q_1, q_2, \alpha, \hat{s}_1 s'_1 s''_1, \hat{s}_2)$ называется недостижимой, если выполнено неравенство $|\hat{F}(\hat{s}_1)| \geqslant |\hat{F}(\hat{s}_2)|$.

Утверждение 20. Пусть $e = (q_1, q_2, \alpha, \hat{s}_1 s'_1 s''_1, \hat{s}_2)$ — регулярная недостижимая сущность. Тогда сущность e не добавляется в очередь.

Доказательство. Предположим обратное: сущность e добавляется в очередь. Обозначим: $k = \mathrm{size}(e) = |\hat{s}_2|$.

В силу регулярности пара $(\hat{s}_1 s'_1 s''_1, \hat{s}_2)$ является одноходовой, т. е. $|\hat{s}_1 s'_1 s''_1| = |\hat{s}_2| + 1$. Следовательно, $|\hat{s}_2| = |\hat{s}_1| + 1$. Таким образом, пара $(\hat{s}_2, \hat{s}_1)$ также является одноходовой.

Так как $k = |\hat{s}_2| = |\hat{s}_1| + 1$, то $k > 0$ и сущность e не является инициальной. Следовательно, она добавляется в очередь во время обработки некой регулярной сущности e_1, $\mathrm{size}(e_1) = \mathrm{size}(e) - 1 = k - 1$. Таким образом, сущность e_1 размера $k-1$ подаётся как аргумент в процедуру обработки.

В силу регулярности пара $(\hat{s}_1 s'_1 s''_1, \hat{s}_2)$ не является аномалией 1-го рода для функции F, т. е. слова $\hat{F}(\hat{s}_1 s'_1 s''_1), \hat{F}(\hat{s}_2)$ не имеют общей абсолютно различимой позиции. Так как $\hat{F}(\hat{s}_1)$ — префикс $\hat{F}(\hat{s}_1 s'_1 s''_1)$, то общей абсолютно различимой позиции не имеют также слова $\hat{F}(\hat{s}_1)$ и $\hat{F}(\hat{s}_2)$. Таким образом, пара $(\hat{s}_2, \hat{s}_1)$ также не является аномалией 1-го рода для функции F.

Так как $|\hat{F}(\hat{s}_1)| \geqslant |\hat{F}(\hat{s}_2)|$, пара $(\hat{s}_2, \hat{s}_1)$ является аномалией 2-го рода для функции F. Ввиду $|\hat{s}_1| = |\hat{s}_2| - 1 = k - 1$ и утверждения 19 это значит, что алгоритм является $(k-2)$-прерывным. Тогда в соответствии с утверждением 15 сущность e_1 не может подаваться как аргумент в процедуру обработки — противоречие. ■

Путём длины 2 автоматного графа называется последовательность вида $q_1 s_1 q_2 s_2 q_3$, состоящая из состояний $q_1, q_2, q_3 \in Q$ и символов $s_1, s_2 \in S$, таких, что $q_2 = \varphi(q_1, s_1)$, $q_3 = \varphi(q_2, s_2)$. Данному пути длины 2 ставится в соответствие слово $\psi(q_1, s_1)\psi(q_2, s_2)$. Определим множество P_2, состоящее из слов алфавита A, следующим образом: слово α лежит в множестве P_2 тогда и только тогда, когда существует путь длины 2, поставленный в соответствие данному слову α. Путь длины 2 однозначно задаётся первым состоянием q_1 и символами s_1, s_2. Следовательно, выполняется неравенство $|P_2| \leqslant |Q||S|^2$. Каждое слово из P_2 получено конкатенацией двух слов, являющихся значениями выходной функции автомата. Таким образом, длина слов из P_2 не превышает $2L$.

Будем говорить, что слово β_1 является строгим суффиксом слова β_2, если β_1 — суффикс β_2 и $\beta_1 \neq \beta_2$. Множество слов, являющихся непустыми строгими суффик-

сами слов из множества P_2, обозначим P_*. Количество непустых строгих суффиксов каждого слова из P_2 не превышает $2L-1$. Следовательно, справедлива оценка $|P_*| \leqslant (2L-1)|Q||S|^2$.

Утверждение 21. Пусть $e = (q_{\hat{s}_1 s'_1 s_1}, q_{\hat{s}_2 s_2}, \alpha, \hat{s}_1 s'_1 s_1, \hat{s}_2 s_2)$ — регулярная сущность, не являющаяся недостижимой и не являющаяся инициальной. Тогда $\alpha \in P_*$.

Доказательство. Выпишем некоторые условия регулярности: $(\hat{s}_1 s'_1 s_1, \hat{s}_2 s_2)$ не является аномалией 2-го рода для функции F, т. е. $|\hat{F}(\hat{s}_2 s_2)| < |\hat{F}(\hat{s}_1 s'_1 s_1)|$. Слово α — суффикс слова $\hat{F}(\hat{s}_1 s'_1 s_1)$ и $|\alpha| = |\hat{F}(\hat{s}_1 s'_1 s_1)| - |\hat{F}(\hat{s}_2 s_2)| > 0$. Следовательно, существует слово α_1, такое, что $\hat{F}(\hat{s}_1 s'_1 s_1) = \alpha_1 \alpha$, $|\alpha_1| = |\hat{F}(\hat{s}_2 s_2)|$.

Так как сущность $(q_{\hat{s}_1 s'_1 s_1}, q_{\hat{s}_2 s_2}, \alpha, \hat{s}_1 s'_1 s_1, \hat{s}_2 s_2)$ не является недостижимой, то $|\hat{F}(\hat{s}_1)| < |\hat{F}(\hat{s}_2 s_2)| = |\alpha_1|$. Положим $\alpha' = F(\hat{s}_1 s'_1) F(\hat{s}_1 s'_1 s_1) = \psi(q_{\hat{s}_1}, s'_1)\psi(q_{\hat{s}_1 s'_1}, s_1)$. Слово α' соответствует пути длины 2 $q_{\hat{s}_1} s'_1 q_{\hat{s}_1 s'_1} s_1 q_{\hat{s}_1 s'_1 s_1}$ автоматного графа. Таким образом, $\alpha' \in P^2$.

Выпишем тождество: $\alpha_1 \alpha = \hat{F}(\hat{s}_1 s'_1 s_1) = \hat{F}(\hat{s}_1) F(\hat{s}_1 s'_1) F(\hat{s}_1 s'_1 s_1) = \hat{F}(\hat{s}_1)\alpha'$. Так как $|\hat{F}(\hat{s}_1)| < |\alpha_1|$, слово α является строгим непустым суффиксом слова $\alpha' \in P_2$. Следовательно, $\alpha \in P_*$. ■

3.6. Оценка сложности

Используя полученные результаты, оценим сверху количество запусков процедуры обработки (и тем самым покажем, что оно конечно), а также получим оценку сложности алгоритма.

Утверждение 22. В очередь попадает не более $|S| + (2L-1)|Q|^3|S|^2$ сущностей. Следовательно, процедура обработки запускается не более $|S| + (2L-1)|Q|^3|S|^2$ раз.

Доказательство.

1. Процедура инициализации добавляет в очередь не более $|S|$ инициальных сущностей.

2. Рассмотрим множество регулярных сущностей, не являющихся недостижимыми и не являющихся инициальными. Каждая такая сущность имеет вид $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$, где $q_1, q_2 \in Q$; $\alpha \in A^*$; $\hat{s}_1, \hat{s}_2 \in S^*$. В соответствии с утверждением 21 выполнено $\alpha \in P_*$.

3. Разделим данное множество на классы следующим образом: две сущности принадлежат одному классу, если они подобны, т. е. совпадают их первые три компоненты. Первые две компоненты могут принимать не более $|Q|$ различных значений, а третья компонента — не более $|P_*|$ значений. Таким образом, число классов разбиения конечно и не превышает $|Q|^2|P_*| \leqslant |Q|^2(2L-1)|Q||S|^2 = (2L-1)|Q|^3|S|^2$.

4. В соответствии с утверждениями 10 и 20 в очередь могут добавляться только инициальные сущности и сущности из множества п. 2. Согласно утверждению 9, в очередь может быть добавлена не более чем одна сущность из каждого класса разбиения. Следовательно, всего в очередь попадает не более $|S| + (2L-1)|Q|^3|S|^2$ сущностей.

Утверждение 22 доказано. ■

Утверждение 23. Возможность бесконечно долгой работы алгоритма исключена. Если F — правильная функция, то алгоритм остановится в состоянии пустой очереди, иначе он выдаст исключение. Сложность алгоритма составляет $O(L|Q|^3|S|^4)$.

Доказательство. В соответствии с теоремой 1 правильность функции F эквивалентна отсутствию аномалии 2-го рода для функции F, не являющейся аномалией 1-го рода для функции F.

В соответствии с утверждениями 18 и 19 исключение выдаётся тогда и только тогда, когда таковая аномалия присутствует, т. е. F не является правильной функцией.

В соответствии с утверждением 22 процедура обработки может быть запущена не более $|S| + (2L-1)|Q|^3|S|^2 \leqslant |S| + 2L|Q|^3|S|^2$ раз. Таким образом, возможность бесконечно долгой работы алгоритма исключена и, если функция F правильна, то алгоритм завершает свою работу в состоянии пустой очереди.

Проверка перехода в аномалии 1-го и 2-го рода, расчёт перехода в другую сущность, а также проверка наличия подобной сущности в множестве принятых к обработке занимает константное время. Следовательно, сложность процедуры обработки составляет $O(|S|^2)$.

Процедура предынициализации имеет сложность $O(1) + O(|Q||S|)$, процедура инициализации — $O(|S|)$.

Таким образом, сложность алгоритма составляет $O(1) + O(|Q||S|) + O(|S|) + O(|S|^2)(|S| + 2L|Q|^3|S|^2) = O(|S|) + O(|S|^3) + O(L|Q|^3|S|^4) = O(L|Q|^3|S|^4)$. ■

Формулировка утверждения 23 тождественна формулировке теоремы 4. Теорема 4 доказана.

Заключение

Представлен алгоритм, позволяющий проверить принадлежность функции F классу правильных за полиномиальное время от числа состояний $|Q|$ автомата, представляющего соответствующую ограниченно-детерминированную функцию $\hat{F}$, числа символов $|S|$ входного алфавита и максимальной длины L выходного слова.

Алгоритм хранит в очереди и в множестве принятых к обработке перебираемые сущности, представляющие собой пятёрки $(q_1, q_2, \alpha, \hat{s}_1, \hat{s}_2)$. Однако две последние компоненты были введены в состав сущности только с целью провести доказательство корректности работы алгоритма, а на практике их хранение в памяти является излишним.

Утверждения 1 и 2 гарантируют, что хранить в очереди и в множество принятых к обработке, а также подавать как аргумент в процедуры алгоритма вместо перебираемых сущностей можно краткие перебираемые сущности, т. е. тройки (q_1, q_2, α), где $q_1, q_2 \in Q$; $\alpha \in A^*$. Модифицированный алгоритм не работает бесконечно долго и выдаёт исключение тогда и только тогда, когда его выдаёт исходный алгоритм. Более того, модифицированный алгоритм добавляет в очередь и в множество принятых к обработке столько же сущностей, сколько исходный алгоритм. Сложность модифицированного алгоритма по времени совпадает со сложностью исходного алгоритма, т. е. составляет $O(L|Q|^3|S|^4)$.

Поскольку как в очередь, так и в множество принятых к обработке добавляется не более чем $|S| + (2L-1)|Q|^3|S|^2$ сущностей, то в памяти хранится не более чем $2|S| + 2(2L-1)|Q|^3|S|^2 = O(L|Q|^3|S|^2)$ сущностей. Хранение одной краткой сущности требует $O(L \ln|Q| \ln|A|)$ бит памяти. Таким образом, сложность модифицированного алгоритма по памяти составляет $O(L^2|Q|^3|S|^2 \ln|Q| \ln|A|)$

Отметим, что особый интерес представляет частный случай, когда в структуре частичного стирания нет абсолютно различимых символов. В этом случае условие правильности упрощается: так как отсутствуют аномалии 1-го рода для функции F, то для принадлежности функции F классу правильных необходимо и достаточно отсутствия аномалий 2-го рода. В дальнейшем планируется публикация, посвящённая имеющему меньшую сложность алгоритму валидации, который приспособлен специально к указанному частному случаю.

ЛИТЕРАТУРА

1. *Казаков И. Б.* Передача информации в каналах, задаваемых структурами частичного стирания. Ч. 1 // Программная инженерия. 2020. Т. 11. № 5. С. 277–284.
2. *Казаков И. Б.* Передача информации в каналах, задаваемых структурами частичного стирания. Ч. 2 // Программная инженерия. 2020. Т. 11. № 6. С. 322–329.
3. *Казаков И. Б.* Критерий существования корректного протокола в канале частичного стирания // Чебышевский сборник. 2021. Т. 22. № 1. С. 133–151.
4. *Lampson B. W.* A note on the confinement problem // Comm. ACM. 1973. V. 16. No. 10. P. 613–615.
5. *McFarland J.* Covert Channels: An Overview. `https://www.researchgate.net/publication/330875417_Covert_Channels_An_Overview`. 2017.
6. *Wendzel S., Zander S., Fechner S., et al.* A pattern-based survey and categorization of network covert channel techniques // ACM Comput. Surveys. 2015. V. 47. No. 3. P. 1–26.
7. *Zander S., Armitage G., and Branch P.* Covert channels in multiplayer first person shooter online game // 33rd IEEE Conf. Local Computer Networks. Montreal, Quebec, Canada, 2008. P. 215–222.

REFERENCES

1. *Kazakov I. B.* Peredacha informatsii v kanalakh, zadavaemykh strukturami chastichnogo stiraniya. Ch. 1 [Transmission of information in channels specified by structures of partial erasure. P. 1]. Programmnaya Inzheneriya, 2020, vol. 11, no. 5, pp. 277–284. (in Russian)
2. *Kazakov I. B.* Peredacha informatsii v kanalakh, zadavayemykh strukturami chastichnogo stiraniya. Ch. 2 [Transmission of information in channels specified by structures of partial erasure. P. 2]. // Programmnaya Inzheneriya, 2020, vol. 11, no. 6, pp. 322–329. (in Russian)
3. *Kazakov I. B.* Kriteriy sushchestvovaniya korrektnogo protokola v kanale chastichnogo stiraniya [Criterion for the existence of a consistent protocol in a partial erasure channel]. Chebyshevckiy Sbornik, 2021, vol. 22, no. 1, pp. 133–151. (in Russian)
4. *Lampson B. W.* A note on the confinement problem. Comm. ACM, 1973, vol. 16, no. 10, pp. 613–615.
5. *McFarland J.* Covert Channels: An Overview. `https://www.researchgate.net/publication/330875417_Covert_Channels_An_Overview`. 2017.
6. *Wendzel S., Zander S., Fechner S., et al.* A pattern-based survey and categorization of network covert channel techniques. ACM Comput. Surveys, 2015, vol. 47, no. 3, pp. 1–26.
7. *Zander S., Armitage G., and Branch P.* Covert channels in multiplayer first person shooter online game. 33rd IEEE Conf. Local Computer Networks, Montreal, Quebec, Canada, 2008, pp. 215–222.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

УДК 519.7 DOI 10.17223/20710410/59/7

ПОСТРОЕНИЕ И ВИЗУАЛИЗАЦИЯ ОБОБЩЁННОГО ДИАЛОГОВОГО ГРАФА ПО КОРПУСУ ДИАЛОГОВ

П. Д. Штыков, А. Г. Дьяконов

Московский государственный университет имени М. В. Ломоносова, г. Москва, Россия

E-mail: shtykov.pa@gmail.com, djakonov@mail.ru

Предлагается определение обобщённого диалогового графа, с помощью которого описывается структура диалога по корпусу однородных диалогов. Задача построения такого графа является актуальной в современном разговорном искусственном интеллекте, однако работ с конкретными результатами мало, часто не даётся полного описания алгоритмов, не выкладывается код с их реализацией. В настоящей работе предложен метод построения обобщённого диалогового графа, который реализован на языке программирования Python и выложен в открытый доступ. Проведены эксперименты на открытых данных и описаны их результаты.

Ключевые слова: *диалоговая система, обработка естественного языка, граф, диалоговый граф, кластеризация, векторные представления.*

A GENERALIZED DIALOGUE GRAPH CONSTRUCTION AND VISUALIZATION BASED ON A CORPUS OF DIALOGUES

P. D. Shtykov, A. G. Dyakonov

Lomonosov Moscow State University, Moscow, Russia

A definition of a generalized dialogue graph is proposed to describe the structure of a dialogue in terms of a corpus of homogeneous dialogues. The task of constructing such a graph is relevant in modern conversational artificial intelligence, however, there are few works with meaningful results, often a complete description of the algorithms is not given and the code with the implementation is not published. In the paper, a method for constructing a generalized dialogue graph is proposed, which is implemented in the Python programming language and made publicly available. Experiments were carried out on open data and the results were described.

Keywords: *dialogue system, NLP, graph, dialogue graph, clustering, embeddings.*

Введение

Обработка естественного языка (Natural Language Processing, NLP) является одним из ключевых направлений в области искусственного интеллекта. При этом одной из важнейших задач в NLP является обработка и понимание диалогов. В данной работе

исследуется один из подходов к представлению и анализу диалогов, а также представлению общей структуры диалога в виде графа, идея которого не нова, но публикаций по его реализации крайне мало.

Предположим, что у нас есть достаточное число диалогов из некоторой узкой предметной области. Например, это диалоги работников колл-центра банка с клиентами (здесь диалоги ещё и проблемно-ориентированные — task-oriented dialogs). Естественно предположить, что их можно разбить на группы одноцелевых диалогов, например «предложение новой услуги», «перевыпуск банковской карты» и т. п. Каждой группе сопоставлен граф диалога; у каждого работника колл-центра есть чёткая инструкция по общению с клиентом и ей соответствует ориентированный граф (рис. 1).

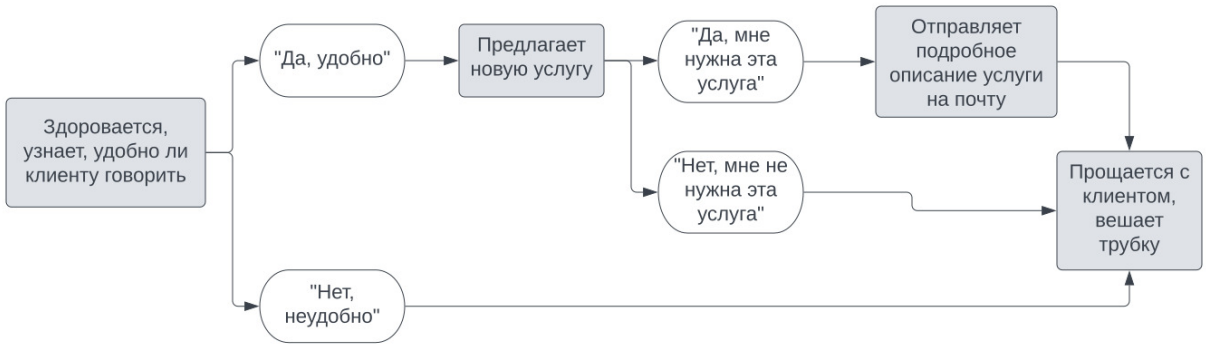


Рис. 1. Пример диалогового графа работника колл-центра с клиентом

По одному диалогу в общем случае невозможно восстановить диалоговый граф, поскольку не реализуются все варианты прохода по этому графу. По нескольким диалогам, которые соответствуют одному графу, это уже возможно. Теоретически (хороших практических реализаций этой идеи нет) можно восстановить набор графов по корпусу диалогов из некоторой узкой доменной области. Отметим специфику этой задачи:

— даже если графов несколько, у них есть общие вершины (например, вершина «поздороваться оператору»);
— не всегда диалог может идти по задуманному графу, возможны отклонения (например, пользователь просит повторить, отказывается от услуги и просит помощи в чём-то и т. п.);
— не всегда текущий ответ пользователя однозначно определяет переход на новую вершину графа (например, пользователь может попросить отключить услугу и закрыть счёт, оператор сам решает, с чего начать, и это определяет следующую вершину).

Отметим, что диалоговый граф, автоматически построенный по корпусу однородных диалогов, позволяет представить информацию в сжатой форме, подходящей как для визуализации, так и для встраивания в сложные диалоговые системы. В данной работе даётся понятие обобщённого диалогового графа и предлагаются способы его построения и визуализации (рассмотрим построение одного графа, а не набора).

1. Существующие подходы

Базовая идея построения диалогового графа состоит в поиске некоторой общей структуры диалогов в однородном корпусе. Такую структуру чаще всего представляют в виде ориентированного графа, вершины которого отражают темы реплик, а

дуги — переходы между ними (рис. 2). Приведём работы по изучению таких диалоговых графов.

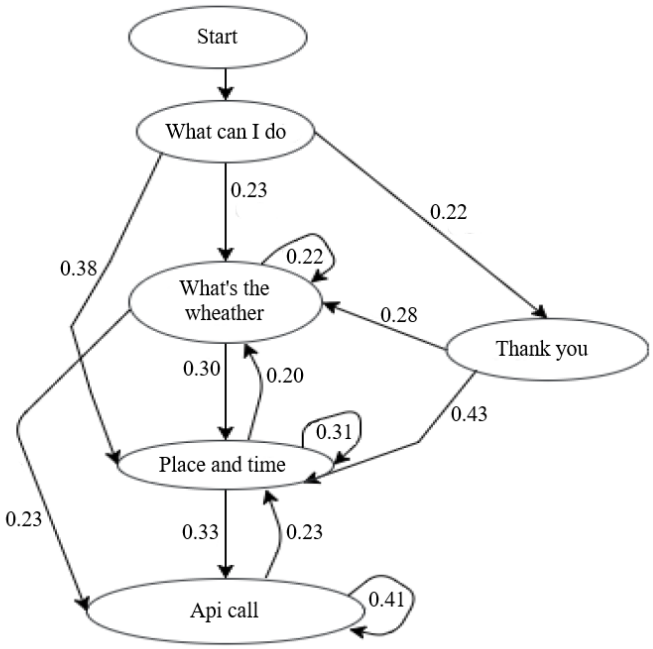


Рис. 2. Пример диалогового графа из [3]

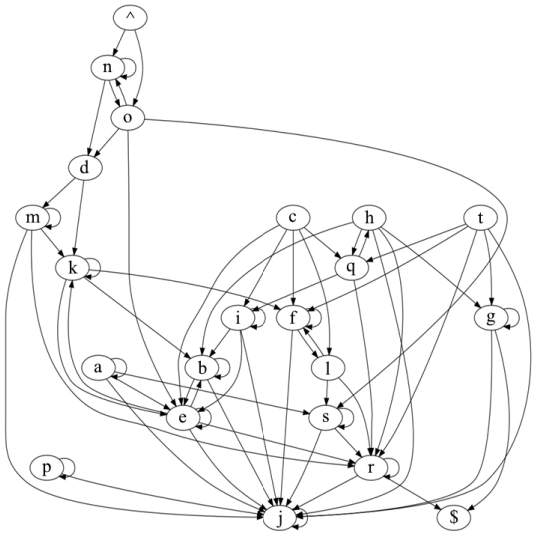
В [1, 2] предпринимались попытки обнаружения «структуры диалога», но без явного построения графа. Наиболее часто цитируемой является серия работ [3, 4], в которой предлагаются два способа построения такого графа. Первый основан на использовании рекуррентного вариационного автокодировщика [5] (Variational Recurrent Neural Network — VRNN), результат работы этого алгоритма представлен на рис. 2. Во второй авторы добавили в данную архитектуру механизм внимания [6], что позволило улучшить качество (Structured-Attention Variational Recurrent Neural Network — SVRNN).

В последние годы появляются работы на более амбициозные темы, например в [7] строится двухуровневый диалоговый граф для «открытого домена» (open domain dialogs). Обычно структура диалога выявляется для проблемно-ориентированных диалогов, здесь же рассматриваются более лексически разнообразные диалоги. В [7] используется довольно сложная техника: DVAE-GNN (Discrete Variational Auto-Encoder with Graph Neural Network).

На русском языке можно отметить работы [8, 9]. Первая выгодно отличается от многих наличием реализации в открытом доступе, в ней признаки, извлечённые из диалогового графа, используются при генерации реплик диалоговой системой.

В данной работе будем ориентироваться на метод TSCAN (Text SCAN) [10], в которой для построения диалогового графа применяется алгоритм классификации изображений с самообучением — SCAN (Semantic Clustering using Nearest Neighbors) [11]. Авторы адаптировали данный алгоритм для работы с текстами, использовав для получения векторных представлений (embeddings) текстов нейросеть BERT [12] архитектуры трансформер; один из примеров графа, полученного алгоритмом, приведён на рис. 3. В данной работе исследовано использование векторного представления, полученного с помощью модели SBERT (SentenceBERT, [13]), которое больше подходит

для семантической кластеризации. Однако в [10] авторы используют закрытый набор данных для сравнения алгоритма с более простыми методами, в частности с алгоритмом кластеризации k-средних (k-means) [14]. Кроме этого, авторы не предоставляют ни подробного алгоритма, ни его программного кода. В данной работе алгоритм реализован, выложен в открытый доступ, проведены эксперименты на открытых данных с разными методами кластеризации. Дополнено определение диалогового графа для более простого дальнейшего анализа определяемого объекта и визуализации.



Cluster-name	Self-label
^	Start Node
n	I am X
o	Am I talking to Y
d	I am calling from brand Z
m	This call is about product Z that you purchased last month
k	Have you made the payment for the last installment
f	The amount is five thousand three hundred dollars
l	Your due date is third of June
r	Your transaction id is five nine eight zero double two
g	When will the payment be done
$	End Node

Рис. 3. Пример диалогового графа из [10]

2. Постановка задачи

Пусть $D = \{d_1, d_2, \ldots, d_{|D|}\}$ — корпус диалогов, каждый диалог d_i является упорядоченным набором из нескольких высказываний: $d_i = \{d_i^1, d_i^2, \ldots, d_i^{n(i)}\}$. Каждое высказывание d_i^j — это текст, например «*К сожалению, такого товара нет в наличии. Можем ли мы Вам предложить что-то ещё*?». В дальнейшем мы будем работать с векторными представлениями текстов и высказывание d_i^j отождествлять с вещественным вектором. Множество всех высказываний в корпусе обозначим через

$$U = \bigcup_{i=1}^{|D|} \bigcup_{j=1}^{n(i)} d_i^j.$$

Мы будем работать с неразмеченными корпусами диалогов, в общем же случае нет ограничения на использование разметки (когда у каждого диалога или высказывания есть метка — некоторая дополнительная метаинформация). Для удобства считаем, что каждый диалог d_i начинается с «технического» высказывания «*начало диалога*»:

$$d_i^1 = \text{BEGIN},$$

а заканчивается «техническим» высказыванием «*конец диалога*»:

$$d_i^{n(i)} = \text{END},$$

аналогичные вершины будут и в диалоговом графе.

Определение 1. Назовём обобщённым диалоговым графом тройку

$$T = (G, p'(v'|v), p(u|v)),$$

где

— $G = (V, E)$ — ориентированный граф;

— $V \neq \varnothing$ — множество вершин графа G; $p(u|v)$ — функция вероятности отнесения высказывания $u \in U$ к вершине $v \in V$. Множество V интерпретируется как множество тем высказываний, поэтому $|V| < |U|$ (как правило, таких тем немного);

— E — множество дуг графа G, при этом каждой дуге $(v_j, v_i) \in E$ сопоставлена вероятность перехода по ней $p'(v_i|v_j)$, сумма вероятностей для дуг, выходящих из каждой вершины $v_j \in V$, равна 1: $\sum\limits_i p'(v_i|v_j) = 1$.

В определении есть произвол в выборе множеств V и E, формально в качестве G подойдёт любой ориентированный граф. Более того, если в качестве множества U взять не множество реплик диалогов корпуса, а множество всевозможных высказываний, то диалоговый граф не будет связан с конкретным корпусом. Однако понятна интерпретация указанных множеств: вершины $v \in V$ описывают темы реплик диалогов корпуса, а дуги $(v_j, v_i) \in E$ — чередование тем в диалогах. При этом функция p определяет связь между репликами и темами, а функция p' описывает смену тем. Основная задача при построении диалогового графа по корпусу диалога — чтобы описанные функции соответствовали чередованию реплик в диалогах конкретного корпуса. Дальше предлагается способ построения диалогового графа, в котором естественны указанные интерпретации.

Определение 1 не ограничивает нас в выборе модели для построения обобщённого диалогового графа. Обобщение (определений диалогового графа из предыдущих работ) связано с введением функций вероятности. Дополнительное требование наличия функции $p(u|v)$ позволяет вычислять статистики, полезные для визуализации и дальнейшего использования графа (например, самое вероятное предложение или самые частотные слова среди предложений, ассоциированных с текущей вершиной).

Такой граф достаточно просто обобщается на случай персонализированных диалогов (например, диалогов вида «пользователь» — «система») введением раскраски вершин, т. е. дополнительной функции $\phi(v)$, ставящей в соответствие каждой вершине персональный идентификатор пользователя (ID). Однако для корректности необходимо ввести дополнительные ограничения: смежные вершины не должны быть одинаково окрашены (так как высказывания пользователей чередуются), в графе нет петель (заметим, что основное определение их не запрещает). В данной работе мы будем строить диалоговый граф без дополнительной раскраски.

3. Предложенный метод

Чтобы не работать с текстами напрямую, в машинном обучении часто используют векторные представления: отображения вида

$$\mathrm{emb} : U \to \mathbb{R}^n, \quad n \in \mathbb{N},$$

которые реализуются с помощью нейронных сетей и позволяют работать не с текстом $u \in U$, а с вектором $\mathrm{emb}(u)$ фиксированной размерности. Для реализации представления мы использовали предобученную сиамскую нейронную сеть SBERT [13] с разными базовыми сетями (подробнее в п. 4). В дальнейшем, если не оговорено другого,

под высказыванием u будем подразумевать его представление $\mathrm{emb}(u)$ и отождествлять множество высказываний и множество их представлений, т. е.

$$U = \bigcup_{i=1}^{|D|} \bigcup_{j=1}^{n(i)} \mathrm{emb}(d_i^j) \subset \mathbb{R}^n.$$

При этом $n = 768$ или 384 в зависимости от использованной базовой сети в SBERT. В пространстве представлений введена косинусная мера сходства, отражающая семантическую близость высказываний, что позволяет использовать в этом пространстве простые методы кластеризации для объединения близких по смыслу высказываний.

Опишем алгоритм построения обобщённого диалогового графа $T{=}(G, p'(v'|v), p(u|v))$ для высказываний в пространстве представлений. Пусть выбран некоторый алгоритм кластеризации a, который разбивает множество U на кластеры — подмножества похожих высказываний. Множество полученных кластеров обозначим через V — оно будет множеством вершин графа. В результате работы алгоритма кластеризации определяется дискретная вероятность $p(v|u)$ принадлежности высказывания $u \in U$ к кластеру $v \in V$. При этом кластеризация может быть как жёсткой, например методом k-средних, так и мягкой, например смесью гауссиан (Gaussian mixture model, GMM) [14]. Зная $p(v|u)$, можно вычислить $p(u|v)$, используя теорему Байеса:

$$p(u|v) = \frac{p(v|u)p(u)}{\sum\limits_{i=1}^{|U|} p(v|u_i)p(u_i)},$$

где $p(u)$ — вероятность встречаемости высказывания u во всем корпусе диалогов. На практике будем пользоваться оценками вероятностей — частотами. Заметим, что вероятность $p(u)$ не одинакова для всех высказываний, так как в диалогах корпуса могут встречаться повторяющиеся высказывания.

Отдельно потребуем, чтобы технические высказывания BEGIN и END попадали в отдельные кластеры {BEGIN} и {END}, для этих высказываний может не быть представлений, для удобства можно считать, что

$$\mathrm{emb}(\mathrm{BEGIN}) = (+\infty, \ldots, +\infty),$$
$$\mathrm{emb}(\mathrm{END}) = (-\infty, \ldots, -\infty)$$

(и их представления достаточно далеки от представлений других высказываний диалогов).

Осталось определить в графе G дуги и найти вероятности, ассоциированные с ними. Введём вспомогательный ориентированный граф $\hat{G} = (\hat{V}, \hat{E})$, множество вершин $\hat{V} = U$ — множество высказываний из корпуса, множество рёбер

$$\hat{E} = \bigcup_{i=1}^{|D|} \bigcup_{j=1}^{n(i)-1} \{(d_i^j, d_i^{j+1})\} \subset \hat{V} \times \hat{V}$$

— множество пар последовательных высказываний в диалогах корпуса, каждое ребро (u_i, u_j) имеет вероятностную метку $p''(u_j|u_i)$, равную апостериорной вероятности встретить ответ u_j на высказывание u_i. Данный граф строится напрямую по корпусу диалогов.

На рис. 4 показана схема совместного размещения обоих графов G и $\hat{G}$. Соответственно матрица смежности $\hat{A}$ графа $\hat{G}$ определяется как

$$\hat{A} = (\hat{a}_{ij}), \quad \hat{a}_{ij} = p''(u_j|u_i)$$

для $1 \leqslant i, j \leqslant |U|$. Зная вероятности $p(u|v)$, $p''(u|u)$ и $p(v|u)$, можно вычислить вероятности дуг $p'(v|v)$ в графе G:

$$p'(v_j|v_i) = \sum_{\alpha,\beta} p(v_j|u_\beta) p''(u_\beta|u_\alpha) p(u_\alpha|v_i),$$

т. е. вероятность перехода из вершины v_i в вершину v_j графа G равна сумме вероятностей всех простых путей из v_i в v_j, проходящих через пары высказываний вида (u_α, u_β) в графе $\hat{G}$. Пример такого пути выделен на рис. 4 штрихпунктирной линией.

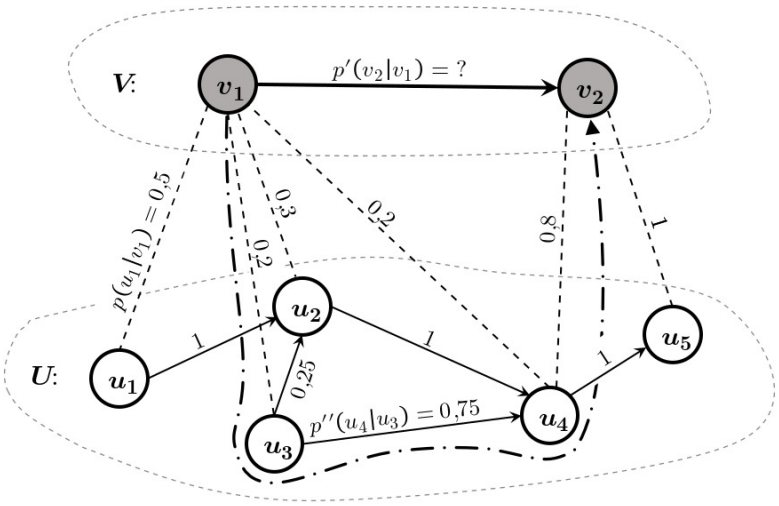


Рис. 4. Пример двух графов: G (сверху) в пространстве вершин V и $\hat{G}$ (снизу) в пространстве высказываний U

Вероятностная матрица смежности взвешенного графа G имеет вид

$$A = (a_{ij}), \quad a_{ij} = p'(v_j|v_i)$$

для $1 \leqslant i, j \leqslant |V|$. Так как в нашем случае совместные распределения $p(u|v)$ и $p(v|u)$ дискретны, то они могут быть представлены в виде матриц, поэтому способ вычисления матрицы смежности A может быть задан в явной матричной форме

$$A = P^{uv} \cdot \hat{A} \cdot P^{vu},$$

где матрицы в произведении определяются следующим образом:

$$P^{uv} = (p_{ij}) : \; p_{ij} = p(u_j|v_i), \quad P^{vu} = (p_{ij}) : \; p_{ij} = p(v_j|u_i).$$

Описание построения обобщённого диалогового графа T закончено. Заметим, что этод метод применим не только в случае кластеризации в пространстве представлений, но и в случае использования любого другого алгоритма, способного оценить апостериорные вероятности $p(v|u)$ (например, с помощью латентного размещения Дирихле (Latent Dirichlet allocation, LDA [15]) или нейронной сети, решающей задачу «от начала до конца» без промежуточного использования представлений). Кластеризация была выбрана как наиболее простой метод. Исходный код алгоритма и экспериментов доступен по ссылке [16].

4. Эксперименты

4.1. Данные для экспериментов

Для многих задач в анализе данных и машинном обучении есть стандартные наборы данных (так называемые «датасеты» — datasets), на которых отслеживается качество предложенных решений и определяется лучшее текущее решение — SotA (state of the Art), например на ресурсе `paperswithcode.com`. Постановка задачи, рассматриваемая в данной работе, достаточно нова: стандартных наборов данных, т. е. корпусов диалогов, имеющих некоторую общую *известную* структуру, почти нет. Этому критерию удовлетворяет лишь корпус STAR (Schema-Guided Dialog Dataset for Transfer Learning) [17], на котором не тестировались упомянутые выше методы. Поэтому для проведения экспериментов выбраны два известных корпуса, для которых можно предположить наличие общей структуры. Первый корпус — Customer Support on Twitter [18], в котором собраны ответы официальных аккаунтов технической поддержки крупных компаний. Для обеспечения однородности из него выбрано подмножество сообщений аккаунтов шести разных авиакомпаний США. Второй корпус — DailyDialog [19], в котором собраны обычные диалоги из повседневной жизни на разные темы. Для экспериментов были взяты диалоги на тему работы, как наиболее однородные. В результате подготовлены два набора данных: 8081 диалог в среднем по 3,6 высказываний в диалоге и 1924 диалога в среднем по 7,5 высказываний. Примеры диалогов из двух наборов приведены на рис. 5.

Twitter Customer Support

— @AlaskaAir it says you open at 5:15 @317258 where is everyone? #hellooooooo https://t.co/WePfUANLsZ

— @429415 @317258 Ticket counter opens at 615 is what I see on our website.

— @AlaskaAir @317258 all good! They just showed up thanks Andre

— @429415 That is good news

DailyDialog

— Everything's gone wrong.

— I know, it's not as I had planned.

— What are we going to do now?

— I'll speak to Bob, he'll be able to help us.

Рис. 5. Примеры диалогов из корпусов Twitter Customer Support и DailyDialog

Видно, что корпус Twitter Customer Support очень зашумлён, так как переписка в соцсети Twitter публична и в неё часто вмешиваются третьи лица. Поэтому в корпусе были оставлены только те диалоги между компаниями и пользователями, которые соответствуют следующей схеме:

«система» — «пользователь N» — «система» — «пользователь N» и т. д.

Из высказываний были убраны все идентификаторы пользователей, а идентификаторы компаний были заменены на единый токен «companyname». После этого была применена следующая предобработка текста:

— приведение всего текста к нижнему регистру;
— удаление слов с цифрами;

— удаление ссылок;
— лемматизация с помощью пакета NLTK [20];
— удаление стоп-слов (использовался набор стоп-слов из пакета NLTK).

Корпус DailyDialog зашумлён существенно меньше, поэтому к нему была применена только описанная предобработка.

4.2. Оценка качества кластеризации

Сначала рассмотрим визуализацию пространства представлений, полученных с помощью SBERT с базовой нейронной сетью Distill-RoBERTa [21]. Пространство отображено на плоскость с помощью t-SNE (t-distributed Stochastic Neighbor Embedding) [22] с перплексией равной 50 (рис. 6 и 7). Для диалогов из обоих корпусов заметна кластерная структура, однако кластеры имеют небольшие размеры и между ними много шума. Это может привести к зашумлению и самого диалогового графа.

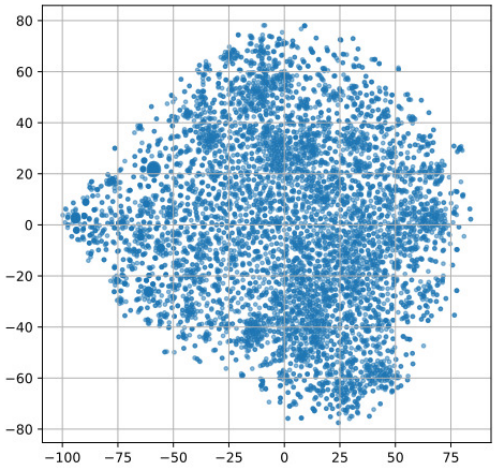


Рис. 6. Пространство представлений для корпуса DailyDialog

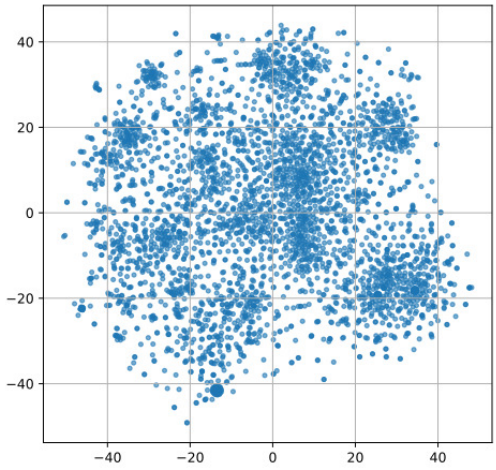


Рис. 7. Пространство представлений для корпуса Twitter Customer Support

Измерим качество кластеризации. В табл. 1 представлены результаты работы алгоритма с 5-ю вершинами в графе в зависимости от следующих параметров:

— базовая модель в SBERT: MPNet [23], DistillRoBERTa [21, 24] и MiniLM [25];
— кластеризатор: k-средних, смесь гауссиан (GMM) и SCAN [10].

Так как в [10] не указаны оптимальные гиперпараметры для SCAN, он обучался для каждой конфигурации с нуля со следующими стандартными гиперпараметрами:

— количество голов классификатора: 1;
— темп обучения: 10^{-5};
— количество эпох: 15.

Для графов с количеством вершин 10 и 15 аналогичная статистика приведена в табл. 2 и 3.

Таблица 1

Результаты сравнения качества кластеризации для графа с 5-ю вершинами для исследованных наборов данных

	Model	Twitter Customer Support				DailyDialog			
		Silh.	C.-H.	D.-B.	Entr.	Silh.	C.-H.	D.-B.	Entr.
$\|V\| = 5$	MPNet_KMeans	**0,052**	1043,9	3,912	0,535	0,011	**292,3**	5,166	0,712
	RoBERTa_KMeans	0,043	1003,3	4,43	0,606	**0,024**	281,8	5,176	0,719
	MiniLM_KMeans	0,045	1054,0	3,869	0,523	0,023	286,5	5,287	0,724
	MPNet_GMM	0,036	940,4	4,152	0,524	−0,001	253,3	5,517	0,694
	RoBERTa_GMM	0,034	988,7	4,544	0,617	0,022	278,1	5,156	0,71
	MiniLM_GMM	0,044	1046,8	**3,856**	**0,519**	0,01	273,4	**4,74**	**0,663**
	MPNet_SCAN	0,042	**1117,5**	4,169	0,576	0,017	230,8	5,992	0,717
	RoBERTa_SCAN	0,037	947,8	4,569	0,625	0,024	258,9	5,563	0,718
	MiniLM_SCAN	0,036	901,7	4,493	0,624	0,021	247,2	5,596	0,705

Таблица 2

Результаты сравнения качества кластеризации для графа с 10-ю вершинами для корпусов Twitter Customer Support и DailyDialog

	Model	Twitter Customer Support				DailyDialog			
		Silh.	C.-H.	D.-B.	Entr.	Silh.	C.-H.	D.-B.	Entr.
$\|V\| = 10$	MPNet_KMeans	0,04	672,1	4,018	0,659	0,016	**210,4**	4,422	0,778
	RoBERTa_KMeans	0,041	634,1	4,147	0,675	**0,021**	195,1	4,394	0,758
	MiniLM_KMeans	**0,054**	**693,4**	3,827	0,668	0,015	198,3	4,482	0,759
	MPNet_GMM	0,037	652,9	**3,669**	**0,597**	−0,002	195,5	4,899	0,776
	RoBERTa_GMM	0,036	617,6	3,992	0,611	0,018	185,1	4,724	0,767
	MiniLM_GMM	0,026	668,6	3,807	0,623	0,009	182,8	**4,373**	**0,73**
	MPNet_SCAN	0,032	626,7	4,569	0,663	0,014	176,7	5,117	0,809
	RoBERTa_SCAN	0,031	563,0	4,465	0,68	0,021	165,6	5,414	0,809
	MiniLM_SCAN	0,03	585,5	4,278	0,679	0,021	175,4	5,003	0,81

Таблица 3

Результаты сравнения качества кластеризации для графа с 15-ю вершинами для корпусов Twitter Customer Support и DailyDialog

	Model	Twitter Customer Support				DailyDialog			
		Silh.	C.-H.	D.-B.	Entr.	Silh.	C.-H.	D.-B.	Entr.
$\|V\| = 15$	MPNet_KMeans	0,038	530,2	3,704	0,659	0,018	**167,7**	4,329	0,796
	RoBERTa_KMeans	0,04	490,5	3,936	0,651	**0,023**	155,2	4,27	0,785
	MiniLM_KMeans	**0,049**	**533,7**	3,781	0,668	0,018	160,8	4,234	0,788
	MPNet_GMM	0,015	508,8	3,711	0,626	0,003	158,4	4,404	0,781
	RoBERTa_GMM	0,014	466,8	3,79	**0,623**	0,016	150,7	**4,14**	**0,772**
	MiniLM_GMM	0,03	509,1	**3,663**	0,637	0,014	155,6	4,341	0,775
	MPNet_SCAN	0,024	475,5	4,457	0,693	0,005	136,3	4,779	0,832
	RoBERTa_SCAN	0,024	447,4	4,418	0,694	0,023	135,2	4,937	0,829
	MiniLM_SCAN	0,024	439,9	4,43	0,696	0,022	140,1	5,132	0,826

В качестве показателей качества кластеризации использовались следующие базовые для неразмеченных данных: коэффициент силуэта (Silh.) [26], индекс Калински — Харабаса (C.-H.) [27] и индекс Дэвиса — Болдина (D.-B.) [28]. Введём дополнительный показатель для оценки структуры графа. Нам хотелось бы, чтобы граф был «более детерминированным»: если случайно блуждать по графу согласно приписанным дугам вероятностям, то в идеале переходы должны быть детерминированы (т. е. только

одна исходящая из вершины дуга имеет вероятность 1, а остальные — 0). Для этого будем измерять среднюю нормализованную энтропию (Entr.):

$$H(G) = \frac{1}{|V| \log |V|} \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} p'(v_j|v_i) \ln p'(v_j|v_i).$$

Чем меньше энтропия, тем более детерминирован граф.

Из табл. 1–3 видно, что мы не смогли в точности повторить результаты авторов TSCAN [10] их же методом: наша реализация SCAN-кластеризатора уступает стандартным алгоритмам k-средних и смеси гауссиан по всем показателям. Возможно, это связано с неправильно подобранными гиперпараметрами.

4.3. Визуализация графов

Построим и визуализируем графы, полученные предложенным методом. Для всех графов использовалась лучшая (по результатам из табл. 1–3) модель для данного количества вершин и набора данных. В качестве маркировки вершин будем использовать четыре слова из высказываний, которые соответствуют вершине, с самым большим значением Tf-Idf (TF — term frequency, IDF — inverse document frequency) [29]. Tf-Idf-представления строились для двухсот наиболее вероятных для данной вершины высказываний. Для удобства визуализации дуги не помечаются вероятностями, вероятность отображается толщиной дуги: чем толще дуга, тем больше вероятность перехода по ней. Также убраны дуги с вероятностями меньше 0,1. Визуализация графов производилась с помощью пакета GraphViz [30].

На рис. 8 и 9 представлены графы с 5-ю вершинами, составленные по обоим корпусам, на рис. 10 и 11 — с 10-ю вершинами, на рис. 12 и 13 — графы с 15-ю вершинами.

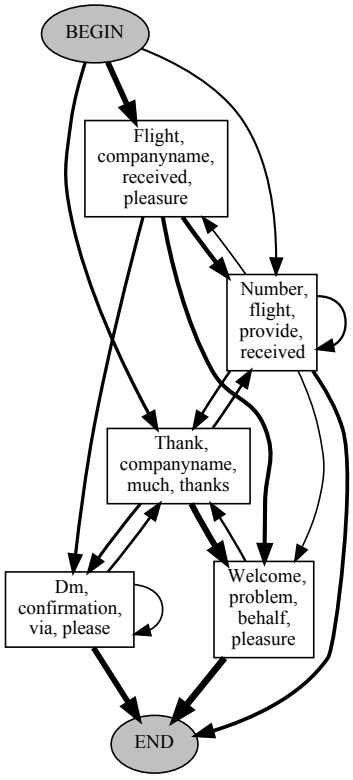


Рис. 8. Диалоговый граф с 5-ю вершинами для корпуса Twitter Customer Support

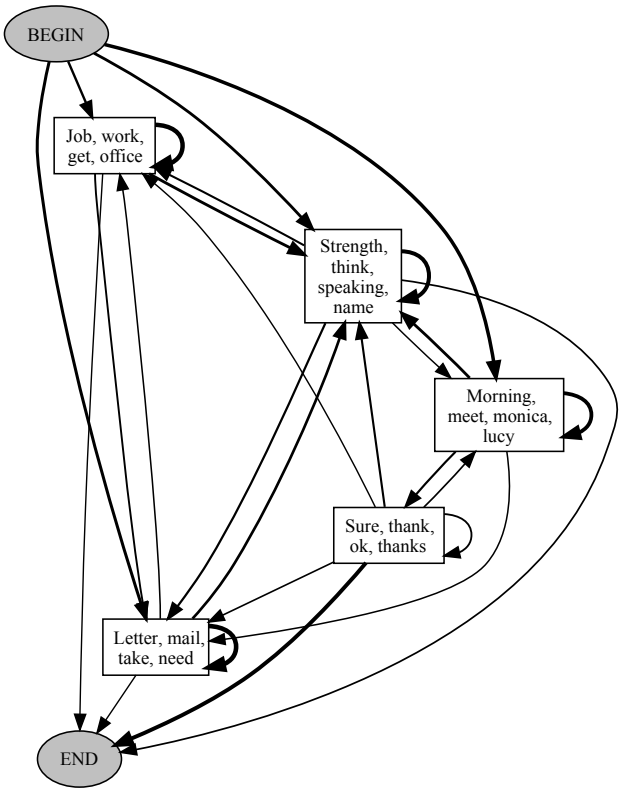


Рис. 9. Диалоговый граф с 5-ю вершинами для корпуса DailyDialog

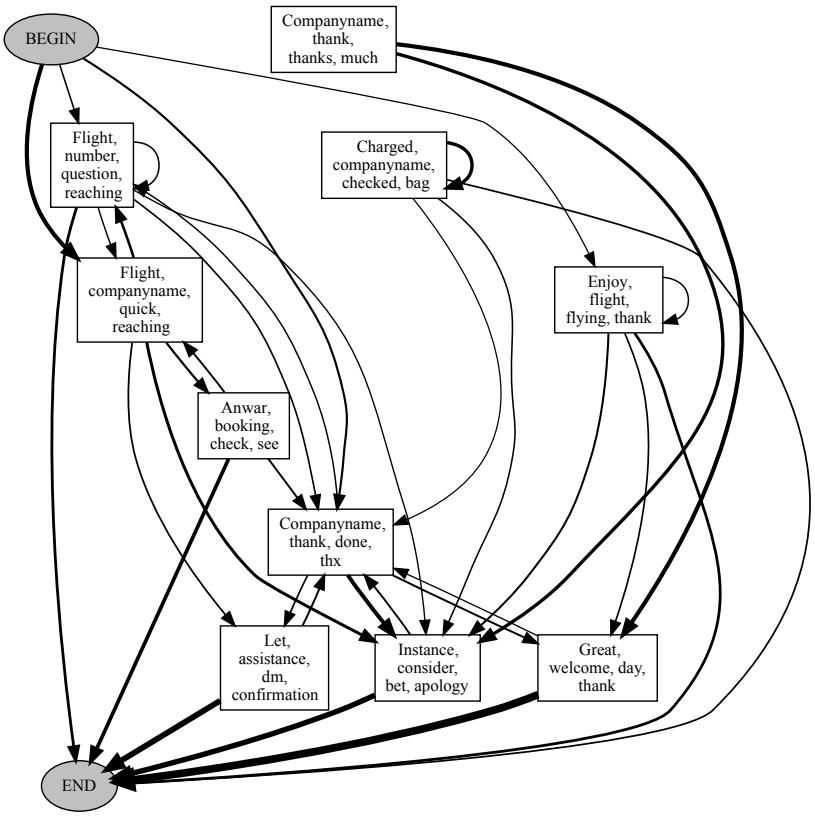


Рис. 10. Диалоговый граф с 10-ю вершинами для корпуса Twitter Customer Support

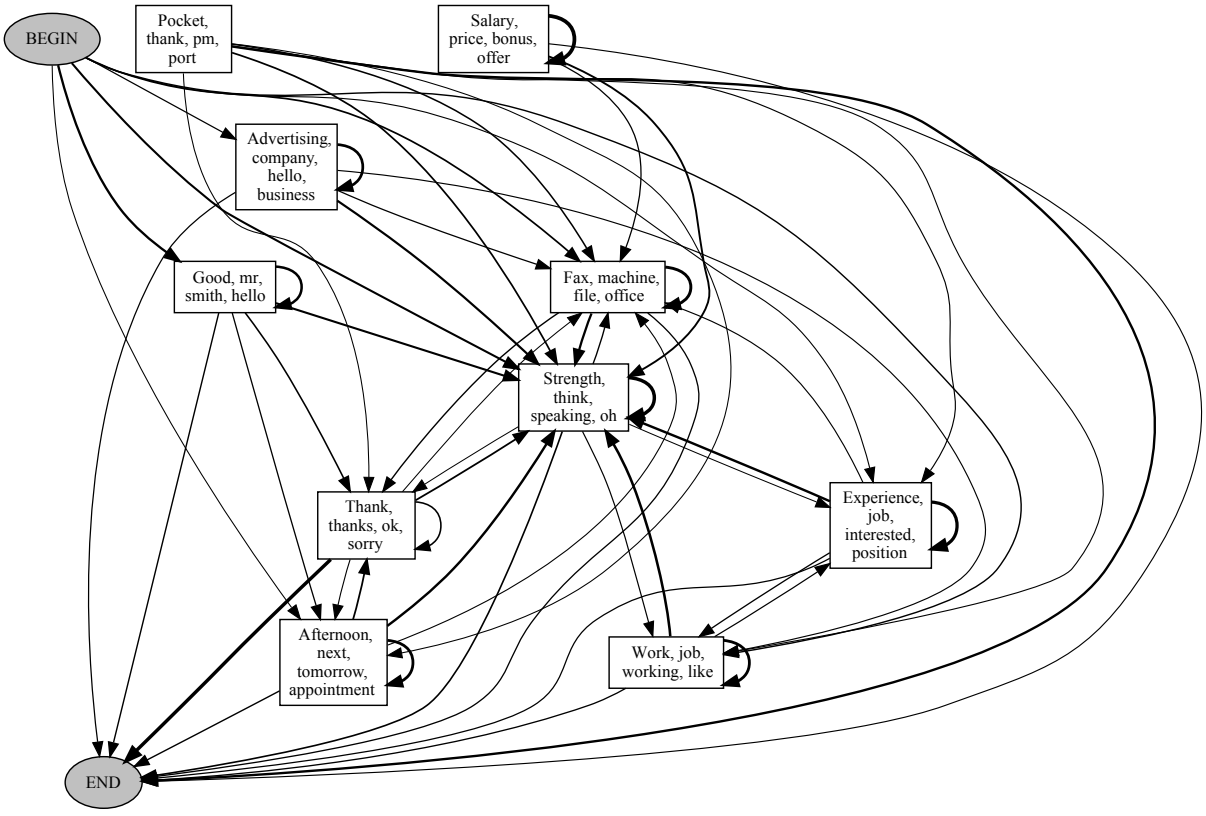


Рис. 11. Диалоговый граф с 10-ю вершинами для корпуса DailyDialog

Графы с 5-ю вершинами выглядят приемлемыми для анализа, дуги с разной толщиной позволяют понять, какие диалоги наиболее вероятны. Хотя Tf-Idf-представление является простым инструментом маркировки вершин, оно позволяет понять тему высказываний, которые соответствуют вершине. Графы с 10-ю и 15-ю вершинами становятся почти полносвязными, наиболее вероятные диалоги на них менее заметны, интерпретировать такие графы сложнее.

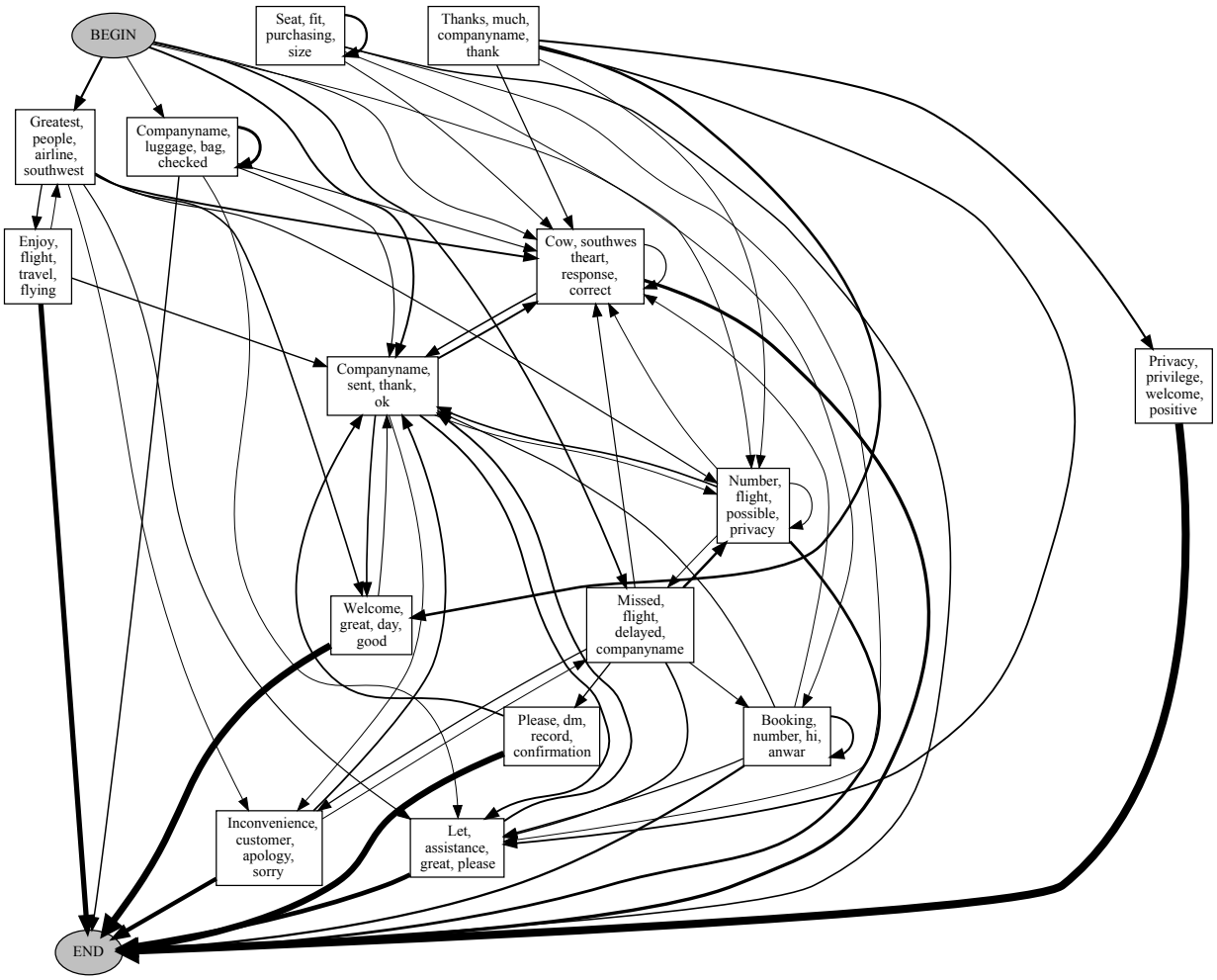


Рис. 12. Диалоговый граф с 15-ю вершинами для корпуса Twitter Customer Support

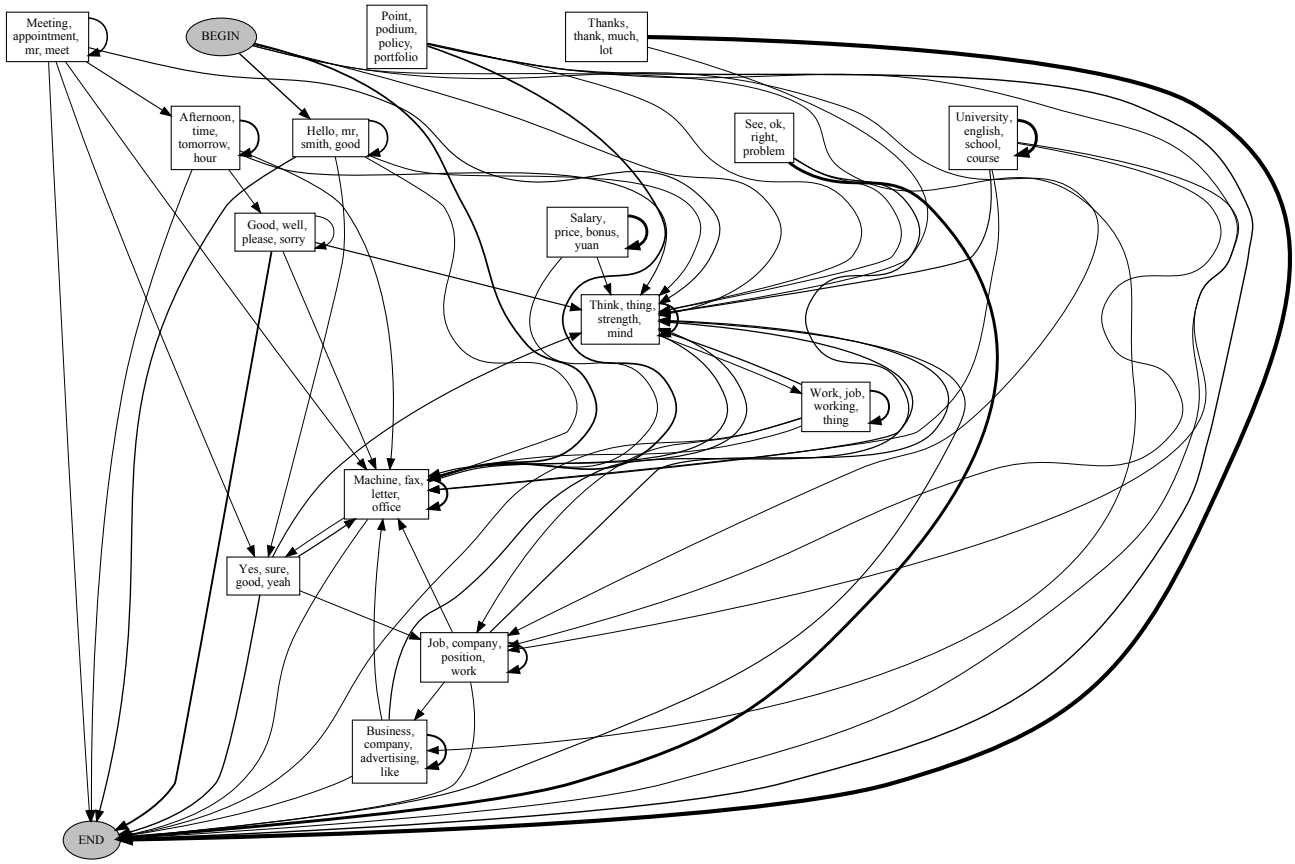


Рис. 13. Диалоговый граф с 15-ю вершинами для корпуса DailyDialog

Заключение

Предложен и исследован на двух наборах данных алгоритм построения обобщённого диалогового графа с помощью кластеризации в пространстве представлений SBERT. Проведены сравнения простых методов кластеризации со SCAN. В результате получены изображения диалоговых графов, пригодные для визуального анализа. Отметим, что работа оставляет довольно большой задел для будущих исследований:

— автоматическое определение числа вершин в графе (простейший вариант решения — использование алгоритма DBSCAN, см., например, [8, 9]);
— автоматическое определение высказываний, которые не соответствуют вершинам (например, отклонения от темы, здесь также можно использовать DBSCAN);
— автоматическая пометка вершин (хотелось бы, чтобы оно производилось полноценным предложением, здесь напрашивается применить технику реферирования, в [8] рассматривался вариант использования высказывания, чьё представление наиболее близко к центру соответствующего кластера);
— исследование оптимального представления высказываний и оптимальной кластеризации (это более эффективно решается с наборами данных, заточенных под решаемую задачу, например, если диалоговые графы построены экспертами или заданы изначально как в STAR [17]);
— проблема сравнения диалоговых графов (в идеале сравниваются ответы разных алгоритмов — графы, а не промежуточные результаты их работы — кластеризации);
— проблема построения нескольких графов (как описано во введении, по-видимому, такая постановка задачи ранее не рассматривалась).

Авторы выражают благодарность анонимному рецензенту за внимание к работе и полезные замечания, которые помогли существенно улучшить статью.

ЛИТЕРАТУРА

1. *Chotimongkol A.* Learning the structure of task-oriented conversations from the corpus of in-domain dialogs. PhD thesis. Carnegie Mellon University, 2008.
2. *Tang D., Li X., Gao J., et al.* Subgoal discovery for hierarchical dialogue policy learning // Proc. EMNLP. Brussels, Belgium, 2018. P. 2298–2309.
3. *Shi W., Zhao T., and Yu Z.* Unsupervised Dialog Structure Learning. ArXiv. 2019. arxiv.org/abs/1904.03736.
4. *Qiu L., Zhao Y., Shi W., et al.* Structured Attention for Unsupervised Dialogue Structure Induction. ArXiv. 2020. arxiv.org/abs/2009.08552.
5. *Chung J., Kastner K., Dinh L., et al.* A Recurrent Latent Variable Model for Sequential Data. ArXiv. 2015. arxiv.org/abs/1506.02216.
6. *Vaswani A., Shazeer N., Parmar N., et al.* Attention Is All You Need. ArXiv. 2017. arxiv.org/abs/1706.03762.
7. *Xu J., Lei Z., Wang H., et al.* Discovering Dialog Structure Graph for Open-Domain Dialog Generation. ArXiv. 2020. arxiv.org/abs/2012.15543.
8. *Юсупов И. Ф., Трофимова М. В., Бурцев М. С.* Построение и использование диалогового графа для улучшения оценки качества в целенаправленном диалоге // ТРУДЫ МФТИ. 2020. Т. 21. № 3. С. 75–86.
9. *Фельдина Е. А., Махныткина О. В.* Автоматическое построение дерева диалога по неразмеченным текстовым корпусам на русском языке // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21. № 5. С. 709–719.
10. *Nath A. and Kubba A.* TSCAN: Dialog Structure Discovery using SCAN. ArXiv. 2021. arxiv.org/abs/2107.06426.
11. *Van Gansbeke W., Vandenhende S., Georgoulis S., et al.* SCAN: Learning to Classify Images without Labels. ArXiv. 2020. arxiv.org/abs/2005.12320.
12. *Devlin J., Chang M.-W., Lee K., and Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv. 2018. arxiv.org/abs/1810.04805.
13. *Reimers N. and Gurevych I.* Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // ArXiv. 2019. arxiv.org/abs/1908.10084.
14. *Bishop C.* Pattern Recognition and Machine Learning. N.Y.: Springer, 2006. 738 p.
15. *Blei D., Ng A., and Jordan M.* Latent Dirichlet allocation // J. Machine Learning Res. 2003. V. 3. P. 993–1022.
16. https://github.com/PavelShtykov/generalized_dialogue_graph — Построение и визуализация обобщённого диалогового графа по корпусу диалогов. 2022.
17. *Mosig J., Mehri S., and Kober T.* STAR: A Schema-Guided Dialog Dataset for Transfer Learning. ArXiv. 2020. arxiv.org/abs/2010.11853.
18. www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter — Customer Support on Twitter. 2022.
19. *Li Y., Su H., Shen X., et al.* DailyDialog: A manually labelled multi-turn dialogue dataset // Proc. Eighth Int. Joint Conf. Natural Language Processing. Taipei, Taiwan, 2017. V. 1. P. 986–995.
20. https://www.nltk.org — Natural Language Toolkit. 2022.
21. *Liu Y., Ott M., Goyal N., et al.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv. 2019. arxiv.org/abs/1907.11692.

22. *Van der Maaten L. and Hinton G.* Viualizing data using t-SNE // J. Machine Learning Res. 2008. V. 9. P. 2279–2605.
23. *Song K., Tan X., Qin T., et al.* MPNet: Masked and Permuted Pre-training for Language Understanding. ArXiv. 2020. arxiv.org/abs/2004.09297.
24. *Sanh V., Debut L., Chaumond J., and Wolf T.* DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. ArXiv. 2019. arxiv.org/abs/1910.01108.
25. *Wang W., Wei F., Dong L., et al.* MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. ArXiv. 2020. arxiv.org/abs/2002.10957.
26. *Rousseeuw P. J.* Silhouettes: A graphical aid to the interpretation and validation of cluster analysis // J. Comput. Appl. Math. 1987. V. 20. P. 53–65.
27. *Caliński T. and Harabasz J.* A dendrite method for cluster analysis // Commun. in Statistics — Theory and Methods. 1974. V. 3. No. 1. P. 1–27.
28. *Davies D. L. and Bouldin D. W.* A cluster separation measure // IEEE Trans. Pattern Analysis and Machine Intelligence. 1979. V. 1. No. 2. P. 224–227.
29. *Spärck K. J.* A statistical interpretation of term specificity and its application in retrieval // J. Documentatio. 2004. V. 60. P. 493–502.
30. https://graphviz.org — Graphviz: open source graph visualization software. 2022.

REFERENCES

1. *Chotimongkol A.* Learning the structure of task-oriented conversations from the corpus of in-domain dialogs. PhD thesis, Carnegie Mellon University, 2008.
2. *Tang D., Li X., Gao J., et al.* Subgoal discovery for hierarchical dialogue policy learning. Proc. EMNLP, Brussels, Belgium, 2018, pp. 2298–2309.
3. *Shi W., Zhao T., and Yu Z.* Unsupervised Dialog Structure Learning. ArXiv, 2019, arxiv.org/abs/1904.03736.
4. *Qiu L., Zhao Y., Shi W., et al.* Structured Attention for Unsupervised Dialogue Structure Induction. ArXiv, 2020, arxiv.org/abs/2009.08552.
5. *Chung J., Kastner K., Dinh L., et al.* A Recurrent Latent Variable Model for Sequential Data. ArXiv, 2015, arxiv.org/abs/1506.02216.
6. *Vaswani A., Shazeer N., Parmar N., et al.* Attention Is All You Need. ArXiv, 2017, arxiv.org/abs/1706.03762.
7. *Xu J., Lei Z., Wang H., et al.* Discovering Dialog Structure Graph for Open-Domain Dialog Generation. ArXiv, 2020, arxiv.org/abs/2012.15543.
8. *Yusupov I. F., Trofimova M. V., and Burtsev M. S.* Postroenie i ispol'zovanie dialogovogo grafa dlya uluchsheniya otsenki kachestva v tselenapravlennom dialoge [Unsupervised graph extraction for improvement of multi-domain task-oriented dialogue modelling]. Proc. MIPT, 2020, vol. 21, no. 3, pp. 75–86. (in Russian)
9. *Fel'dina E. A. and Makhnytkina O. V.* Avtomaticheskoe postroenie dereva dialoga po nerazmechennym tekstovym korpusam na russkom yazyke [Automatic construction of the dialog tree based on unmarked text corpora in Russian]. Nauchno-Tekhnicheskiy Vestnik Informatsionnykh Tekhnologiy, Mekhaniki i Optiki, 2021, vol. 21, no 5, pp. 709–719. (in Russian)
10. *Nath A. and Kubba A.* TSCAN: Dialog Structure Discovery using SCAN. ArXiv, 2021, arxiv.org/abs/2107.06426.
11. *Van Gansbeke W., Vandenhende S., Georgoulis S., et al.* SCAN: Learning to Classify Images without Labels. ArXiv, 2020, arxiv.org/abs/2005.12320.
12. *Devlin J., Chang M.-W., Lee K., and Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv, 2018, arxiv.org/abs/1810.04805.

13. *Reimers N. and Gurevych I.* Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // ArXiv, 2019, arxiv.org/abs/1908.10084.
14. *Bishop C.* Pattern Recognition and Machine Learning. N.Y., Springer, 2006. 738 p.
15. *Blei D., Ng A., and Jordan M.* Latent Dirichlet allocation. J. Machine Learning Res., 2003, vol. 3, pp. 993–1022.
16. https://github.com/PavelShtykov/generalized_dialogue_graph. 2022.
17. *Mosig J., Mehri S., and Kober T.* STAR: A Schema-Guided Dialog Dataset for Transfer Learning. ArXiv, 2020, arxiv.org/abs/2010.11853.
18. www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter — Customer Support on Twitter, 2022.
19. *Li Y., Su H., Shen X., et al.* DailyDialog: A manually labelled multi-turn dialogue dataset. Proc. Eighth Int. Joint Conf. Natural Language Processing, Taipei, Taiwan, 2017. vol. 1, pp. 986–995.
20. https://www.nltk.org — Natural Language Toolkit, 2022.
21. *Liu Y., Ott M., Goyal N., et al.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv, 2019, arxiv.org/abs/1907.11692.
22. *Van der Maaten L. and Hinton G.* Viualizing data using t-SNE. J. Machine Learning Res., 2008, vol. 9, pp. 2279–2605.
23. *Song K., Tan X., Qin T., et al.* MPNet: Masked and Permuted Pre-training for Language Understanding. ArXiv, 2020, arxiv.org/abs/2004.09297.
24. *Sanh V., Debut L., Chaumond J., and Wolf T.* DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. ArXiv, 2019, arxiv.org/abs/1910.01108.
25. *Wang W., Wei F., Dong L., et al.* MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. ArXiv, 2020, arxiv.org/abs/2002.10957.
26. *Rousseeuw P. J.* Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math., 1987, vol. 20, pp. 53–65.
27. *Caliński T. and Harabasz J.* A dendrite method for cluster analysis. Commun. in Statistics — Theory and Methods, 1974, vol. 3, no. 1, pp. 1–27.
28. *Davies D. L. and Bouldin D. W.* A cluster separation measure. IEEE Trans. Pattern Analysis and Machine Intelligence, 1979, vol. 1, no. 2, pp. 224–227.
29. *Spärck K. J.* A statistical interpretation of term specificity and its application in retrieval. J. Documentatio, 2004, vol. 60, pp. 493–502.
30. https://graphviz.org — Graphviz: open source graph visualization software, 2022.

СВЕДЕНИЯ ОБ АВТОРАХ

ВОБЛЫЙ Виталий Антониевич — доктор физико-математических наук, ведущий научный сотрудник Всероссийского института научной и технической информации РАН, г. Москва. E-mail: **vitvobl@yandex.ru**

ДУРНЕВ Валерий Георгиевич — доктор физико-математических наук, профессор, профессор кафедры компьютерной безопасности и математических методов обработки информации математического факультета Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **durnev@uniyar.ac.ru**

ДЬЯКОНОВ Александр Геннадьевич — доктор физико-математических наук, профессор Московского государственного университета имени М. В. Ломоносова, г. Москва. E-mail: **djakonov@mail.ru**

ЖАРКОВА Анастасия Владимировна — кандидат физико-математических наук, доцент кафедры теоретических основ компьютерной безопасности и криптографии Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского, г. Саратов. E-mail: **ZharkovaAV3@gmail.com**

ЗЕТКИНА Алена Игоревна — аспирантка кафедры теоретической информатики факультета информатики и вычислительной техники Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **a.zetkina1@uniyar.ac.ru**

КАЗАКОВ Илья Борисович — кандидат физико-математических наук, ассистент кафедры высшей математики Московского физико-технического института, г. Долгопрудный. E-mail: **i_b_kazakov@mail.ru**

МАРТЫНЕНКОВ Игорь Владимирович — АО «КВАНТ-ТЕЛЕКОМ», г. Москва. E-mail: **mivpost@yandex.ru**

ЦАРЕГОРОДЦЕВ Кирилл Денисович — старший специалист-исследователь лаборатории криптографии, АО «НПК “Криптонит”», г. Москва.
E-mail: **k.tsaregorodtsev@kryptonite.ru**

ШТЫКОВ Павел Дмитриевич — бакалавр Московского государственного университета имени М. В. Ломоносова, г. Москва. E-mail: **shtykov.pa@gmail.com**