МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2013 №3(21)

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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Адрес редакции: 634050, г. Томск, пр. Ленина, 36
E-mail: vestnik_pdm@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 надежности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Подписано к печати 11.09.2013.
Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 13,4. Уч.-изд. л. 15. Тираж 300 экз.

Издательство ТГУ. 634029, Томск, ул. Никитина, 4

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 519.1

ПРОИЗВОДЯЩИЕ ФУНКЦИИ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ СВЯЗНЫХ ПОКРЫТИЙ

Р. М. Ганопольский

Тюменский государственный университет, г. Тюмень, Россия

E-mail: rodion@utmn.ru

Вводится понятие связных покрытий, рассматриваются производящие функции последовательности комбинаторных чисел, исчисляющих количество связных покрытий конечного множества подмножествами с заданными мощностями и свойствами. Проведён анализ производящих функций, приведены примеры преобразований, получен ряд рекуррентных соотношений.

Ключевые слова: *покрытие, связное покрытие, конечное множество, подмножества, комбинаторные числа, производящие функции, связные графы.*

Введение

В работе [1] введены комбинаторные числа неупорядоченных покрытий конечного множества мощности n подмножествами с фиксированными мощностями

$$ {}_nN(k_1, k_2, \ldots, k_n), \tag{1} $$

где k_i — количество подмножеств мощности i в покрытии. В случае, когда часть коэффициентов $k_i = 0$, используется альтернативное обозначение

$$ {}_nN^{k_1 k_2 \cdots k_m}_{l_1\, l_2 \cdots l_m}, $$

где k_i — количество подмножеств мощности l_i в покрытии. Для введённых комбинаторных чисел получена формула

$$ {}_nN^{k_1 k_2 \cdots k_m}_{l_1\, l_2 \cdots l_m} = \prod_{i=1}^{m} C^{k_i}_{C_n^{l_i}} + \sum_{i \geqslant 1} (-1)^i C_n^i \prod_{j=1}^{m} C^{k_j}_{C_{n-i}^{l_j}}, $$

где $C_j^i = \dfrac{j!}{i!(j-i)!}$ — биномиальный коэффициент, и соотношение

$$ \sum_{i \geqslant 0} C_n^i \, {}_{n-i}N^{k_1 k_2 \cdots k_m}_{l_1\, l_2 \cdots l_m} = \prod_{i=1}^{m} C^{k_i}_{C_n^{l_i}}. $$

В случае, когда в (1) $k_n = 1$, получим ${}_nN(k_1, k_2, \ldots, k_{n-1}, 1) = \prod\limits_{i=1}^{n-1} C^{k_i}_{C_n^i}$. Кроме того, принято, что ${}_0N_0^0 = 1$, то есть число покрытий пустого множества нулевым количеством пустых подмножеств равно 1.

В работе [2] рассматриваются производящие функции последовательности чисел (1): обычная —

$$F(x; A_1, A_2, A_3, \ldots) = \sum_{j\geqslant 0} \frac{x^j \prod\limits_{i=1}^{j} (1 + A_i)^{C_j^i}}{(1 + x)^{j+1}}$$

и экспоненциальная —

$$E(x; A_1, A_2, A_3, \ldots) = e^{-x} \sum_{j\geqslant 0} \frac{x^j}{j!} \prod_{i=1}^{j} (1 + A_i)^{C_j^i}. \tag{2}$$

Числа (1) в этих функциях являются коэффициентами перед мономами $x^n \prod\limits_i A_i^{k_i}$ в случае обычной производящей функции и перед выражением $\dfrac{x^n}{n!} \prod\limits_i A_i^{k_i}$ — в случае экспоненциальной.

В работе [2] получено соотношение между производными производящей функции по A_1 и x:

$$x \left(E + \frac{\partial E}{\partial x} \right) = \frac{\partial E}{\partial A_1} (1 + A_1). \tag{3}$$

Заменяя все переменные A_i одной переменной A, получаем экспоненциальную производящую функцию для последовательности чисел k-покрытий (покрытия, содержащие ровно k подмножеств [3]):

$$E(x; A) = e^{-x} \sum_{n\geqslant 0} \frac{x^n}{n!} (1 + A)^{2^n - 1} = \sum_{n\geqslant 0} \frac{x^n}{n!} \sum_{k=0}^{2^n-1} A^k \sum_{j=0}^{\infty} (-1)^j C_n^j C_{2^{n-j}-1}^k, \tag{4}$$

где коэффициент перед $x^n A^k / n!$ — это количество k-покрытий множества мощности n.

При $A = 1$ получаем экспоненциальную производящую функцию последовательности чисел покрытий множества мощности n [3, 4]:

$$E(x; 1) = \sum_{n\geqslant 0} \frac{x^n}{n!} \sum_{j=0}^{n} (-1)^j C_n^j 2^{2^{n-j}-1}. \tag{5}$$

1. Связные покрытия и производящие функции

Каждому покрытию можно поставить в соответствие двудольный граф, где одна часть вершин соответствует элементам множества, а другая — подмножествам, входящим в покрытие [1]. По аналогии со связными графами [5, 6] введём понятие связного покрытия: покрытие является связным, если соответствующий ему двудольный граф является связным, — а также понятие компоненты связности покрытия: из двудольного графа выделяем компоненту связности и соответствующее ей подмножество, и его покрытие будем называть компонентой связности покрытия. Таким образом, количества компонент связности покрытия и соответствующего ему двудольного графа равны. Покрытие, состоящее из k компонент связности, представляет собой объединение k непересекающихся подсемейств, являющихся связными покрытиями k непересекающихся подмножеств исходного множества, то есть совокупность пар (множество, покрытие) можно разбить на k независимых непересекающихся частей.

Рассмотрим классы покрытий со следующим свойством: если покрытие, состоящее из k компонент связности, входит в один из этих классов, то в этот же класс входит каждая компонента связности. Такими классами покрытий являются: все неповторяющиеся покрытия (покрытия, подмножества которых различны); минимальные

покрытия (покрытия, из которых нельзя изъять ни одного подмножества так, чтобы оно осталось покрытием исходного множества) [3]; антицепи (покрытия, никакое подмножество которых не является подмножеством другого) [3]; разбиения (покрытия, где каждый элемент множества принадлежит только одному подмножеству из покрытия); покрытия, где каждый элемент принадлежит как минимум двум подмножествам из покрытия, и т. п. Согласно экспоненциальной теореме, экспоненциальные производящие функции чисел связных покрытий $f(x)$ связаны с экспоненциальными производящими функциями чисел покрытий $E(x)$ следующими соотношениями [5]:

$$\begin{aligned} E(x) &= e^{f(x)}, \\ f(x) &= \ln\left(E(x)\right). \end{aligned} \tag{6}$$

Разложение экспоненциальных производящих функций по степеням связности имеет вид

$$E(x;y) = e^{f(x)y}, \tag{7}$$

где выражение $\dfrac{1}{k!}(f(x))^k$ перед y^k является производящей функцией последовательности чисел покрытий, состоящих из k компонент связности. Число всех m-покрытий множества мощности n (${}_nN_m$) является суммой всех чисел m-покрытий этого же множества, состоящих из k компонент связности (${}_n^k\tilde{N}_l$) [5]:

$${}_nN_m = \sum_{k=0}^{m} {}_n^k\tilde{N}_m,$$

а число всех покрытий множества мощности n равно сумме всех чисел покрытий этого множества, состоящих из k компонент связности:

$${}_nN = \sum_{k=0}^{n} {}_n^k\tilde{N}. \tag{8}$$

Коэффициент при нулевой степени y в (7) равен 1, то есть пустое покрытие пустого множества является покрытием, состоящим из 0 компонент связности. Если пустое покрытие пустого множества не входит в класс покрытий (примером может служить класс покрытий, в которых каждый элемент принадлежит как минимум двум подмножествам из покрытия), то соотношение (6) будет иметь вид $E(x) = e^{f(x)} - 1$, а выражение (7) — $E(x) = e^{f(x)y} - 1$.

2. Анализ производящих функций

В работе [2] показано, что если произвольная производящая функция чисел покрытий является сложной функцией от экспоненциальной производящей функции (5), то производящая функция чисел покрытий с фиксированными числом и мощностями подмножеств будет сложной функцией от функции (2):

$$f(x; A_1, A_2, A_3, \ldots) = \ln\left(E(x; A_1, A_2, A_3, \ldots)\right).$$

Эквивалентное равенство справедливо и для экспоненциальных производящих функций последовательности чисел k-покрытий:

$$f(x; A) = \ln\left(E(x; A)\right).$$

Используя явные выражения для экспоненциальных производящих функций (2), получим соответствующие выражения для производящих функций связных покрытий:

$$f(x; A_1, A_2, A_3, \ldots) = \ln\left(\sum_{j\geqslant 0} \frac{x^j}{j!} \prod_{i=1}^{j} (1 + A_i)^{C_j^i}\right) - x, \tag{9}$$

$$f(x; A) = \ln\left(\sum_{n\geqslant 0} \frac{x^n}{n!} \sum_{k=0}^{2^n-1} C_{2^n-1}^k A^k\right) - x.$$

Приравнивая A_i для всех $i > 1$ нулю, получим

$$f(x; A_1) = A_1 x. \tag{10}$$

Подставив в соотношение (3) выражение (6), получим соотношение между частными производными функции f:

$$x\left(1 + \frac{\partial f}{\partial x}\right) = \frac{\partial f}{\partial A_1}(1 + A_1). \tag{11}$$

Разлагая (2) по степеням x и A_i, получим

$$f(x; A_1, A_2, \ldots) = P_1(A_1)x + P_2(A_1, A_2)\frac{x^2}{2!} + P_3(A_1, A_2, A_3)\frac{x^3}{3!} + \cdots, \tag{12}$$

где P_i — некие полиномы от переменных A_i. Из (10) следует, что $P_1(A_1) = A_1 x$ и $P_i(A_1, 0, \ldots, 0) = 0$. Подставляя (12) в (11) и учитывая (10), получаем, что производящая функция чисел связных покрытий f имеет вид

$$f(x; A_1, A_2, \ldots) = A_1 x + p_2(A_2)\frac{(1 + A_1)^2 x^2}{2!} + p_3(A_2, A_3)\frac{(1 + A_1)^3 x^3}{3!} + \cdots,$$

где p_i — полиномы, свободные члены которых равны нулю.

Все неповторяющиеся покрытия можно разделить на два непересекающихся класса: антицепи и покрытия, в которых хотя бы одно множество включает в себя другое множество этого покрытия (назовём их не-антицепи). Таким образом, сумма производящих функций последовательности чисел не-антицепей G и антицепей H равна

$$E(x, A) = H(x, A) + G(x, A).$$

Покрытие, состоящее из k компонент связности, является не-антицепью, если хотя бы одна компонента связности покрытия является не-антицепью. Воспользовавшись для E и H соотношением (6), получим выражение для G:

$$G(x, A) = e^{h(x,A)}\left(e^{g(x,A)} - 1\right),$$

где g — производящая функция связных не-антицепей, а h — связных антицепей.

3. Рекуррентные соотношения

Получим систему рекуррентных соотношений для комбинаторных чисел неповторяющихся связных k-покрытий. Умножим левые части (4) на $(1 + A)$ и найдём производную по A:

$$\frac{\partial}{\partial A}\left((1 + A)E(x, A)\right) = e^{-x} \sum_{n\geqslant 0} \frac{2^n x^n}{n!}(1 + A)^{2^n-1} = e^x E(2x, A).$$

С другой стороны, $\dfrac{\partial}{\partial A}\left((1+A)E(x,A)\right) = (1+A)\dfrac{\partial E(x,A)}{\partial A} + E(x,A)$.

Продифференцируем выражение (6): $\dfrac{\partial E(x,A)}{\partial A} = e^{f(x)}\dfrac{\partial f(x,A)}{\partial A}$, выразим производную $f(x,A)$ через производную $E(x,A)$ и подставим полученное ранее соотношение

$$\frac{\partial f(x,A)}{\partial A} = e^x e^{f(2x,A)-f(x,A)} - A\frac{\partial f(x,A)}{\partial A} - 1. \tag{13}$$

В левой части (13) коэффициент перед мономом $A^k x^n/n!$ — число связных $(k+1)$-покрытий множества мощности n, а в правой части коэффициент перед таким же мономом содержит числа связных покрытий множеств мощности не больше n, в которых меньше k подмножеств [2, 7]. Используя разложение экспоненты в ряд Тейлора

$$e^x e^{f(2x,A)-f(x,A)} = \sum_{l=0}\frac{x^l}{l!}\sum_{m=0}\frac{1}{m!}\left(\sum_{n=0}\sum_{k=0}\frac{(2x)^n - x^n}{n!}{}_n\tilde{N}_k\right)^m,$$

получим явный вид системы рекуррентных соотношений

$${}_n\tilde{N}_{k+1} = \frac{1}{k+1}\sum_{l=0}^{n-1}\sum_{m=1}^{n-l}\left(\sum_{\substack{\sum i=n-l\\ \sum j=k}}\frac{n!}{(n-l)!}\frac{{}_{i_1}\tilde{N}_{j_1}}{i_1!}(2^{i_1}-1)\cdots\frac{{}_{i_m}\tilde{N}_{j_m}}{i_m!}(2^{i_m}-1)\right) - \frac{k}{k+1}({}_n\tilde{N}_k),$$

где в скобках — сумма по всем возможным комбинациям натуральных чисел $i_1 \ldots i_m$ и $j_1 \ldots j_m$, таких, что

$$\sum i = i_1 + \cdots + i_m = n-l, \quad \sum j = j_1 + \cdots + j_m = k.$$

Другая система соотношений показывает связь между числом покрытий, состоящих из k компонент связности, и числами связных покрытий. Найдём производную по y от обеих частей выражения (7):

$$\frac{\partial E(x;y)}{\partial y} = f(x)e^{f(x)y} = f(x)E(x;y).$$

Приравнивая в обеих частях коэффициенты перед мономами $x^n y^k$ и учитывая (8), получим соотношения $(k>0)$

$$({}^{k+1}_n\tilde{N}) = \frac{1}{k+1}\sum_{l=1}^{n-k} C_l^n({}^1_l\tilde{N})({}^k_{n-l}\tilde{N}). \tag{14}$$

Система соотношений (14) даёт процедуру получения числа покрытий, состоящих из k компонент связности $(k>1)$, при известных числах связных покрытий: число связных покрытий множества мощности 1 равно числу всех покрытий этого множества; для каждого $n>1$ по формулам (14) получаем все числа покрытий, состоящих из k компонент связности $(k>1)$; по формуле

$${}_n\tilde{N} = {}_nN - \sum_{k=2}^{n}{}^k_n\tilde{N}$$

вычисляем числа связных покрытий.

Заключение

В работе введены понятия связных покрытий и компонент связности покрытий, приведены выражения для экспоненциальных производящих функций последовательности чисел связных покрытий и показаны основные их свойства. На примере неповторяющихся покрытий рассмотрено, как с помощью преобразований выражений для производящих функций последовательности чисел связных покрытий можно получать рекуррентные соотношения. Получено несколько систем рекуррентных соотношений, на основе одной из них приведён последовательный алгоритм получения чисел связных покрытий и k-покрытий.

Использование связных покрытий и производящих функций последовательности чисел связных покрытий позволяет снизить объём вычислений чисел покрытий различных классов, среди которых минимальные покрытия и покрытия-антицепи, широко используемые в вычислениях количества булевых функций, в том числе монотонных [4].

ЛИТЕРАТУРА

1. *Ганопольский Р. М.* Число неупорядоченных покрытий конечного множества подмножествами фиксированного размера // Прикладная дискретная математика. 2010. № 4(10). С. 5–17.
2. *Ганопольский Р. М.* Производящие функции последовательности чисел покрытий конечного множества // Прикладная дискретная математика. 2011. № 1(11). С. 5–13.
3. *Macula A. J.* Covers of a finite set // Mathematics Magazine. 1994. V. 67. No. 2. P. 141–144.
4. *Comtet L.* Advanced combinatorics. The art of finite and infinite expansions. Dordrecht, Holland: D. Reidel Publishing Company, 1974.
5. *Стенли Р.* Перечислительная комбинаторика. Деревья, производящие функции и симметрические функции. М.: Мир, 2005.
6. *Харари Ф.* Теория графов. М.: УРСС, 2003.
7. *Ландо С. К.* Лекции о производящих функциях. М.: МЦНМО, 2002.

УДК 519.4

КОЛИЧЕСТВО ПОЯВЛЕНИЙ ЭЛЕМЕНТОВ В ВЫХОДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЯХ ФИЛЬТРУЮЩИХ ГЕНЕРАТОРОВ

О. В. Камловский

ООО «Центр сертификационных исследований», г. Москва, Россия

E-mail: ov-kam@yandex.ru

Получены оценки для числа r-грамм на отрезках выходных последовательностей фильтрующих генераторов над конечными полями. Оцениваются коэффициенты кросс-корреляции рассматриваемых последовательностей и приводятся условия, при которых последовательности, выработанные на различных начальных векторах, не совпадают.

Ключевые слова: *фильтрующие генераторы, конечные поля, линейные рекуррентные последовательности, суммы аддитивных характеров.*

Введение

Пусть $P = \mathrm{GF}(q)$ — конечное поле из q элементов, $F(x)$ — унитарный реверсивный многочлен (старший его коэффициент равен единице, а младший отличен от нуля) степени m над полем P, $f : P^k \to P$. Рассмотрим выходную последовательность $v = (v(i))_{i=0}^{\infty}$ фильтрующего генератора над полем P, удовлетворяющую соотношению

$$v(i) = f(u(i + t_1), u(i + t_2), \ldots, u(i + t_k)), \quad i \geqslant 0, \tag{1}$$

где t_1, t_2, …, t_k — фиксированный набор целых чисел, таких, что $0 \leqslant t_1 < t_2 < \ldots < t_k \leqslant m - 1$; $u = (u(i))_{i=0}^{\infty}$ — линейная рекуррентная последовательность (ЛРП) порядка m над полем P с характеристическим многочленом $F(x)$ [1–5].

Исследуются частотные характеристики последовательности v. Для каждого элемента $z \in P$ и натурального числа l обозначим через $N_l(z, v)$ количество появлений элемента $z \in P$ среди элементов $v(0)$, $v(1)$, …, $v(l - 1)$. В данной работе при некоторых ограничениях на начальный вектор $(u(0), \ldots, u(m - 1))$ ЛРП u и набор t_1, …, t_k (в частности, если $F(x)$ неприводим над полем P, то требуется лишь условие $(u(0), \ldots, u(m - 1)) \neq (0, \ldots, 0)$) доказывается, что для всех l, не превосходящих период $T(F)$ многочлена $F(x)$, выполнено неравенство

$$\left| N_l(z, v) - |f^{-1}(z)| \frac{l}{q^k} \right| \leqslant (q - 1) q^{\frac{k}{2} - 1} C_l(F), \tag{2}$$

где $f^{-1}(z)$ — множество всех прообразов элемента z при действии f, а величина $C_l(F)$ определена равенствами

$$C_l(F) = \begin{cases} \left(\dfrac{4}{\pi^2} \ln T(F) + \dfrac{9}{5} \right) q^{m/2}, & \text{если } l < T(F), \\ \left(3(q^m l - l^2) \right)^{1/3}, & \text{если } l < T(F) \text{ и } F(x) \text{ неприводим над } P, \\ (q^m - T(F))^{1/2}, & \text{если } l = T(F). \end{cases} \tag{3}$$

Оценка (2) нетривиальна (правая часть неравенства меньше l) при выполнении условий

$$l > (q-1)\left(\frac{4}{\pi^2}\ln T(F) + \frac{9}{5}\right) q^{(m+k)/2-1},$$

$$l \geqslant \sqrt{3} q^{m/2+3k/4},$$

$$l = T(F) \geqslant (q-1) q^{(m+k)/2-1}$$

соответственно.

Получено обобщение оценки (2) на случай r-грамм. Ранее аналогичные оценки для случая $q = 2$ и $T(F) = 2^m - 1$ были получены М. В. Левашовым и А. Б. Горчаковым.

Автору не известны другие общие оценки частот $N_l(z, v)$ при $l < T(F)$. В случае, когда $T(F) = 2^m - 1$, u — ЛРП максимального периода над полем $P = \mathrm{GF}(2)$ с характеристическим многочленом $F(x)$, а f задается многочленом от переменных x_1, …, x_k степени $\deg f$, набор $(v(0), \ldots, v(2^m - 2))$ является кодовым словом кода, эквивалентного коду Рида — Маллера порядка $\deg f$ с выколотой первой координатой [6]. Спектр весов кодов Рида — Маллера известен для значений $\deg f \in \{1, 2\}$, а также в некоторых диапазонах значений частот для произвольной степени f [6, § 15.3]. Из этих результатов следуют оценки частот $N_l(z, v)$ при $l = 2^m - 1$. Другие оценки для данных частот при фиксированном значении $\deg f$ получены в работе [7, теорема 1]. В [8, теоремы 1, 2] получены оценки частот появлений цепочек из подряд идущих нулей и единиц в ситуации, когда $q = 2$, $l = T(F) = 2^m - 1$, $f(x_1, \ldots, x_k) = x_1 x_l$, где $l \in \{2, 3, \ldots, k\}$.

Методом, применённым для доказательства оценки (2), получены оценки сверху для коэффициентов кросс-корреляции выходных последовательностей фильтрующих генераторов. С использованием этих оценок приводятся условия, при которых не совпадают последовательности, полученные на различных начальных векторах.

1. Условия на числа t_1, …, t_k и начальный вектор

Всюду в дальнейшем будем накладывать следующее условие на ЛРП u и набор чисел t_1, t_2, …, t_k: для всех ненулевых векторов $\bar{a} = (a_1, a_2, \ldots, a_k) \in P^k$ последовательность

$$\omega = a_1 x^{t_1} u + a_2 x^{t_2} u + \ldots + a_k x^{t_k} u,$$

элементы которой определены соотношением

$$\omega(i) = a_1 u(i + t_1) + a_2 u(i + t_2) + \ldots + a_k u(i + t_k), \quad i \geqslant 0,$$

имеет минимальный многочлен $M_\omega(x)$ равный $F(x)$. Минимальный многочлен последовательности ω связан с минимальным многочленом $M_u(x)$ ЛРП u следующим равенством [2, гл. 25, теорема 5], [9, лемма 1]:

$$M_\omega(x) = \frac{M_u(x)}{(M_u(x), a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k})}.$$

Таким образом, сформулированное выше условие на ЛРП u и числа t_1, …, t_k равносильно следующим равенствам:

$$M_u(x) = F(x),\ (F(x), a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k}) = e, \quad (a_1, \ldots, a_k) \in P^k \backslash \{\bar{0}\}. \qquad (4)$$

Приведём два примера, при которых соотношение (4) имеет место.

Утверждение 1. Пусть $F(x)$ — унитарный реверсивный многочлен степени m над полем P. Тогда

1) если $F(x)$ неприводим над полем P, то условие (4) выполнено тогда и только тогда, когда начальный вектор $(u(0), \ldots, u(m-1))$ ЛРП u ненулевой;
2) если $F(x)$ приводим и m_0 — наименьшая из степеней всех неприводимых делителей многочлена $F(x)$, $M_u(x) = F(x)$, а $t_k \leqslant m_0 + t_1 - 1$, то условие (4) выполнено.

Доказательство. Пункт 1 очевиден. Докажем пункт 2. Пусть $F_1(x)$, …, $F_t(x)$ — все различные неприводимые делители многочлена $F(x)$. Тогда для каждого $j \in \{1, 2, \ldots, t\}$ в силу реверсивности $F_j(x)$ получим

$$(F_j(x), a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k}) = (F_j(x), a_1 + a_2 x^{t_2 - t_1} + \ldots + a_k x^{t_k - t_1}).$$

Так как $t_k - t_1 \leqslant m_0 - 1 < \deg F_j(x)$, то для всех $(a_1, \ldots, a_k) \in P^k \backslash \{\bar{0}\}$ будем иметь $(F_j(x), a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k}) = e$. ■

Приведём другие соотношения, эквивалентные равенствам (4), полученные в работе [9, формула (10)]. Введём обозначение

$$V(F) = \{(a_1, \ldots, a_k) \in P^k : (F(x), a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k}) = e\}.$$

Пусть $F_1(x)$, …, $F_t(x)$ — все различные неприводимые делители многочлена $F(x)$. Для каждого $j \in \{1, 2, \ldots, t\}$ рассмотрим множество

$$W(F_j) = \{(a_1, \ldots, a_k) \in P^k : F_j(x) \mid a_1 x^{t_1} + a_2 x^{t_2} + \ldots + a_k x^{t_k}\}.$$

Оно является линейным пространством над полем P. Для всех целых чисел i_1, …, i_s, таких, что $1 \leqslant i_1 < \ldots < i_s \leqslant t$, обозначим через $d(i_1, \ldots, i_s)$ размерность линейного пространства $W(F_{i_1}) \cap \ldots \cap W(F_{i_s})$. Справедливо равенство

$$V(F) = P^k \backslash (W(F_1) \cup \ldots \cup W(F_t)),$$

и с использованием формулы для вычисления мощности объединения множеств будем иметь

$$|V(F)| = q^k + \sum_{s=1}^{t} (-1)^s \sum_{1 \leqslant i_1 < \ldots < i_s \leqslant t} |W(F_{i_1}) \cap \ldots \cap W(F_{i_s})| =$$
$$= q^k + \sum_{s=1}^{t} (-1)^s \sum_{1 \leqslant i_1 < \ldots < i_s \leqslant t} q^{d(i_1, \ldots, i_s)}.$$

Таким образом, условие (4) равносильно следующим соотношениям:

$$M_u(x) = F(x) \text{ и } \sum_{s=1}^{t} (-1)^s \sum_{1 \leqslant i_1 < \ldots < i_s \leqslant t} q^{d(i_1, \ldots, i_s)} = -1.$$

2. Оценки сумм характеров от линейных рекуррент

Рассмотрим χ — канонический аддитивный характер поля $P = \mathrm{GF}(q)$, определённый равенством

$$\chi(x) = e^{2\pi i \, \mathrm{tr}_p^q(x)/p}, \quad x \in P, \tag{5}$$

где i — мнимая комплексная единица; tr_p^q — функция следа из поля P в простое подполе $P_0 = \mathrm{GF}(p)$. Для каждой последовательности u над полем P и натурального числа l рассмотрим сумму

$$S_l(u) = \sum_{i=0}^{l-1} \chi(u(i)).$$

Приведём оценки величины $S_l(u)$.

Теорема 1. Пусть $P = \mathrm{GF}(q)$, $F(x)$ — унитарный реверсивный многочлен степени m над полем P. Тогда для каждой ЛРП u над полем P с минимальным многочленом $F(x)$ при всех $l \leqslant T(F)$ справедлива оценка

$$|S_l(u)| \leqslant C_l(F),$$

где величина $C_l(F)$ определена равенством (3).

Доказательство. В работе [5, теорема 8.81] при $l \leqslant T(F)$ получена оценка

$$|S_l(u)| \leqslant q^{\frac{m}{2}} \frac{1}{T(F)} \sum_{a=0}^{T(F)-1} \left| \sum_{k=0}^{l-1} e^{2\pi i\, ak/T(F)} \right|,$$

а как показано в работе [10, теорема 1],

$$\frac{1}{T(F)} \sum_{a=0}^{T(F)-1} \left| \sum_{k=0}^{l-1} e^{2\pi i\, ak/T(F)} \right| \leqslant \frac{4}{\pi^2} \ln T(F) + \frac{9}{5}.$$

Таким образом,

$$|S_l(u)| \leqslant \left(\frac{4}{\pi^2} \ln T(F) + \frac{9}{5} \right) q^{m/2}. \tag{6}$$

Неравенство

$$|S_l(u)| \leqslant \left(3(q^m l - l^2)\right)^{1/3} \tag{7}$$

для случая, когда P — простое поле, получено в работе [11, теорема 1]. Несложно заметить, что доказательство из этой работы справедливо для произвольного поля $P = \mathrm{GF}(q)$. При $l = T(F)$ неравенство

$$|S_l(u)| \leqslant (q^m - T(u))^{1/2} \tag{8}$$

вытекает из доказательства теоремы 8.78 из [5] (см. также упражнение 8.66). ■

Отметим, что оценки (6) и (8) несколько усиливают известные оценки Н. М. Коробова [12, теорема 1]. Оценка (6) является асимптотически неулучшаемой [9, теорема 3]. Оценка (7) нетривиальна (её правая часть меньше l) и точнее оценки (6) при условии

$$\sqrt{3} q^{m/2} \leqslant l \leqslant \frac{1}{3} \left(\frac{4}{\pi^2} \ln T(F) + \frac{9}{5} \right)^3 q^{m/2}.$$

3. Спектральные коэффициенты отображений

Пусть χ — канонический аддитивный характер поля P, определённый равенством (5). Через $\bar{\chi}$ обозначим характер, сопряжённый к характеру χ, т. е. $\bar{\chi}(x) = \overline{\chi(x)}$, $x \in P$, где черта означает комплексное сопряжение. Для каждого вектора $\bar{a} = (a_1, \ldots, a_k) \in P^k$ рассмотрим отображение $\chi_{\bar{a}}$, осуществляемое по правилу

$$\chi_{\bar{a}}(x_1, \ldots, x_k) = \chi(a_1 x_1 + \ldots + a_k x_k), \quad x_1, \ldots, x_k \in P. \tag{9}$$

Несложно показать, что $\chi_{\bar a}$ является характером группы $(P^k, +)$. Обозначим через $\bar a\bar b$ скалярное произведение векторов $\bar a$, $\bar b \in P^k$, т. е.

$$\bar a\bar b = (a_1, \ldots, a_k)(b_1, \ldots, b_k) = a_1b_1 + \ldots + a_kb_k.$$

В этих обозначениях равенство (9) запишется в виде

$$\chi_{\bar a}(\bar x) = \chi(\bar a\bar x), \quad \bar x = (x_1, \ldots, x_k) \in P^k.$$

Приведём вспомогательный результат из работы [5, соотношение (5.9)].

Лемма 1.

$$\sum_{c\in P}\chi(cx) = \begin{cases} q, & \text{если } x = 0, \\ 0, & \text{если } x \in P\backslash\{0\}. \end{cases}$$

Нам понадобятся следующие свойства спектральных коэффициентов отображений.

Утверждение 2. Пусть ψ — произвольное отображение из P^k в мультипликативную группу поля комплексных чисел. Тогда

1) для всех x_1, …, $x_k \in P$ выполнено равенство

$$\psi(x_1, \ldots, x_k) = \frac{1}{q^k}\sum_{\bar a\in P^k} W_\psi(\bar a)\chi_{\bar a}(x_1, \ldots, x_k), \tag{10}$$

где для всех $\bar a \in P^k$ комплексные числа $W_\psi(\bar a)$ (спектральные коэффициенты) определяются по правилу

$$W_\psi(\bar a) = \sum_{\bar b=(b_1,\ldots,b_k)\in P^k} \psi(\bar b)\bar\chi(\bar a\bar b); \tag{11}$$

2) если при этом $|\psi(\bar x)| = 1$ для всех $\bar x \in P^k$, то для чисел, определённых равенством (11), имеет место соотношение

$$\sum_{\bar a\in P^k} |W_\psi(\bar a)|^2 = q^{2k};$$

3) если в условиях п. 2 $M(\psi) = \{\bar a \in P^k : W_\psi(\bar a) \neq 0\}$, то справедливо неравенство

$$\sum_{\bar a\in P^k} |W_\psi(\bar a)| \leqslant |M(\psi)|^{1/2}q^k;$$

4) в условиях п. 2 справедливо неравенство

$$\sum_{\bar a\in P^k} |W_\psi(\bar a)| \leqslant q^{3k/2}.$$

Доказательство.

1) Пусть комплексные числа $W_\psi(\bar a)$ определены равенством (11). Тогда

$$\frac{1}{q^k}\sum_{\bar a\in P^k} W_\psi(\bar a)\chi_{\bar a}(x_1, \ldots, x_k) = \frac{1}{q^k}\sum_{\bar a\in P^k}\chi(\bar a\bar x)\sum_{\bar b\in P^k}\psi(\bar b)\bar\chi(\bar a\bar b) = \frac{1}{q^k}\sum_{\bar b\in P^k}\psi(\bar b)\sum_{\bar a\in P^k}\chi(\bar a(\bar x - \bar b)).$$

С использованием леммы 1 получим

$$\begin{aligned}\sum_{\bar a\in P^k}\chi(\bar a(\bar x - \bar b)) &= \left(\sum_{a_1\in P}\chi(a_1(x_1 - b_1))\right)\ldots\left(\sum_{a_k\in P}\chi(a_k(x_k - b_k))\right) = \\ &= \begin{cases} q^k, & \text{если } \bar x = \bar b, \\ 0, & \text{если } \bar x \neq \bar b. \end{cases}\end{aligned} \tag{12}$$

Таким образом,

$$\frac{1}{q^k}\sum_{\bar a\in P^k} W_\psi(\bar a)\chi_{\bar a}(x_1,\dots,x_k)=\psi(\bar x)=\psi(x_1,\dots,x_k).$$

2) Из равенства (11) имеем

$$\sum_{\bar a\in P^k}|W_\psi(\bar a)|^2=\sum_{\bar a\in P^k}W_\psi(\bar a)\overline{W_\psi(\bar a)}=\sum_{\bar a\in P^k}\left(\sum_{\bar b\in P^k}\psi(\bar b)\bar\chi(\bar a\bar b)\right)\left(\sum_{\bar c\in P^k}\overline{\psi(\bar c)}\chi(\bar a\bar c)\right)=$$
$$=\sum_{\bar b\in P^k}\sum_{\bar c\in P^k}\psi(\bar b)\overline{\psi(\bar c)}\sum_{\bar a\in P^k}\chi(\bar a(\bar c-\bar b)).$$

Используя соотношение (12), получим

$$\sum_{\bar a\in P^k}|W_\psi(\bar a)|^2=q^k\sum_{\bar b\in P^k}\psi(\bar b)\overline{\psi(\bar b)}=q^k\sum_{\bar b\in P^k}|\psi(\bar b)|^2=q^{2k}.$$

3) Согласно неравенству Коши и п. 2, будем иметь

$$\sum_{\bar a\in P^k}|W_\psi(\bar a)|=\sum_{\bar a\in M(\psi)}|W_\psi(\bar a)|\leqslant\left(\sum_{\bar a\in M(\psi)}|W_\psi(\bar a)|^2\right)^{1/2}\left(\sum_{\bar a\in M(\psi)}1\right)^{1/2}=q^k|M(\psi)|^{1/2}.$$

4) Непосредственно следует из п. 3. ■

Равенства (10) и (11) являются известными разложениями функций по базису характеров [13, соотношение (12)]. Пункт 2 устанавливает хорошо известное равенство Парсеваля [13, равенство (16)], [14, утверждение 1]. Данные результаты приведены лишь для полноты изложения материала.

4. Оценки числа появлений элементов на отрезках выходных последовательностей

Получим теперь оценки для числа $N_l(z,v)$ появлений элемента $z\in P$ среди элементов $v(0)$, $v(1)$, …, $v(l-1)$ последовательности v, построенной по правилу (1).

Теорема 2. Пусть $F(x)$ — унитарный реверсивный многочлен степени m над полем $P=\mathrm{GF}(q)$, u — ЛРП над полем P с характеристическим многочленом $F(x)$ и выполнены условия (4). Тогда для всех $z\in P$ и $l\in\mathbb{N}$, таких, что $l\leqslant T(F)$, справедлива оценка

$$\left|N_l(z,v)-|f^{-1}(z)|\frac{l}{q^k}\right|\leqslant(q-1)q^{k/2-1}C_l(F),\tag{13}$$

где $f^{-1}(z)=\{(x_1,\dots,x_k)\in P^k : f(x_1,\dots,x_k)=z\}$.

Доказательство. С использованием леммы 1 получим

$$N_l(z,v)=\frac{1}{q}\sum_{c\in P}\chi(-cz)\sum_{i=0}^{l-1}\chi(cv(i))$$

и, следовательно,

$$N_l(z,v)=\frac{l}{q}+\frac{1}{q}\sum_{c\in P^*}\chi(-cz)\sum_{i=0}^{l-1}\chi(cv(i)).$$

Для каждого $c\in P$ рассмотрим отображение ψ_c, осуществляемое по правилу

$$\psi_c(x_1,\dots,x_k)=\chi(cf(x_1,\dots,x_k))=e^{2\pi i\,\mathrm{tr}_p^q(cf(x_1,\dots,x_k))/p},\quad x_1,\dots,x_k\in P.\tag{14}$$

Тогда

$$\psi_c(u(i+t_1),\dots,u(i+t_k)) = \chi(cf(u(i+t_1),\dots,u(i+t_k))) = \chi(cv(i)), \quad i \geqslant 0,$$

а значит,

$$N_l(z,v) = \frac{l}{q} + \frac{1}{q}\sum_{c\in P^*}\chi(-cz)\sum_{i=0}^{l-1}\psi_c(u(i+t_1),\dots,u(i+t_k)).$$

С использованием равенства (10) будем иметь

$$N_l(z,v) = \frac{l}{q} + \frac{1}{q}\sum_{c\in P^*}\chi(-cz)\sum_{i=0}^{l-1}\frac{1}{q^k}\sum_{\bar a\in P^k} W_{\psi_c}(\bar a)\chi_{\bar a}(u(i+t_1),\dots,u(i+t_k)).$$

Изменив порядок суммирования, получим

$$N_l(z,v) = \frac{l}{q} + \frac{1}{q^{k+1}}\sum_{c\in P^*}\chi(-cz)\sum_{\bar a\in P^k} W_{\psi_c}(\bar a)\sum_{i=0}^{l-1}\chi_{\bar a}(u(i+t_1),\dots,u(i+t_k)).$$

Выделяя отдельно слагаемое, соответствующее $\bar a = \bar 0$, будем иметь

$$N_l(z,v) = \frac{l}{q} + \frac{l}{q^{k+1}}\sum_{c\in P^*}\chi(-cz)W_{\psi_c}(\bar 0) + \frac{1}{q^{k+1}}\sum_{c\in P^*}\chi(-cz)\sum_{\bar a\in P^k\setminus\{\bar 0\}} W_{\psi_c}(\bar a)\sum_{i=0}^{l-1}\chi(u_{\bar a}(i)), \quad (15)$$

где для каждого $\bar a = (a_1,\dots,a_k) \in P^k$ через $u_{\bar a}$ обозначена последовательность над полем P с элементами $u_{\bar a}(i) = a_1u(i+t_1) + \dots + a_ku(i+t_k)$, $i \geqslant 0$.

Из равенств (11), (14), леммы 1 и соотношения $W_{\psi_0}(\bar 0) = q^k$ следует, что

$$\begin{aligned}\frac{l}{q} + \frac{l}{q^{k+1}}\sum_{c\in P^*}\chi(-cz)W_{\psi_c}(\bar 0) &= \frac{l}{q^{k+1}}\sum_{c\in P}\chi(-cz)\sum_{b_1,\dots,b_k\in P}\psi_c(b_1,\dots,b_k) = \\ &= \frac{l}{q^{k+1}}\sum_{c\in P}\chi(-cz)\sum_{b_1,\dots,b_k\in P}\chi(cf(b_1,\dots,b_k)) = \\ &= \frac{l}{q^{k+1}}\sum_{b_1,\dots,b_k\in P}\sum_{c\in P}\chi(c(f(b_1,\dots,b_k)-z)) = \frac{l}{q^k}|f^{-1}(z)|.\end{aligned}$$

Тогда из равенства (15) получим

$$\left|N_l(z,v) - |f^{-1}(z)|\frac{l}{q^k}\right| \leqslant \frac{1}{q^{k+1}}\sum_{c\in P^*}\sum_{\bar a\in P^k\setminus\{\bar 0\}}|W_{\psi_c}(\bar a)|\max_{\bar a\in P^k\setminus\{\bar 0\}}\left|\sum_{i=0}^{l-1}\chi(u_{\bar a}(i))\right|.$$

Заметим, что по условию (4) для всех $\bar a \in P^k\setminus\{\bar 0\}$ ЛРП $u_{\bar a}$ имеет минимальный многочлен $F(x)$, поэтому согласно теореме 1

$$\left|N_l(z,v) - |f^{-1}(z)|\frac{l}{q^k}\right| \leqslant \frac{C_l(F)}{q^{k+1}}\sum_{c\in P^*}\sum_{\bar a\in P^k\setminus\{\bar 0\}}|W_{\psi_c}(\bar a)|. \quad (16)$$

Остаётся воспользоваться п. 4 утверждения 2. ■

Отметим, что если f — сбалансированное отображение, т. е. $|f^{-1}(z)| = q^{k-1}$ для всех $z \in P$, то величина $|f^{-1}(z)|l/q^k$ равна «естественному» среднему значению для числа появлений элемента z на отрезке длины l последовательности v.

Правая часть оценки из теоремы 2 при $l = T(F)$ будет меньше периода $T(F)$ многочлена $F(x)$ при условии $(q-1)q^{k/2-1}(q^m - T(F))^{1/2} < T(F)$, которое заведомо выполнено при $T(F) \geqslant (q-1)q^{(m+k)/2-1}$.

При $l < T(F)$ правая часть оценки из теоремы 2 меньше l тогда и только тогда, когда

$$l > (q-1)\left(\frac{4}{\pi^2}\ln T(F) + \frac{9}{5}\right)q^{(m+k)/2-1} \text{ и } l > (q-1)\left(3(q^m l - l^2)\right)^{1/3} q^{k/2-1}$$

соответственно. Второе из этих неравенств заведомо выполнено, если $l \geqslant (3q^m l)^{1/3} q^{k/2}$, т. е. при $l \geqslant \sqrt{3}q^{m/2+3k/4}$.

В случае, когда для отображений ψ_c, $c \in P^*$, сопоставленных функции f по правилу (14), известны коэффициенты $W_{\psi_c}(\bar{a})$, $\bar{a} \in P^k$, оценку из теоремы 2 можно уточнить, воспользовавшись неравенством (16). Отсюда с использованием п. 3 утверждения 2 получим также неравенство

$$\left| N_l(z, v) - |f^{-1}(z)|\frac{l}{q^k} \right| \leqslant \frac{(q-1)C_l(F)}{q} \max_{c \in P^*} |M(\psi_c)|^{1/2}. \tag{17}$$

Для случая $q = 2$ в оценках (16) и (17) рассматривается только значение $c = e$, и в этой ситуации коэффициенты $W_{\psi_c}(\bar{a})$ совпадают с коэффициентами Уолша — Адамара булевой функции $f(x_1, \ldots, x_k)$ [15, с. 77]. Если f является бент-функцией, то оценки (13), (16) и (17) совпадают. В случае, когда f имеет нечётный вес, выполнено равенство $M(\psi_e) = P^k$ и оценка (13) совпадает с оценкой (17). Если f — платовидная функция порядка $2t$ [15, § 6.5], то $W_{\psi_e}(\bar{a}) \in \{0, \pm 2^{k-t}\}$ для всех $\bar{a} \in P^k$, $|M(\psi_e)| = 2^{2t}$, а значит, оценки (16) и (17) совпадают, причём они точнее оценки (13).

Как показано в работе [16, теорема 2], при случайном равновероятном выборе булевой функции $f(x_1, \ldots, x_k)$ с чётным весом случайная величина η_k, равная числу нулевых коэффициентов Уолша — Адамара, имеет математическое ожидание, удовлетворяющее при $k \to \infty$ условию $E\eta_k \sim (2^{k+3}/\pi)^{1/2}$. Таким образом, при больших k для почти всех булевых функций f чётного веса правая часть неравенства (17) равна

$$C_l(F)\left(2^{k-2} - \left(\frac{2^{k-1}}{\pi}\right)^{1/2}\right)^{1/2},$$

и при $k \to \infty$ она эквивалентна правой части неравенства (13).

5. Коэффициенты кросс-корреляции

Пусть $P = \mathrm{GF}(q)$, $F(x)$ — унитарный реверсивный многочлен степени m над полем P, u_1, u_2 — ЛРП над полем P с характеристическим многочленом $F(x)$, $f_1 : P^k \to P$, $f_2 : P^s \to P$. Рассмотрим последовательности v_1 и v_2, полученные по правилу

$$v_1(i) = f_1(u_1(i + t_1), \ldots, u_1(i + t_k)), \quad v_2(i) = f_2(u_2(i + l_1), \ldots, u_2(i + l_s)), \quad i \geqslant 0,$$

где t_1, …, t_k, l_1, …, l_s — некоторые целые неотрицательные числа.

Определим коэффициент кросс-корреляции $C_{v_1,v_2}(l, t)$ последовательностей v_1 и v_2 равенством

$$C_{v_1,v_2}(l, t) = \sum_{i=0}^{l-1} \chi(v_1(i) - v_2(i + t)), \quad t = 0, 1, \ldots, T(F) - 1,$$

где χ — аддитивный характер поля P, заданный равенством (5). В случае $l = T(F)$ функция $C_{v_1,v_2}(l,t)$ от переменной t называется кросс-корреляционной функцией [5, с. 579]. Величина $|C_{v_1,v_2}(l,t)|$ характеризует близость начальных отрезков длины l последовательностей v_1 и $x^t v_2$.

Будем говорить, что система ЛРП $\omega_1, \ldots, \omega_n$ над полем P с характеристическим многочленом $F(x) \in P[x]$ *сильно линейно независима*, если для всех $(c_1, \ldots, c_n) \in \in P^n \backslash \{\bar{0}\}$ последовательность $c_1\omega_1 + \ldots + c_n\omega_n$ имеет минимальный многочлен $F(x)$. Если $\Phi_1(x), \ldots, \Phi_n(x)$ — генераторы ЛРП $\omega_1, \ldots, \omega_n$ [2, глава 25, § 2] относительно характеристического многочлена $F(x)$ соответственно, то сильная линейная независимость равносильна соотношениям

$$(c_1\Phi_1(x) + \ldots + c_n\Phi_n(x), F(x)) = e \text{ для всех } \bar{c} = (c_1, \ldots, c_n) \in P^n \backslash \{\bar{0}\}.$$

Определим отображения ψ_1 и ψ_2 равенствами

$$\psi_1(x_1, \ldots, x_k) = \chi(f_1(x_1, \ldots, x_k)), \quad x_1, \ldots, x_k \in P,$$
$$\psi_2(y_1, \ldots, y_k) = \chi(-f_2(y_1, \ldots, y_s)), \quad y_1, \ldots, y_s \in P.$$

Теорема 3. Пусть система

$$x^{t_1}u_1,\ x^{t_2}u_1,\ \ldots,\ x^{t_k}u_1,\ x^{l_1+t}u_2,\ x^{l_2+t}u_2,\ \ldots,\ x^{l_s+t}u_2$$

сильно линейно независима над полем P. Тогда для всех $l \leqslant T(F)$

$$\left| C_{v_1,v_2}(l,t) - \frac{l}{q^{k+s}} W_{\psi_1}(\bar{0}) W_{\psi_2}(\bar{0}) \right| \leqslant q^{\frac{k+s}{2}} C_l(F),$$

где

$$W_{\psi_1}(\bar{0}) = \sum_{(a_1,\ldots,a_k)\in P^k} \chi(f_1(a_1, \ldots, a_k)), \quad W_{\psi_2}(\bar{0}) = \sum_{(b_1,\ldots,b_s)\in P^s} \chi(-f_2(b_1, \ldots, b_s)),$$

а величина $C_l(F)$ определена соотношениями (3).

Доказательство. Из равенства

$$C_{v_1,v_2}(l,t) = \sum_{i=0}^{l-1} \psi_1(u_1(i+t_1), \ldots, u_1(i+t_k))\psi_2(u_2(i+l_1+t), \ldots, u_2(i+l_s+t))$$

с использованием формулы (10) будем иметь

$$C_{v_1,v_2}(l,t) = \frac{1}{q^{k+s}} \sum_{\bar{a}\in P^k, \bar{b}\in P^s} W_{\psi_1}(\bar{a}) W_{\psi_2}(\bar{b}) \sum_{i=0}^{l-1} \chi(w_{\bar{a},\bar{b}}(i)),$$

где $w_{\bar{a},\bar{b}}$ — последовательность, определённая для всех $\bar{a} = (a_1, \ldots, a_k)$, $\bar{b} = (b_1, \ldots, b_s)$ равенством

$$w_{\bar{a},\bar{b}}(i) = a_1u_1(i+t_1) + \ldots + a_ku_1(i+t_k) + b_1u_2(i+l_1+t) + \ldots + b_su_2(i+l_s+t), \quad i \geqslant 0.$$

Поэтому

$$C_{v_1,v_2}(l,t) - \frac{l}{q^{k+s}} W_{\psi_1}(\bar{0}) W_{\psi_2}(\bar{0}) = \frac{1}{q^{k+s}} \sum_{(\bar{a},\bar{b})\neq(\bar{0},\bar{0})} W_{\psi_1}(\bar{a}) W_{\psi_2}(\bar{b}) \sum_{i=0}^{l-1} \chi(w_{\bar{a},\bar{b}}(i)),$$

где суммирование осуществляется по всем наборам $(\bar{a}, \bar{b}) \in P^k \times P^s$, для которых $\bar{a} \neq \bar{0}$ или $\bar{b} \neq \bar{0}$. Учитывая, что для всех рассматриваемых наборов минимальный многочлен последовательности $w_{\bar{a},\bar{b}}$ равен $F(x)$, с использованием теоремы 1 и п. 4 утверждения 2 будем иметь

$$\left|C_{v_1,v_2}(l,t) - \frac{l}{q^{k+s}} W_{\psi_1}(\bar{0}) W_{\psi_2}(\bar{0})\right| \leqslant \frac{1}{q^{k+s}} \left(\sum_{\bar{a}\in P^k} |W_{\psi_1}(\bar{a})|\right) \left(\sum_{\bar{b}\in P^s} |W_{\psi_2}(\bar{b})|\right) C_l(F) \leqslant q^{(k+s)/2} C_l(F).$$

Теорема доказана. ■

Следствие 1. Пусть в условиях теоремы 3 хотя бы одно из отображений f_1 или f_2 сбалансировано. Тогда

$$|C_{v_1,v_2}(l,t)| \leqslant q^{(k+s)/2} C_l(F).$$

Доказательство. Пусть отображение f_1 сбалансировано. Согласно равенству (11) и лемме 1,

$$W_{\psi_1}(\bar{0}) = \sum_{a_1,\ldots,a_k\in P} \chi(f_1(a_1,\ldots,a_k)) = q^{k-1} \left(\sum_{a\in P} \chi(a)\right) = 0.$$

Аналогично, если f_2 сбалансировано, то $W_{\psi_2}(\bar{0}) = 0$. ■

Следствие 2. Пусть система

$$x^{t_1}u_1,\ x^{t_2}u_1,\ \ldots,\ x^{t_k}u_1,\ x^{l_1}u_2,\ x^{l_2}u_2,\ \ldots,\ x^{l_s}u_2$$

сильно линейно независима над полем P и хотя бы одно из отображений f_1, f_2 сбалансировано. Тогда при выполнении каждого из двух условий

1) $T(F) \geqslant q^{(m+k+s)/2}$,

2) $F(x)$ — многочлен максимального периода и $k + s < 2m$

последовательности v_1 и v_2 различны.

Доказательство. Допустим $v_1 = v_2$, тогда, согласно следствию 1, $T(F) = |C_{v_1,v_2}(T(F),0)| \leqslant q^{(k+s)/2}(q^m - T(F))^{1/2}$, что противоречит условию 1. Если $F(x)$ — многочлен максимального периода, то по следствию 1 получим $T(F) = q^m - 1 = |C_{v_1,v_2}(T(F),0)| \leqslant q^{(k+s)/2}$, что противоречит условию 2. ■

6. Обобщения на случай r-грамм

Пусть всюду далее $F(x)$ — унитарный реверсивный многочлен степени m над полем $P = \mathrm{GF}(q)$, v — выходная последовательность фильтрующего генератора, удовлетворяющая равенству (1), $l \in \mathbb{N}$. Для набора $\bar{s} = (s_1,\ldots,s_r)$ целых неотрицательных чисел и r-граммы $\bar{z} = (z_1,\ldots,z_r) \in P^r$ обозначим через $N_l(\bar{z},\bar{s},v)$ количество чисел $i \in \{0,1,\ldots,l-1\}$, таких, что

$$\begin{cases} v(i+s_1) = z_1, \\ v(i+s_2) = z_2, \\ \qquad\ldots \\ v(i+s_r) = z_r, \end{cases}$$

т. е. $N_l(\bar{z},\bar{s},v)$ — число появлений $\bar{z}$ среди r-грамм $(v(i+s_1),\ldots,v(i+s_r))$, где $i \in \{0,1,\ldots,l-1\}$. Потребуем, чтобы числа t_1, …, t_k и s_1, …, s_r были выбраны так, что система

$$x^{s_1+t_1}u,\ \ldots,\ x^{s_1+t_k}u,\ x^{s_2+t_1}u,\ \ldots,\ x^{s_2+t_k}u,\ \ldots,\ x^{s_r+t_1}u,\ \ldots,\ x^{s_r+t_k}u \qquad (18)$$

сильно линейно независима над полем P. Данное требование равносильно тому, что $M_u(x) = F(x)$ и для всех элементов $a_1^{(1)}, \ldots, a_k^{(1)}, \ldots, a_1^{(r)}, \ldots, a_k^{(r)} \in P$, не равных нулю одновременно, выполнено соотношение

$$(a_1^{(1)}x^{s_1+t_1} + \ldots + a_k^{(1)}x^{s_1+t_k} + \ldots + a_1^{(r)}x^{s_r+t_1} + \ldots + a_k^{(r)}x^{s_r+t_k}, F(x)) = e.$$

В частности, условие сильной линейной независимости системы (18) означает, что числа $s_1+t_1, \ldots, s_1+t_k, s_2+t_1, \ldots, s_2+t_k, \ldots, s_r+t_1, \ldots, s_r+t_k$ попарно различны и $rk \leqslant m$. В случае, когда $s_1 = 0$, $s_2 = 1$, $\ldots$, $s_r = r-1$, попарное различие рассматриваемых чисел имеет место тогда и только тогда, когда

$$t_2 \geqslant t_1 + r,\ t_3 \geqslant t_2 + r,\ \ldots,\ t_k \geqslant t_{k-1} + r.$$

Лемма 2. Пусть χ — канонический аддитивный характер поля $P = \mathrm{GF}(q)$, задаваемый равенством (5), а ψ_c — отображение, определённое соотношением (14), где $c \in P$. Тогда

$$\frac{l}{q^{(k+1)r}} \sum_{c_1,\ldots,c_r \in P} \chi(-c_1z_1 - \ldots - c_rz_r) W_{\psi_{c_1}}(\bar{0}) \cdots W_{\psi_{c_r}}(\bar{0}) = \frac{l}{q^{kr}} |f^{-1}(z_1)| \cdots |f^{-1}(z_r)|.$$

Доказательство. С использованием равенства (11) получим

$$\frac{l}{q^{(k+1)r}} \sum_{c_1,\ldots,c_r \in P} \chi(-c_1z_1 - \ldots - c_rz_r) W_{\psi_{c_1}}(\bar{0}) \cdots W_{\psi_{c_r}}(\bar{0}) =$$

$$= \frac{l}{q^{(k+1)r}} \sum_{c_1,\ldots,c_r \in P} \chi(-c_1z_1 - \ldots - c_rz_r) \times$$

$$\times \left(\sum_{b_1^{(1)},\ldots,b_k^{(1)} \in P} \chi(c_1 f(b_1^{(1)}, \ldots, b_k^{(1)})) \right) \cdots \left(\sum_{b_1^{(r)},\ldots,b_k^{(r)} \in P} \chi(c_r f(b_1^{(r)}, \ldots, b_k^{(r)})) \right) =$$

$$= \frac{l}{q^{(k+1)r}} \left(\sum_{b_1^{(1)},\ldots,b_k^{(1)} \in P} \sum_{c_1 \in P} \chi(c_1(f(b_1^{(1)}, \ldots, b_k^{(1)}) - z_1)) \right) \cdots$$

$$\cdots \left(\sum_{b_1^{(r)},\ldots,b_k^{(r)} \in P} \sum_{c_r \in P} \chi(c_r(f(b_1^{(r)}, \ldots, b_k^{(r)}) - z_r)) \right).$$

Для завершения доказательства остаётся воспользоваться леммой 1. ■

Теорема 4. Пусть $F(x)$ — унитарный реверсивный многочлен степени m над полем $P = \mathrm{GF}(q)$, v — последовательность, удовлетворяющая равенству (1). Тогда если система ЛРП (18) сильно линейно независима над полем P, то для всех $l \leqslant T(F)$ и $\bar{z} \in P^r$ справедлива оценка

$$\left| N_l(\bar{z}, \bar{s}, v) - |f^{-1}(z_1)| \cdots |f^{-1}(z_r)| \frac{l}{q^{kr}} \right| \leqslant \frac{((q-1)q^{k/2}+1)^r - 1}{q^r} C_l(F),$$

где $C_l(F)$ — величина, определённая равенством (3).

Доказательство. С использованием леммы 1 получим

$$N_l(\bar{z}, \bar{s}, v) = \sum_{i=0}^{l-1} \left(\frac{1}{q} \sum_{c_1 \in P} \chi(c_1(v(i+s_1) - z_1)) \right) \cdots \left(\frac{1}{q} \sum_{c_r \in P} \chi(c_r(v(i+s_r) - z_r)) \right).$$

Отсюда будем иметь

$$N_l(\bar{z},\bar{s},v)=\frac{1}{q^r}\sum_{(c_1,\ldots,c_r)\in P^r}\chi(-\bar{c}\bar{z})\sum_{i=0}^{l-1}\chi(c_1v(i+s_1))\cdots\chi(c_rv(i+s_r)),$$

где $\bar{c}\bar{z}=(c_1,\ldots,c_r)(z_1,\ldots,z_r)=c_1z_1+\ldots+c_rz_r$. Выделяя слагаемое, соответствующее случаю $(c_1,\ldots,c_r)=(0,\ldots,0)=\bar{0}$, получим

$$N_l(\bar{z},\bar{s},v)-\frac{l}{q^r}=\frac{1}{q^r}\sum_{\bar{c}\in P^r\setminus\{\bar{0}\}}\chi(-\bar{c}\bar{z})D_l(\bar{c}),\tag{19}$$

где

$$D_l(\bar{c})=\sum_{i=0}^{l-1}\chi(c_1v(i+s_1))\cdots\chi(c_rv(i+s_r)).\tag{20}$$

Обозначим через $|\bar{c}|$ вес (количество ненулевых координат) вектора $\bar{c}\in P^r$. Тогда равенство (19) запишется в виде

$$N_l(\bar{z},\bar{s},v)-\frac{l}{q^r}=\frac{1}{q^r}\sum_{d=1}^{r}\sum_{\substack{\bar{c}\in P^r,\\|\bar{c}|=d}}\chi(-\bar{c}\bar{z})D_l(\bar{c}).\tag{21}$$

Пусть вектор $\bar{c}=(c_1,\ldots,c_r)\in P^r$ имеет вес d, ненулевые элементы в нём стоят на местах j_1, …, j_d и выполнено равенство $c_{j_1}=h_1$, …, $c_{j_d}=h_d$. Тогда с использованием равенств (14) и (20) получим

$$D_l(\bar{c})=\sum_{i=0}^{l-1}\psi_{h_1}(u(i+s_{j_1}+t_1),\ldots,u(i+s_{j_1}+t_k))\cdots\psi_{h_d}(u(i+s_{j_d}+t_1),\ldots,u(i+s_{j_d}+t_k)).$$

Используя равенство (10), $D_l(\bar{c})$ запишем следующим образом:

$$D_l(\bar{c})=\sum_{i=0}^{l-1}\left(\tfrac{1}{q^k}\sum_{\bar{a}_1\in P^k}W_{\psi_{h_1}}(\bar{a}_1)\chi_{\bar{a}_1}(u(i+s_{j_1}+t_1),\ldots,u(i+s_{j_1}+t_k))\right)\cdots$$
$$\cdots\left(\frac{1}{q^k}\sum_{\bar{a}_d\in P^k}W_{\psi_{h_d}}(\bar{a}_d)\chi_{\bar{a}_d}(u(i+s_{j_d}+t_1),\ldots,u(i+s_{j_d}+t_k))\right).$$

Значит,

$$D_l(\bar{c})=\frac{1}{q^{kd}}\sum_{\bar{a}_1,\ldots,\bar{a}_d\in P^k}W_{\psi_{h_1}}(\bar{a}_1)\cdots W_{\psi_{h_d}}(\bar{a}_d)\sum_{i=0}^{l-1}\chi(\omega_{\bar{a}_1,\ldots,\bar{a}_d}(i)),\tag{22}$$

где $\omega_{\bar{a}_1,\ldots,\bar{a}_d}$ — последовательность над полем P, определённая для всех векторов $\bar{a}_1=(a_1^{(1)},\ldots,a_k^{(1)})$, …, $\bar{a}_d=(a_1^{(d)},\ldots,a_k^{(d)})$ равенством

$$\omega_{\bar{a}_1,\ldots,\bar{a}_d}(i)=a_1^{(1)}u(i+s_{j_1}+t_1)+\ldots+a_k^{(1)}u(i+s_{j_1}+t_k)+\ldots$$
$$\ldots+a_1^{(d)}u(i+s_{j_d}+t_1)+\ldots+a_k^{(d)}u(i+s_{j_d}+t_k),\quad i\geqslant 0.$$

Так как система (18) сильно линейно независима над полем P, то минимальный многочлен последовательности $\omega_{\bar{a}_1,\ldots,\bar{a}_d}$ равен $F(x)$, за исключением случая, когда $\bar{a}_1=\ldots=\bar{a}_d=\bar{0}$. Выделив отдельно в равенстве (22) слагаемое, соответствующее этому случаю, будем иметь

$$D_l(\bar{c})=\frac{l}{q^{kd}}W_{\psi_{h_1}}(\bar{0})\cdots W_{\psi_{h_d}}(\bar{0})+D_l^*(\bar{c}),\tag{23}$$

где

$$D_l^*(\bar{c}) = \frac{1}{q^{kd}} \sum_{\bar{a}_1,\ldots,\bar{a}_d \in P^k \setminus \{(\bar{0},\ldots,\bar{0})\}} W_{\psi_{h_1}}(\bar{a}_1)\cdots W_{\psi_{h_d}}(\bar{a}_d) \sum_{i=0}^{l-1} \chi(\omega_{\bar{a}_1,\ldots,\bar{a}_d}(i)). \tag{24}$$

Подставив равенство (23) в соотношение (21), получим

$$N_l(\bar{z},\bar{s},v) - \delta_l(\bar{z}) = \frac{1}{q^r} \sum_{d=1}^{r} \sum_{\substack{\bar{c}\in P^r,\\ |\bar{c}|=d}} \chi(-\bar{c}\bar{z}) D_l^*(\bar{c}), \tag{25}$$

где $\delta_l(\bar{z}) = \dfrac{l}{q^r} + \dfrac{1}{q^r} \sum\limits_{d=1}^{r} \sum\limits_{\substack{\bar{c}\in P^r,\\ |\bar{c}|=d}} \chi(-\bar{c}\bar{z}) \dfrac{l}{q^{kd}} W_{\psi_{h_1}}(\bar{0})\cdots W_{\psi_{h_d}}(\bar{0})$.

Для вычисления $\delta_l(\bar{z})$ воспользуемся леммой 2 и равенством $W_{\psi_0}(\bar{0}) = q^k$:

$$\begin{aligned}\delta_l(\bar{z}) &= \frac{l}{q^{(k+1)r}} \underbrace{W_{\psi_0}(\bar{0})\cdots W_{\psi_0}(\bar{0})}_{r} + \frac{l}{q^{(k+1)r}} \sum_{\bar{c}\in P^r\setminus\{\bar{0}\}} \chi(-\bar{c}\bar{z}) W_{\psi_{c_1}}(\bar{0})\cdots W_{\psi_{c_r}}(\bar{0}) = \\ &= \frac{l}{q^{(k+1)r}} \sum_{\bar{c}\in P^r} \chi(-\bar{c}\bar{z}) W_{\psi_{c_1}}(\bar{0})\cdots W_{\psi_{c_r}}(\bar{0}) = \frac{l}{q^{kr}} |f^{-1}(z_1)|\cdots|f^{-1}(z_r)|.\end{aligned}$$

Подставим найденное значение в равенство (25) и перейдём к исследованию абсолютных величин, в результате будем иметь

$$\left| N_l(\bar{z},\bar{s},v) - |f^{-1}(z_1)|\cdots|f^{-1}(z_r)| \frac{l}{q^{kr}} \right| \leqslant \frac{1}{q^r} \sum_{d=1}^{r} \sum_{\substack{\bar{c}\in P^r,\\ |\bar{c}|=d}} |D_l^*(\bar{c})|. \tag{26}$$

Из равенства (24), теоремы 1 и п. 4 утверждения 2 получим

$$\begin{aligned}|D_l^*(\bar{c})| &\leqslant \frac{C_l(F)}{q^{kd}} \sum_{\bar{a}_1,\ldots,\bar{a}_d \in P^k\setminus\{(\bar{0},\ldots,\bar{0})\}} |W_{\psi_{h_1}}(\bar{a}_1)|\cdots|W_{\psi_{h_d}}(\bar{a}_d)| \leqslant \\ &\leqslant \frac{C_l(F)}{q^{kd}} \left(\sum_{\bar{a}_1\in P^k} |W_{\psi_{h_1}}(\bar{a}_1)| \right) \cdots \left(\sum_{\bar{a}_d\in P^k} |W_{\psi_{h_d}}(\bar{a}_d)| \right) \leqslant C_l(F) q^{kd/2}.\end{aligned} \tag{27}$$

Тогда с использованием неравенства (26) можно записать

$$\begin{aligned}\left| N_l(\bar{z},\bar{s},v) - |f^{-1}(z_1)|\cdots|f^{-1}(z_r)| \frac{l}{q^{kr}} \right| &\leqslant \frac{1}{q^r} \sum_{d=1}^{r} \binom{r}{d} (q-1)^d C_l(F) q^{kd/2} = \\ &= \frac{C_l(F)}{q^r} \left(((q-1)q^{k/2}+1)^r - 1 \right).\end{aligned}$$

Теорема доказана. ■

Отметим, что при $r = 1$ оценка из теоремы 4 совпадает с оценкой из теоремы 2. Таким образом, теорема 4 обобщает теорему 2 на случай r-грамм.

Если f — сбалансированное отображение, то

$$|f^{-1}(z_1)|\cdots|f^{-1}(z_r)| \frac{l}{q^{kr}} = \frac{l}{q^r},$$

и теорема 4 устанавливает оценку отклонения частоты $N_l(\bar{z},\bar{s},v)$ от «естественного» среднего значения l/q^r.

Приведём условия, при которых применима оценка из теоремы 4. Как отмечалось ранее, на длину r набора $\bar{z}$ накладывается условие $r \leqslant m/k$, где m — степень многочлена $F(x)$, а k — число переменных функции f. Укажем условия, при которых оценка из теоремы 4 нетривиальна (правая часть неравенства меньше l). С использованием неравенств (26) и (27) получим, что справедлива менее точная оценка

$$\left| N_l(\bar{z}, \bar{s}, v) - |f^{-1}(z_1)| \cdots |f^{-1}(z_r)| \frac{l}{q^{kr}} \right| \leqslant \frac{(q^r - 1) C_l(F)}{q^r} q^{kr/2}. \tag{28}$$

Если $l = T(F)$, то данная оценка нетривиальна при условии $T(F) \geqslant q^{(m+kr)/2}$. Если $l < T(F)$, то оценка (28) нетривиальна при условиях

$$l > \frac{q^r - 1}{q^r} \left(\frac{4}{\pi^2} \ln T(F) + \frac{9}{5} \right) q^{(m+kr)/2} \text{ и } l > \frac{q^r - 1}{q^r} \left(3(q^m l - l^2) \right)^{1/3} q^{kr/2}$$

соответственно. Второе из этих неравенств заведомо выполнено, если $l \geqslant \sqrt{3} q^{m/2 + 3kr/4}$.

Согласно соотношению (27), $|D_l^*(\bar{c})| \leqslant \dfrac{C_l(F)}{q^{kd}} \left(\max\limits_{c \in P^*} \sum\limits_{\bar{a} \in P^k} |W_{\psi_c}(\bar{a})| \right)^d$, поэтому в условиях теоремы 4 справедлива оценка

$$\left| N_l(\bar{z}, \bar{s}, v) - |f^{-1}(z_1)| \cdots |f^{-1}(z_r)| \frac{l}{q^{kr}} \right| \leqslant \frac{C_l(F)}{q^r} \left(\left(\frac{q-1}{q^k} \left(\max_{c \in P^*} \sum_{\bar{a} \in P^k} |W_{\psi_c}(\bar{a})| \right) + 1 \right)^r - 1 \right).$$

Отсюда с использованием п. 3 утверждения 2 получим

$$\left| N_l(\bar{z}, \bar{s}, v) - |f^{-1}(z_1)| \cdots |f^{-1}(z_r)| \frac{l}{q^{kr}} \right| \leqslant \frac{C_l(F)}{q^r} \left(\left((q-1) \max_{c \in P^*} |M(\psi_c)|^{1/2} + 1 \right)^r - 1 \right).$$

Пусть $P = \mathrm{GF}(q)$, $f(x_1, \ldots, x_k) = c_1 x_1 + \ldots + c_k x_k$ для некоторых не всех равных нулю элементов $c_1, \ldots, c_k \in P$. Тогда $|M(\psi_c)| = 1$ для всех $c \in P$, последовательность v, удовлетворяющая равенству (2), является ЛРП с характеристическим многочленом $F(x)$, и для неё из последнего неравенства получим известные оценки (см. [9, теорема 2], [11, теорема 3], [17, теорема 1])

$$\left| N_l(\bar{z}, \bar{s}, v) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r} C_l(F).$$

При этом из доказательства теоремы 4 следует, что вместо сильной линейной независимости системы (18) достаточно потребовать сильную линейную независимость системы $x^{s_1} v, \ldots, x^{s_r} v$.

ЛИТЕРАТУРА

1. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. С.* Основы криптографии. М.: Гелиос АРВ, 2001. 480 с.
2. *Глухов М. М., Елизаров В. П., Нечаев А. А.* Алгебра: учебник. Т. 2. М.: Гелиос АРВ, 2003. 416 с.
3. *Kurakin V. L., Kuzmin A. S., Mikhalev A. V., and Nechaev A. A.* Linear recurring sequences over rings and modules // J. Math. Sci. 1995. V. 76. No. 6. P. 2793–2915.
4. *Лаксов Д.* Линейные рекуррентные последовательности над конечными полями // Математика: сб. переводов. 1967. Т. 11. № 6. С. 145–158.

5. *Лидл Р., Нидеррайтер Г.* Конечные поля. М.: Мир, 1988. Т. 1, 2. 822 с.
6. *Мак-Вильямс Ф. Д., Слоэн Н. Д. А.* Теория кодов, исправляющих ошибки. М.: Связь, 1979. 744 с.
7. *Dai Z. D., Feng X. N., Liu M. L., and Wan Z. X.* Some statistical properties of feedforward sequences (I) // Science in China (Ser. A). 1994. V. 37. No. 1. P. 34–41.
8. *Dai Z. D., Feng X. N., Liu M. L., and Wan Z. X.* Some statistical properties of feedforward sequences (II) // Science in China (Ser. A). 1994. V. 37. No. 2. P. 129–136.
9. *Niederreiter H.* Distribution properties of feedback shift register sequences // Probl. Control and Inform. Theory. 1986. V. 15. No. 1. P. 19–34.
10. *Cochran T.* On a trigonometric inequality of Vinogradov // J. Number Theory. 1987. V. 27. No. 1. P. 9–16.
11. *Сидельников В. М.* Оценки для числа появлений знаков на отрезках рекуррентной последовательности над конечным полем // Дискретная математика. 1991. Т. 3. № 2. С. 87–95.
12. *Коробов Н. М.* Распределение невычетов и первообразных корней в рекуррентных рядах // Докл. Акад. наук СССР. 1953. Т. 88. № 4. С. 603–606.
13. *Солодовников В. И.* Бент-функции из конечной абелевой группы в конечную абелеву группу // Дискретная математика. 2002. Т. 14. № 1. С. 99–113.
14. *Амбросимов А. С.* Свойства бент-функций q-значной логики над конечными полями // Дискретная математика. 1994. Т. 6. № 3. С. 50–60.
15. *Логачев О. А.*, *Сальников А. А.*, *Ященко В. В.* Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004. 470 с.
16. *Рязанов Б. В.* О распределении сектральной сложности булевых функций // Дискретная математика. 1994. Т. 6. № 2. С. 111–119.
17. *Нечаев В. И.* Распределение знаков в последовательности прямоугольных матриц над конечным полем // Труды математического института им. В. А. Стеклова. 1997. Т. 218. С. 335–342.

УДК 519.7

О НАДСТРУКТУРЕ КЛАССА КВАЗИОДНОРОДНЫХ k-ЗНАЧНЫХ ФУНКЦИЙ[1]

В. Б. Ларионов

ООО «Атес Медика Софт», г. Москва, Россия

E-mail: VitalyBLarionov@yandex.ru

Рассматривается фрагмент решётки замкнутых классов функций многозначной логики — надструктура класса, являющегося обобщением класса однородных функций. Доказано, что никаких классов, содержащих класс квазиоднородных функций, кроме классов квазисамодвойственных функций и их пересечений, не существует.

Ключевые слова: *многозначная логика, решётка замкнутых классов, самодвойственные функции.*

Введение

Известно [1], что решётка замкнутых относительно операции суперпозиции классов функций k-значной логики для любого $k \geqslant 3$ содержит континуальное число классов. Данный факт делает практически невозможным описание указанной решётки при $k \geqslant 3$, поэтому впоследствии изучались лишь её фрагменты. К указанному направлению и принадлежит данная работа. Автором развивается техника, разработанная в [2], для описания надструктуры классов, обладающих определёнными свойствами, с помощью которой изучается надструктура класса квазиоднородных функций, являющегося обобщением класса однородных функций. Данная техника использует предикатный подход и позволяет перейти от формул из предикатов к графам.

1. Основные понятия

Обозначим через E_k множество $\{0, 1, \ldots, k-1\}$.

Определение 1. Функция $f(x_1, \ldots, x_n)$ называется *функцией k-значной логики* ($k \geqslant 2$), если она определена на E_k^n и все её значения принадлежат E_k.

Будем использовать следующие стандартные обозначения [3]. Множество всех функций k-значной логики обозначим P_k. Для любого подмножества A из P_k через $[A]$ будем обозначать замыкание относительно операции суперпозиции (для функций везде далее будет идти речь именно об этом типе замыкания). Для краткости везде далее под термином «класс» будем подразумевать именно замкнутый класс.

Определение 2. Для данного класса A *надструктурой* будем называть множество классов, строго содержащих класс A.

Пусть на множестве E_k задана некоторая подстановка σ.

Определение 3. Функция k-значной логики $f(x_1, \ldots, x_n)$ называется *самодвойственной относительно подстановки σ*, если выполнено следующее тождество:

$$f(x_1, \ldots, x_n) = \sigma^{-1}(f(\sigma(x_1), \ldots, \sigma(x_n))),$$

где через σ^{-1} обозначается подстановка, обратная к σ.

[1]Работа поддержана грантом РФФИ № 13-01-00684-а.

Известно [4], что множество всех функций из P_k, самодвойственных относительно σ, является замкнутым классом. Обозначим этот класс через S_σ.

Определение 4. Замкнутый класс функций, равный пересечению всех классов самодвойственных функций, называется *классом однородных функций*. Будем обозначать указанный класс через S_k.

Определение 5. Пусть $p(x_1, \ldots, x_m)$ — некоторый предикат, определённый на E_k^m, $f(y_1, \ldots, y_n)$ — функция из множества P_k. Будем говорить, что функция $f(y_1, \ldots, y_n)$ *сохраняет предикат* $p(x_1, \ldots, x_m)$, если для любых n наборов $\tilde{a}_i = (a_{i1}, \ldots, a_{im})$, $i \in \{1, \ldots, n\}$, удовлетворяющих предикату p, набор $f(a_{11}, \ldots, a_{n1}), \ldots, f(a_{1m}, \ldots, a_{nm})$ также удовлетворяет предикату p. По определению будем считать, что тождественно ложный предикат сохраняет любая функция.

Обозначим через $\mathrm{Pol}(p)$ множество функций, сохраняющих предикат p. Для произвольного множества функций A через $\mathrm{Inv}(A)$ обозначим множество предикатов, каждый из которых сохраняет любая функция из A.

На множестве предикатов вводятся следующие операции: конъюнкция, отождествление переменных, добавление квантора существования по какой-либо переменной (проекция). Для произвольного множества предикатов P через $[P]$ будем обозначать замыкание относительно указанных операций. Подробное определение этих операций можно найти в [5, 3].

Лемма 1 [5]. Если $p_1 \in [p_2]$, то $\mathrm{Pol}(p_2) \subseteq \mathrm{Pol}(p_1)$.

Лемма 2 [6]. Пусть $p = p_1 \& \ldots \& p_m$, где предикаты $p_1, \ldots, p_m$ не имеют общих переменных. Тогда $\mathrm{Pol}(p) = \bigcap\limits_{i=1}^{m} \mathrm{Pol}(p_i)$.

2. Классы самодвойственных и квазисамодвойственных функций

Обобщим определение классов самодвойственных функций, заменив подстановку на множестве E_k на взаимно однозначное отображение некоторых подмножеств множества E_k.

Итак, пусть A и B — произвольные непустые подмножества E_k одинаковой мощности. Обозначим через F_{AB} множество всех различных взаимно однозначных отображений множества A во множество B, а через F_k — объединение множеств F_{AB} для всевозможных пар подмножеств A и B указанного вида. Для $f \in F_{AB}$ обозначим $D_f = A$, $T_f = A \bigcup B$.

Для произвольного отображения $f \in F_k$ обозначим через $R_f(x_1, x_2)$ предикат, истинный на всех парах $(a, f(a))$, где $a \in D(f)$, и только на них.

Определение 6. Замкнутые классы функций $S_f = \mathrm{Pol}(R_f)$, где $f \in F_k$, будем называть классами квазисамодвойственных функций, а сами функции, входящие в указанные классы, — квазисамодвойственными. Отметим, что, согласно данному определению, все классы самодвойственных функций [4] (в случае $D_f = E_k$, f — не тождественная подстановка), а также класс P_k (в случае $D_f = E_k$, f — тождественная подстановка на множестве E_k) являются классами квазисамодвойственных функций. Если f — тождественная подстановка на D_f, где $D_f \neq E_k$, то S_f — предполный центральный класс [6].

По аналогии с классом однородных функций, равным пересечению всех классов самодвойственных функций, определим класс квазиоднородных функций KS_k как пересечение всех классов квазисамодвойственных функций.

Класс KS_k можно задать и через предикаты. Пусть R_{KS} — множество всех предикатов R_f, где $f \in F_k$. Тогда $KS_k = \mathrm{Pol}(R_{KS})$.

В работе [7] установлено, что не все классы квазисамодвойственных функций содержат класс однородных функций, то есть класс KS_k строго содержится в классе однородных функций. Возникает вопрос, какие ещё классы, кроме классов квазисамодвойственных функций, содержат класс KS_k? В данной работе даётся ответ на поставленный вопрос.

Лемма 3. Пусть замкнутый класс A содержит класс KS_k. Тогда $\mathrm{Inv}(A) \subseteq [R_{KS}]$.

Доказательство. Из $KS_k \subseteq A$ с учетом антимонотонности оператора Inv [5] получаем $\mathrm{Inv}(A) \subseteq \mathrm{Inv}(KS_k) = \mathrm{Inv}\,\mathrm{Pol}(R_{KS})$.

Обозначим через $d(x_1, x_2)$ предикат, являющийся двухместной диагональю ($d(a, b) = = \texttt{true}$ тогда и только тогда, когда $a = b$). Отметим, что $d \in R_{KS}$, поскольку $d = R_f$, где $f \in F_k$ — тождественное отображение множества E_k в себя. С учетом сказанного из [5] следует $\mathrm{Inv}\,\mathrm{Pol}(R_{KS}) = [R_{KS}]$, откуда $\mathrm{Inv}(A) \subseteq [R_{KS}]$. ■

3. Формулы над R_{KS}

Предположим, что предикат p реализуется над R_{KS} формулой F. Везде далее будем считать, что в формуле F вынесены вперёд все кванторы существования. Сопоставим F ориентированный граф $G(F)$ по следующему правилу: между множеством вершин $G(F)$ и множеством переменных F (учитываем и свободные, и связанные) существует взаимно однозначное соответствие. Вершину, соответствующую переменной x, пометим символом «x», если переменная x свободная, и «$\exists x$», если связанная. Данную вершину будем обозначать v_x. В графе $G(F)$ есть ориентированное ребро (v_x, v_y) с пометкой «f_i» тогда и только тогда, когда в формуле F содержится запись $R_{f_i}(x, y)$.

Отметим, что по графу формулы $G(F)$ формула F с вынесенными вперёд кванторами существования восстанавливается однозначно.

Определение 7. *Путём* из вершины v_1 в вершину v_2 в ориентированном графе G будем называть любую последовательность рёбер вида

$$\{(v_1, w_1), (w_1, w_2), (w_2, w_3), \ldots, (w_m, v_2)\},$$

вершины и рёбра в которой могут повторяться. Ориентация рёбер последовательности может быть любой. *Замкнутым путём* называется путь, в котором первая и последняя вершины совпадают.

Для произвольного пути S в графе определим величину $L(S) \in F_k \cup \{f_0\}$, где f_0 — пустое отображение, т. е. L ставит в соответствие пути некоторое взаимно однозначное отображение подмножеств (возможно, пустых) множества E_k. При этом для пути, состоящего из одного ребра (v_1, v_2) с пометкой f_i, положим $L(\{(v_1, v_2)\}) = f_i$, $L(\{(v_2, v_1)\}) = f_i^{-1}$. Для пути $S = \{(w_1, w_2), (w_2, w_3), \ldots, (w_{m-1}, w_m)\}$ положим

$$L(S) = L(\{(w_{m-1}, w_m)\})L(\{(w_{m-1}, w_{m-2})\}) \ldots L(\{(w_1, w_2)\}).$$

То есть, $L(S)$ — композиция отображений в обратном порядке.

Непосредственно из определений графа формулы и величины L следует

Лемма 4. Пусть предикат p реализуется над R_{KS} формулой F с графом $G(F)$. Тогда для любого набора $\tilde{a}$, такого, что $p(\tilde{a}) = \texttt{true}$, и для двух произвольных вершин графа v_{y_1}, v_{y_2}, соединённых путём S и таких, что переменные y_1, y_2 принимают на наборе $\tilde{a}$ значения b и c соответственно, справедливо $c = (L(S))(b)$.

Для произвольной вершины v_y графа формулы $G(F)$ через T_y обозначим множество элементов $a \in E_k$, таких, что

1) для любого пути S из v_y в некоторую вершину v_z существует $L(S)(a)$;
2) для любого замкнутого пути S, проходящего через вершину v_y, справедливо $a = (L(S))(a)$.

Докажем далее основное свойство формул, которое позволит нам редуцировать множество $[R_{KS}]$, тем самым сделав прозрачной надструктуру класса KS_k.

Лемма 5. Пусть предикат p реализуется над R_{KS} формулой F с множеством свободных переменных $\{x_1, \ldots, x_n\}$ и со связным графом $G(F)$. Пусть между любыми двумя вершинами v_{x_i}, v_{x_j} в G_F существует путь S_{ij}, такой, что $L(S_{ij}) = f_{ij}$, $i, j \in \in \{1, \ldots, n\}$. Тогда $p(\tilde{a}) = \texttt{true}$ тогда и только тогда, когда для некоторого $i \in \{1, \ldots n\}$ справедливо:

1) $a_i \in T_{x_i}$;
2) $a_j = f_{ij}(a_i)$ для любых $j \in \{1, \ldots, n\}$.

Доказательство. Пусть набор $\tilde{a}$ таков, что $p(\tilde{a}) = \texttt{true}$. Предположим, что для некоторого i справедливо $a_i \notin T_{x_i}$. Если нарушено первое требование из определения величины T_y, то в графе G_F найдётся вершина v_z, такая, что для переменной z на наборе $\tilde{a}$ не найдётся подходящего значения, откуда $p(\tilde{a}) = \texttt{false}$. Если же нарушено второе требование, то существует замкнутый путь S, проходящий через вершину v_{x_i}, такой, что $L(S) = f$ и $f(a_i) \neq a_i$. Получаем противоречие с леммой 4. Справедливость соотношения $a_j = f_{ij}(a_i)$ вытекает непосредственно из леммы 4.

Пусть теперь набор $\tilde{a}$ таков, что перечисленные в лемме два условия выполнены. Покажем, что $p(\tilde{a}) = \texttt{true}$.

Пусть i — число из $\{1, \ldots, n\}$, для которого выполняются условия леммы. Рассмотрим произвольную вершину v_y графа $G(F)$. Поскольку указанный граф связный, то существует путь S_1 от вершины v_{x_i} до v_y. Пусть $L(S_1) = f_1$. Присвоим переменной y значение $b = f_1(a_i)$ (согласно первому пункту определения T_y, это значение существует). Пусть S_2 — некоторый путь из v_{x_i} в v_y, отличный от S_1, и $L(S_2) = f_2$. Покажем, что $b = f_2(a_i)$, т. е. найденное значение b не зависит от выбора пути. Соединением путей S_1 и S_2 можно получить замкнутый путь S_3, проходящий через вершину v_{x_i}. При этом $L(S_3) = f_2 f_1^{-1}$. Поскольку $a_i \in T_{x_i}$, то $f_2 f_1^{-1}(a_i) = a_i$, откуда $f_1(a) = f_2(a)$. В силу доказанного и второго условия леммы все переменные x_j, $j \in \{1, \ldots, n\}$, при данном присвоении примут значения a_j.

Покажем, что на присвоенных значениях каждый сомножитель $R_f(y_l, y_j)$ формулы F истинен, т. е. $p(\tilde{a}) = \texttt{true}$. Предположим, что при проведённом выше присвоении переменные y_l, y_j приняли соответственно значения b_l и b_j. Это означает, что из вершины v_{x_i} существуют пути S', S'' до вершин v_{y_l}, v_{y_j} соответственно, при этом $L(S') = f'$, $L(S'') = f''$ и $b_l = f'(a_i)$, $b_j = f''(a_i)$. Сомножитель $R_f(y_l, y_j)$ формулы F соответствует ребру (v_{y_l}, v_{y_j}) графа $G(F)$, $L(\{(v_{y_l}, v_{y_j})\}) = f$. Соединением указанного ребра с путями S' и S'' получим замкнутый путь S, проходящий через вершину v_{x_i}, такой, что $L(S) = f''^{-1} f f'$. Из $a_i \in T_{x_i}$ следует $f''^{-1} f f'(a_i) = a_i$, откуда $f f'(a_i) = f''(a_i)$, или $f(b_l) = b_j$. Получаем, что $R_f(b_l, b_j) = \texttt{true}$. ■

4. Надструктура класса KS_k

Докажем основной результат данной работы.

Теорема 1. Надструктура класса квазиоднородных функций KS_k состоит только из классов квазисамодвойственных функций и их пересечений.

Доказательство. То, что указанные в формулировке теоремы классы входят в надструктуру класса KS_k, следует непосредственно из определения класса квазиоднородных функций. Покажем далее, что никакие другие классы не содержатся в этой надструктуре.

Рассмотрим некоторый класс A, содержащий KS_k. Возьмём произвольный предикат p из множества $\mathrm{Inv}(A)$. По лемме 3 выполняется $p \in [R_{KS}]$. Обозначим через F формулу, реализующую p над множеством R_{KS}, $G(F)$ — её граф. Рассмотрим сначала случай, когда $G(F)$ связен.

Пусть $x_1, \ldots, x_n$ — все свободные переменные формулы F. Поскольку граф $G(F)$ связен, то существуют пути $S_2, \ldots, S_n$ от вершины v_{x_1} до вершин $v_{x_2}, \ldots, v_{x_n}$ соответственно. Обозначим $f_j = L(S_j)$, $H = T_{x_1}$.

Если местность n предиката p равна единице, то по лемме 5 получаем $p(a) = \texttt{true}$ тогда и только тогда, когда $a \in H$. В этом случае $\mathrm{Pol}(p)$ либо совпадает с P_k (если $H = E_k$), либо является предполным центральным классом. При этом $\mathrm{Pol}(p) = \mathrm{Pol}(R_f)$, где f — тождественное отображение множества H в себя. Таким образом, $\mathrm{Pol}(p)$ является классом квазисамодвойственных функций.

Пусть теперь $n = 2$. По лемме 5 $p(a, b) = \texttt{true}$ тогда и только тогда, когда $a \in A$ и $b = f_2(a)$. Получаем, что $p = R_f$, где f — взаимно однозначное отображение, определённое на множестве H и совпадающее на нём с отображением f_2, т. е. $\mathrm{Pol}(p)$ — класс квазисамодвойственных функций.

Остаётся случай $n > 2$. Обозначим предикаты

$$p_i(x_1, x_i) = \exists y_1 \ldots, y_{n-2} p(x_1, y_1, \ldots, y_{i-2}, x_i, y_{i-1}, \ldots, y_{n-2}),$$

где $i \in \{2, \ldots, n\}$. Получаем, что $p_i \in [p]$, откуда $p_i \in [R_{KS}]$. Предикаты p_i попадают в уже рассмотренный случай (граф формулы, реализующей p_i, можно получить из графа G_F перепомечиванием вершин, поэтому он связный), т. е. классы $\mathrm{Pol}(p_i)$ являются классами квазисамодвойственных функций. Рассмотрим предикат $p'(x_1, \ldots, x_n) = p_2(x_1, x_2) \& p_3(x_1, x_3) \& \ldots \& p_n(x_1, x_n)$. Пусть набор $\tilde{a}$ таков, что $p(\tilde{a}) = \texttt{true}$. Получаем, что $p_i(a_1, a_i) = \texttt{true}$ для всех i, откуда $p'(\tilde{a}) = \texttt{true}$. Обратно, пусть $p'(\tilde{a}) = \texttt{true}$, следовательно, все $p_i(a_1, a_i) = \texttt{true}$. Отсюда имеем, что $a_1 \in H$, $a_i = f_i(a_1)$ для всех $i \in \{2, \ldots, n\}$. По лемме 5 получаем, что $p(\tilde{a}) = \texttt{true}$. Окончательно имеем $p' = p$.

Итак, получили представление

$$p(x_1, \ldots, x_n) = p_2(x_1, x_2) \& p_3(x_1, x_3) \& \ldots \& p_n(x_1, x_n).$$

Обозначим через t предикат, равный конъюнкции предикатов $p_2, \ldots, p_n$ без отождествления переменных. Из последнего соотношения следует, что $p \in [t]$ (p получается из t отождествлением переменных). С другой стороны, из $p_i \in [p]$ следует, что $t \in [p]$. По лемме 1 получаем, что $\mathrm{Pol}(p) = \mathrm{Pol}(t)$. По лемме 2 класс $\mathrm{Pol}(t)$, а значит и $\mathrm{Pol}(p)$, является пересечением классов $\mathrm{Pol}(p_i)$, т. е. классов квазисамодвойственных функций.

Пусть теперь G_F — несвязный граф. Каждая компонента связности G_F очевидным образом задаёт свой предикат, для которого справедливы приведённые выше рассуждения. Предикат p является конъюнкцией (без отождествления переменных) указанных предикатов. Опять получаем [6], что $\mathrm{Pol}(p)$ — некоторое пересечение классов квазисамодвойственных функций. ■

Заключение

Итак, никаких классов, содержащих класс квазиоднородных функций, кроме классов квазисамодвойственных функций и их пересечений, не существует. Можно сказать, что класс KS_k находится в решётке замкнутых классов достаточно неглубоко.

ЛИТЕРАТУРА

1. *Янов Ю. И., Мучник А. А.* О существовании k-значных замкнутых классов, не имеющих конечного базиса // ДАН СССР. 1959. Т. 127. № 1. С. 44–46.
2. *Ларионов В. Б.* Замкнутые классы k-значной логики, содержащие классы монотонных или самодвойственных функций: дис. ... канд. физ.-мат. наук. М., 2009. 157 с.
3. *Марченков С. С.* Замкнутые классы булевых функций. М.: Физматлит, 2000.
4. *Яблонский С. В.* Функциональные построения в k-значной логике // Тр. МИАН им. В. А. Стеклова. 1958. Т. 51. С. 5–142.
5. *Боднарчук В. Г., Калужнин В. А., Котов В. Н., Ромов Б. А.* Теория Галуа для алгебр Поста // Кибернетика. 1969. № 3. С. 1–10; № 5. С. 1–9.
6. *Яблонский С. В., Гаврилов Г. П., Набебин А. А.* Предполные классы в многозначных логиках. М.: Изд. дом МЭИ, 1997.
7. *Ларионов В. Б., Федорова В. С.* Замкнутые классы, содержащие класс однородных функций // Вестник МГУ. Сер. 15. Вычислительная математика и кибернетика. 2012. № 1. С. 34–38.

УДК 512.572

ОБ ЭКСПОНЕНТАХ НЕКОТОРЫХ МНОГООБРАЗИЙ ЛИНЕЙНЫХ АЛГЕБР

С. М. Рацеев

Ульяновский государственный университет, г. Ульяновск, Россия

E-mail: RatseevSM@mail.ru

Пусть UT_s — алгебра верхнетреугольных матриц порядка s. Приводятся эквивалентные условия для оценок роста подмногообразий многообразия $var(UT_s)$, многообразий алгебр Лейбница с нильпотентным коммутантом и многообразий алгебр Лейбница — Пуассона, идеалы тождеств которых содержат тождества вида $\{\{x_1, y_1\}, \ldots, \{x_n, y_n\}\} = 0$, $\{x_1, y_1\} \cdot \ldots \cdot \{x_n, y_n\} = 0$.

Ключевые слова: *многообразие линейных алгебр, рост многообразия, экспонента многообразия.*

Алгебра Лейбница над полем K — векторное пространство с K-билинейной операцией умножения $\{,\}$, относительно которого выполнено тождество Лейбница

$$\{\{x, y\}, z\} = \{\{x, z\}, y\} + \{x, \{y, z\}\},$$

превращающее правое умножение в дифференцирование этой алгебры. При этом заметим, что если в алгебре Лейбница выполняется тождество $\{x, x\} = 0$, то она является алгеброй Ли.

Векторное пространство A над полем K с двумя K-билинейными операциями умножения $\cdot$ и $\{,\}$ называется алгеброй Лейбница — Пуассона, если относительно операции $\cdot$ пространство A является коммутативной ассоциативной алгеброй с единицей, относительно операции $\{,\}$ — алгеброй Лейбница и для любых $a, b, c \in A$ данные операции связаны правилами

$$\{a \cdot b, c\} = a \cdot \{b, c\} + \{a, c\} \cdot b, \quad \{c, a \cdot b\} = a \cdot \{c, b\} + \{c, a\} \cdot b.$$

Алгебры Лейбница — Пуассона возникают в различных разделах алгебры, дифференциальной геометрии, топологии, современной теоретической физики и являются обобщениями алгебр Пуассона.

Пусть V — некоторое многообразие линейных алгебр над полем K (необходимые сведения о многообразиях PI-алгебр можно найти, например, в [1, 2]), $K(X, V)$ — свободная алгебра многообразия V, где $X = \{x_1, x_2, \ldots\}$ — счётное множество свободных образующих; $P_n(V)$ — подпространство в $K(X, V)$, состоящее из всех полилинейных элементов степени n от переменных $x_1, \ldots, x_n$. Обозначим

$$c_n(V) = \dim P_n(V), \quad \exp(V) = \lim_{n\to\infty} \sqrt[n]{c_n(V)}.$$

Хорошо известно, что в ассоциативном случае при $\operatorname{char} K = 0$ экспонента произвольного нетривиального многообразия существует и является целым числом (М. В. Зайцев и А. Джамбруно [3]). В случае многообразий алгебр Ли при $\operatorname{char} K = 0$ построен пример разрешимого многообразия [4], экспонента которого находится в интервале (3,4).

Напомним, что в случае основного поля нулевой характеристики S_n-модуль $P_n(V)$ является вполне приводимым и разложение его характера в целочисленную комбинацию неприводимых характеров имеет следующий вид:

$$\chi_n(V) = \chi(P_n(V)) = \sum_{\lambda \vdash n} m_\lambda(V)\chi_\lambda. \tag{1}$$

Обозначим через V_s^A многообразие ассоциативных алгебр, определённое тождеством

$$[x_1, x_2][x_3, x_4] \dots [x_{2s-1}, x_{2s}] = 0,$$

где $[,]$ — операция коммутирования. В работах [5, 6], в частности, показано, что для любого многообразия ассоциативных алгебр V над произвольным полем $\exp(V \cap V_s^A)$ существует и является целым числом. Пусть $UT_s = UT_s(K)$ — алгебра верхнетреугольных матриц порядка s. Хорошо известно [7], что при $\operatorname{char} K = 0$ алгебра UT_s порождает многообразие V_s^A.

Пусть V_s^L — многообразие алгебр Лейбница, определённое тождеством

$$\{x_1, x_2\}\{x_3, x_4\} \dots \{x_{2s+1}, x_{2s+2}\} = 0.$$

В работе [8] показано, что для любого многообразия алгебр Лейбница V над произвольным полем $\exp(V \cap V_s^L)$ существует и является целым числом.

Обозначим через V_s^{LP} многообразие алгебр Лейбница — Пуассона, определённое всеми полилинейными тождествами степени $2s$ вида

$$\{\{x_{11}, y_{11}\}, \{x_{12}, y_{12}\}, \dots, \{x_{1\lambda_1}, y_{1\lambda_1}\}\} \cdot \{\{x_{21}, y_{21}\}, \{x_{22}, y_{22}\}, \dots, \{x_{2\lambda_2}, y_{2\lambda_2}\}\} \cdot \dots$$
$$\dots \cdot \{\{x_{k1}, y_{k1}\}, \{x_{k2}, y_{k2}\}, \dots, \{x_{k\lambda_k}, y_{k\lambda_k}\}\} = 0, \quad \lambda \vdash s.$$

Например, многообразие V_4^{LP} определяется полилинейными тождествами

$$\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}\} = 0,$$
$$\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\} \cdot \{x_4, y_4\} = 0,$$
$$\{\{x_1, y_1\}, \{x_2, y_2\}\} \cdot \{\{x_3, y_3\}, \{x_4, y_4\}\} = 0,$$
$$\{\{x_1, y_1\}, \{x_2, y_2\}\} \cdot \{x_3, y_3\} \cdot \{x_4, y_4\} = 0,$$
$$\{x_1, y_1\} \cdot \{x_2, y_2\} \cdot \{x_3, y_3\} \cdot \{x_4, y_4\} = 0.$$

В работе [9] показано, что для любого многообразия алгебр Лейбница — Пуассона V над произвольным полем $\exp(V \cap V_s^{LP})$ существует и является целым числом. Заметим, что если в многообразии алгебр Лейбница — Пуассона выполнены полилинейные тождества вида

$$\{\{x_1, y_1\}, \dots, \{x_n, y_n\}\} = 0, \quad \{x_1, y_1\} \cdot \dots \cdot \{x_n, y_n\} = 0,$$

то для некоторого s данное многообразие является подмногообразием в V_s^{LP}.

Теорема 1. Пусть характеристика основного поля равна нулю, V^A — многообразие ассоциативных алгебр, V^L — многообразие алгебр Лейбница, V^{LP} — многообразие алгебр Лейбница — Пуассона и d — некоторое неотрицательное целое число. Тогда для любого значения $a = A, L, LP$ следующие условия эквивалентны:

1) $\exp(V^a \cap V_{d+1}^a) \leqslant d$;
2) для любого целого $s > d$ выполнено неравенство $\exp(V^a \cap V_s^a) \leqslant d$;

3) существует такая константа C, что в сумме (1) $m_\lambda(V^a \cap V^a_{d+1}) = 0$ в случае, если выполнено условие $n - (\lambda_1 + \lambda_2 + \ldots + \lambda_d) > C$;
4) для любого целого $s > d$ существует такая константа $C = C(s)$, что в сумме (1) $m_\lambda(V^a \cap V^a_s) = 0$ в случае, если выполнено условие $n - (\lambda_1 + \lambda_2 + \ldots + \lambda_d) > C$.

Доказательство. При $a = A$ утверждение теоремы следует из работы [6], при $a = L$ — из работ [8, 10], при $a = LP$ — из работ [9, 11]. ■

Теорема 2. Пусть для некоторого многообразия линейных алгебр V^a, $a \in \{A, L, LP\}$, над полем нулевой характеристики и некоторого целого неотрицательного d выполнено равенство $\exp(V^a \cap V^a_{d+1}) = d$. Тогда для любого целого $s > d$ выполнено равенство $\exp(V^a \cap V^a_s) = d$.

Доказательство. Так как для любого s выполнено неравенство $\exp(V^a \cap V^a_s) \leqslant \exp(V^a \cap V^a_{s+1})$, то для любого целого $s > d$, с учётом теоремы 1, выполнено двойное неравенство $d \leqslant \exp(V^a \cap V^a_s) \leqslant d$. ■

ЛИТЕРАТУРА

1. *Бахтурин Ю. А.* Тождества в алгебрах Ли. М.: Наука, 1985. 448 с.
2. *Drensky V.* Free algebras and PI-algebras. Graduate course in algebra. Singapore: Springer Verlag, 2000. 272 с.
3. *Giambruno A. and Zaicev M. V.* On codimention growth of finitely generated associative algebras // Adv. Math. 1998. V. 140. P. 145–155.
4. *Zaitcev M. V. and Mishchenko S. P.* Example of variety of Lie algebras with fractional exponent // J. Math. Sci. 1999. V. 93. No. 6. P. 977–982.
5. *Petrogradsky V. M.* Exponents of subvarieties of upper triangular matrices over arbitrary fields are integral // Serdika Math. 2000. V. 26. No. 2. P. 1001–1010.
6. *Рацеев С. М.* Тождества в многообразиях, порожденных алгебрами верхнетреугольных матриц // Сиб. матем. журн. 2011. Т. 54. № 2. С. 416–429.
7. *Мальцев Ю. Н.* Базис тождеств алгебры верхнетреугольных матриц // Алгебра и логика. 1971. Т. 10. С. 393–400.
8. *Рацеев С. М.* Рост некоторых многообразий алгебр Лейбница // Вестник Самарского государственного университета. Естественнонаучная серия. 2006. Т. 46. № 6. С. 70–77.
9. *Ratseev S. M.* Growth of some varieties of Leibniz — Poisson algebras // Serdica Math. J. 2011. V. 37. No. 4. P. 331–340.
10. *Рацеев С. М.* Оценки роста многообразий алгебр Лейбница с нильпотентным коммутантом // Вестник Самарского государственного университета. Естественнонаучная серия. 2010. Т. 78. № 4. С. 65–72.
11. *Рацеев С. М., Черевatенко О. И.* Экспоненты некоторых многообразий алгебр Лейбница — Пуассона // Вестник Самарского государственного университета. Естественнонаучная серия. 2013. Т. 104. № 3. С. 42–52.

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 512.5; 00326.09

КРИПТОГРАФИЧЕСКИЙ АНАЛИЗ НЕКОТОРЫХ СХЕМ ШИФРОВАНИЯ, ИСПОЛЬЗУЮЩИХ АВТОМОРФИЗМЫ[1]

В. А. Романьков

Омский государственный университет им. Ф. М. Достоевского,
Омский государственный технический университет, г. Омск, Россия

E-mail: romankov48@mail.ru

Приводится криптографический анализ схем шифрования и распределения ключа, базирующихся на групповых (луповых) алгебрах и градуированных алгебрах с мультипликативным базисом, предложенных в работах С. К. Росошека, А. В. Михалева и др., А. Махалонобиса и др. Объединяет эти схемы (кроме одной из схем А. В. Михалева и др.) использование в них автоморфизмов. Приводится также криптографический анализ протокола распределения ключа Мегрелишвили и Джинджихадзе. Описывается оригинальный метод нахождения шифрованного сообщения или общего ключа, основанный на обычном аппарате линейной алгебры, при условии, что соответствующая платформа может быть выбрана как конечномерная алгебра, например как матричная алгебра над полем. Метод не предполагает нахождения секретных автоморфизмов, фигурирующих в указанных работах. Теоретические основы метода и ряд атак на его основе схем шифрования и распределения ключа, базирующихся на различных обобщениях задачи дискретного логарифма и идей Диффи — Хеллмана — Меркля на некоммутативные группы, изложены в других работах автора. Здесь метод находит новые применения.

Ключевые слова: *схема шифрования, групповая алгебра, луповая алгебра, матричная алгебра, градуированная алгебра, дискретный логарифм, обобщения дискретного логарифма, схема Диффи — Хеллмана, протокол ЭльГамаля, автоморфизм.*

Введение

Зарождение современной криптографии с открытым ключом обычно связывают с публикацией короткой заметки У. Диффи и М. Хеллмана [1]. В ней авторы не только впервые высказали замечательную идею открытой передачи секретных данных по незащищённым каналам связи без предварительного обмена корреспондентами какими-либо секретами, но также представили соответствующий алгоритм, известный как *протокол Диффи — Хеллмана* разделения ключа. Протокол впоследствии сыграл не только теоретическую роль, но был реализован в различных практических схемах криптографии. Его популярность в настоящее время нисколько не убавилась. Справедливости ради следует сказать, что, по словам самого М. Хеллмана [2], идея подобного

[1]Исследование выполнено при поддержке Министерства образования и науки РФ, проекты № 14.B37.21.0359 и 0859.

распределения ключей принадлежала Мерклю, поэтому сам протокол следует именовать *протокол Диффи — Хеллмана — Меркля*.

Протокол Диффи — Хеллмана — Меркля (DHM) работает следующим образом:

— Двое корреспондентов, скажем Алиса (А) и Боб (Б), выбирают конечную группу G и некоторый элемент g этой группы. При выборе А и Б пользуются незащищённым каналом связи, поэтому величины G и g считаются общеизвестными.
— Далее А выбирает случайным образом натуральное число $k \in \mathbb{N}$, вычисляет элемент g^k и передаёт его по открытому каналу корреспонденту Б. Само число k считается секретным.
— Б поступает аналогично: выбирает $l \in \mathbb{N}$, вычисляет и передает А элемент g^l. Число l считается секретным.
— Получив элемент g^l, А вычисляет элемент $(g^l)^k = g^{kl}$.
— Б делает то же самое, получая g^k и вычисляя $(g^k)^l = g^{kl}$.
— Элемент g^{kl} считается общим секретным ключом.

Реализация DHM должна быть такой, чтобы вычисление по данным G, g^k, g^l общего ключа g^{kl} было трудной вычислительной задачей. Эту задачу называют *проблемой Диффи — Хеллмана* (PDH). Она тесно связана с проблемой дискретного логарифма (PDL): по фиксированному элементу g известной конечной группы G и его степени $f = g^t$, $t \in \mathbb{N}$, определить число t, которое называется *дискретным логарифмом* элемента f относительно базы g и обозначается $\log_g f$. При ограничении $0 \leqslant t \leqslant \mathrm{ord}(g)$, где $\mathrm{ord}(g)$ обозначает порядок элемента g, дискретный логарифм $t = \log_g f$ определён однозначно. Обычно в качестве элемента g берется порождающий элемент конечной циклической группы $G = \mathrm{gp}(g)$. В этом случае $\log_g f$ существует для любого элемента f группы G. Если в протоколе DHM вычислить $k = \log_g g^k$ или $l = \log_g g^l$, то легко вычисляется и g^{kl}.

В оригинальной работе [1] и многих последующих работах в качестве платформ G для протокола DHM использовались мультипликативные группы F_p^* простых конечных полей F_p, p — простое, реализованных как кольца вычетов $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Эти группы очень удобны для построения на них DHM. Во-первых, они циклические, и поэтому при выборе в качестве g порождающего элемента группы F_p^* дискретный логарифм $\log_g f$ определён для любого элемента $f \in F_p^*$. Во-вторых, их элементы можно записывать стандартными именами вычетов $1, 2, \ldots, p-1$, по которым трудно вычислять их дискретные логарифмы относительно g.

В качестве платформ для DHM и других протоколов, основанных на трудности разрешимости PDL, стали использоваться также мультипликативные группы произвольных конечных полей F_q^*, $q = p^r$, p — простое, r — натуральное. Эти группы циклические, их элементы однозначно записываются в виде многочленов из кольца $F_p[x]$ степени не больше чем $r-1$. Вычисления ведутся по модулю неприводимого многочлена $h(x)$ степени r, по которому построено поле $F_q \simeq F_p[x]/(h(x))$.

В дальнейшем в качестве платформ протоколов типа DHM стали предлагаться кроме циклических и другие конечные группы, среди которых выделились группы эллиптических кривых над конечными полями. Обозначились и бесконечные группы, прежде всего — матричные (линейные) группы над полями, кольцами, алгебрами, затем стали широко использоваться полициклические группы, группы кос Артина и т. д. Кроме групп стали предлагаться полугруппы, лупы и т. п.

Оказалось [3, 4], что обычная проблема дискретного логарифма в группах матриц над полями сводится к кратной проблеме дискретного логарифма в поле, содержащем

все характеристические числа матрицы g, являющейся базой дискретного логарифма. Действительно, характеристические числа матрицы g^t являются t-степенями характеристических чисел матрицы g. Если матрица g приводится к диагональному виду, где на главной диагонали стоят эти характеристические числа, то такое сведение очевидно. В общем случае необходимо использовать более детальные рассуждения относительно жордановой формы матрицы g и её степеней. Есть и другие возможности сведения, о них см., например, в [5].

С самого начала использования некоммутативных групп появились аналоги дискретного логарифма. Наиболее популярным стало использование вместо возведения в степень сопряжения, в дальнейшем стали применяться правые и левые умножения и т. п. На этом строится целый ряд известных протоколов [6, 7]. Базовые протоколы, основанные на трудности решения так называемых проблем поиска, в большинстве своем имитировали классические криптографические схемы Диффи — Хеллмана — Меркля, ЭльГамаля, Масси — Омуры, Фиата — Шамира и т. д. [8 – 11].

В [5] автор предложил универсальный подход к криптоанализу схем, криптостойкость которых базируется на сложности решения проблем поиска для различных алгоритмических проблем. Оказалось, что в ряде случаев, в том числе для ряда известных протоколов, среди которых протоколы разделения ключа Ко, Ли и др. (сопряжение), протокол Стикелса (двустороннее домножение), итоговый секретный результат протокола (общий ключ или сообщение) можно получить, не решая соответствующих проблем поиска. При этом подходе применяются обычные методы линейной алгебры. Правда, такие атаки возможны, если соответствующий протокол может быть записан на платформе, представляющей из себя кольцо матриц над конечномерной алгеброй над конструктивным полем. В конечном случае это не является ограничением, поскольку можно всё перевести на платформу матриц над конечным полем. Но это часто можно сделать и в бесконечном случае, например в случае групп кос Артина, которые, как известно [12], линейны. Группы кос Артина — один из наиболее популярных объектов в криптографии [13 – 15]. Также линейны (см., например, [16, 17]) конечнопорождённые нильпотентные или, более общо, полициклические группы, которые всё чаще предлагаются в качестве платформ криптографических протоколов. Почти всегда предлагаемые для платформ алгебраические системы так или иначе представляются матрицами, что позволяет проводить атаку этим методом.

Данная работа посвящена криптографическому анализу ряда других протоколов, отличительной особенностью большинства из которых служит применение в них автоморфизмов в качестве как преобразований, так и ключей. Анализ также использует обычные методы линейной алгебры. Соответствующая атака проводится без вычисления параметров криптографической схемы, результат получается совсем другим способом.

1. Основная идея

Для представления основной общей идеи, позволяющей проводить эффективные атаки на схемы разделения ключа и шифрования в случае, когда платформой служит конечномерная алгебра над конечным полем, рассмотрим следующий достаточно простой протокол разделения ключа.

Протокол распределения ключей Мегрелишвили и Джинджихадзе [18] (см. также [19, 20])

Описание

Установка

Корреспонденты А и Б договариваются о выборе векторного пространства $V = F_2^n$ размерности n над полем F_2. Далее фиксируется квадратная матрица A размера $n \times n$ и вектор $v \in V$. Эти данные открыты.

Генерация ключей

Корреспондент А выбирает случайным образом натуральное число k, вычисляет и пересылает корреспонденту Б вектор $u = vA^k$. В свою очередь, корреспондент Б выбирает число l, вычисляет и пересылает А вектор $w = vA^l$.

Затем каждый из корреспондентов вычисляет общий ключ

$$K = uA^l = wA^k = vA^{k+l}.$$

Криптографический анализ системы Мегрелишвили и Джинджихадзе

Выпишем векторы $v = vA^0$, vA, …, vA^m до максимально возможной степени m с условием линейной независимости этого набора. Ясно, что $m \leqslant n$, поэтому процесс эффективен. Данный набор является базисом линейного пространства $\mathrm{lin}_{F_2}(vA^k, k \in \mathbb{N})$, порождённого всеми векторами вида $vA^k, k \in \mathbb{N}$. Для этого достаточно доказать, что любой вектор $vA^k, k \geqslant m+1$, линейно выражается через данный набор. Поскольку набор $v, vA, \ldots, vA^m, vA^{m+1}$ является первым линейно зависимым набором, вектор vA^{m+1} допускает разложение вида

$$vA^{m+1} = \sum_{i=0}^{m} \alpha_i vA^i,\ \alpha_i \in F_2.$$

Пусть уже доказано, что вектор $vA^k, k \geqslant m+1$, представим в виде

$$vA^k = \sum_{i=0}^{m} \beta_i vA^i,\ \beta_i \in F_2. \tag{1}$$

Умножим обе части (1) справа на матрицу A и проведём преобразование с использованием равенства (1):

$$vA^{k+1} = \sum_{i=0}^{m} \beta_i vA^{i+1} = \sum_{i=0}^{m-1} \beta_i vA^{i+1} + \beta_m \sum_{i=0}^{m} \beta_i vA^i = \beta_m\beta_0 v + \sum_{i=1}^{m} (\beta_{i-1} + \beta_m\beta_i) vA^i.$$

Утверждение о базисе $v, vA, \ldots, vA^m$ пространства $\mathrm{lin}_{F_2}(vA^k, k \in \mathbb{N})$ следует по индукции.

Теперь можно получить разложение

$$u = vA^k = \alpha_0 v + \alpha_1 vA + \ldots + \alpha_m vA^m,\ \alpha_i \in F_2. \tag{2}$$

Заметим, что для получения разложения (2) не нужно знать k, а только u.

После этого подставим в правую часть полученного выражения (2), где все компоненты известны, вектор w вместо v и получим

$$\alpha_0 w + \alpha_1 wA + \ldots + \alpha_m wA^m = (\alpha_0 v + \alpha_1 vA + \ldots + \alpha_m vA^m)A^l = vA^{k+l} = K.$$

Авторы данного протокола, анализируя его криптостойкость, рассматривали возможность нахождения числа k по уравнению вида $vA^k = u$ или числа l по уравнению вида $vA^l = w$. Значительное внимание они уделили способам выбора матрицы A достаточно большого порядка, при котором подобные вычисления становятся трудными. Конечно, существуют способы выбора матрицы A порядка $2^n - 1$. Однако при описанном выше подходе такой выбор не играет существенной роли. Данный пример достаточно хорошо иллюстрирует возможности подхода, основанного на вычислениях в линейных пространствах. В работе [5] дано описание целого ряда известных криптографических протоколов, которые также могут быть атакованы подобным образом. Конечно, конкретные реализации могут выглядеть более сложно, но основная идея проста. Она хорошо работает, если в качестве платформы выбирается конечномерная алгебра над конструктивным (например, конечным) полем. Конструктивность обеспечивает эффективную работу в соответствующем линейном пространстве, вычисление разложения по базису и т. п. В криптографии очень часто в качестве платформ шифрования предлагаются именно конечномерные алгебры над полями. Часто это алгебры матриц. Иногда это группы или полугруппы, допускающие представление матрицами над полем. Достаточно упомянуть группы кос, которые допускают точное представление матрицами над полем.

2. Криптографическая система Росошека [21, 22]

Описание

Установка

Пусть K — конечное ассоциативное кольцо с единицей, группа автоморфизмов $\operatorname{Aut} K$ которого некоммутативна. Пусть G — конечная абелева группа с некоммутативной группой автоморфизмов $\operatorname{Aut} G$. Через KG обозначим групповое кольцо группы G с коэффициентами из K.

Корреспондент A выбирает автоморфизм σ кольца K большого порядка, а также автоморфизм ν группы G также большого порядка. Через $C(\sigma)$ обозначим централизатор элемента σ в группе $\operatorname{Aut} K$, а через $C(\nu)$ — централизатор автоморфизма ν в $\operatorname{Aut} G$. Считаем, что оба этих централизатора строго больше, чем подгруппы $\operatorname{gp}(\sigma)$ и $\operatorname{gp}(\nu)$ соответственно.

Генерация ключей

Корреспондент A выбирает случайным образом автоморфизм $\tau \in C(\sigma)$, не принадлежащий $\operatorname{gp}(\sigma)$, и автоморфизм $\omega \in C(\nu)$, не принадлежащий $\operatorname{gp}(\nu)$. Затем он задаёт автоморфизм φ группового кольца KG следующим образом: для любого $h \in KG$ вида $h = a_{g_1} g_1 + \ldots + a_{g_n} g_n$, где $G = \{g_1, \ldots, g_n\}$, $a_{g_i} \in K$, $i = 1, \ldots, n$, полагает

$$h^\varphi = (a_{g_1}^\tau g_1^\omega + \ldots + a_{g_n}^\tau g_n^\omega)_\mu,$$

где μ — случайная подстановка на множестве номеров слагаемых в записи элементов группового кольца, которая в силу коммутативности сложения не меняет сам элемент h, а только форму его записи. Секретным ключом корреспондента A служит автоморфизм φ.

Далее A выбирает обратимый элемент $x \in KG$ и вычисляет $x^\varphi \in KG$.

Открытым ключом для A служит $(\sigma, \nu, x, x^\varphi)$.

Шифрование

Корреспондент Б для шифрования своего сообщения, закодированного в виде элемента m группового кольца KG, выбирает упорядоченную пару случайных натураль-

ных чисел (i, j), по которым определяет сессионный автоморфизм ψ группового кольца KG, полагая для любого элемента $h = a_{g_1}g_1 + \ldots + a_{g_n}g_n$, где $a_{g_1}, \ldots, a_{g_n} \in K$,

$$h^\psi = (a_{g_1}^{\sigma^i} g_1^{\nu^j} + \ldots + a_{g_n}^{\sigma^i} g_n^{\nu^j})_\xi, \tag{3}$$

где ξ есть случайная подстановка на множестве номеров слагаемых. После этого Б вычисляет $(x^{-1})^\psi$, используя открытый ключ А и автоморфизм ψ. Набор параметров (i, j, ψ) считается секретным сессионным ключом корреспондента Б.

Зашифрованное сообщение m имеет вид

$$c = ((x^{-1})^\psi, m(x^\varphi)^\psi). \tag{4}$$

Расшифрование

Корреспондент А, получив зашифрованное сообщение (4), вычисляет, пользуясь перестановочностью автоморфизмов φ и ψ, очевидной из их построения, элемент $((x^{-1})^\psi)^\varphi = ((x^{-1})^\varphi)^\psi$. Затем, умножив его справа на второй элемент набора c, вычисляет m.

К р и п т о г р а ф и ч е с к и й а н а л и з с и с т е м ы Р о с о ш е к а

Обозначим через $\sigma^i \wedge \nu^j, i, j \geqslant 0$, автоморфизмы алгебры KG, задаваемые указанным выше способом (3). Предположим, что K — алгебра над конечным полем F конечной размерности l и что любой автоморфизм кольца K является автоморфизмом K как алгебры над F. Это условие выполнено автоматически, если F — простое конечное поле. Поэтому достаточно требовать, чтобы K было алгеброй над простым конечным полем. В этом случае KG также естественно является алгеброй над F конечной размерности $m = l \cdot \mathrm{ord}(G)$, где $\mathrm{ord}(G)$ означает порядок группы G. Любой автоморфизм вида $\eta = \lambda \wedge \mu, \lambda \in \mathrm{Aut}K, \mu \in \mathrm{Aut}G$, будет автоморфизмом KG как алгебры над F.

Определим на группе Φ всех автоморфизмов вида $\sigma^i \wedge \nu^j$ для произвольного $r \geqslant 0$ *сферу* и *шар* радиуса r, полагая $S_r = \{\sigma^i \wedge \nu^j : i + j = r\}$ и $B_r = \bigcup\limits_{t=0}^{r} S_t$ соответственно. При этом $S_0 = B_0 = \{\sigma^0 \wedge \nu^0\} = \{1\}$.

Пусть x — фиксированный ненулевой элемент алгебры KG, выбранный корреспондентом А. Обозначим через x^Φ множество всех элементов алгебры KG вида $x^\eta, \eta \in \Phi$, другими словами — Φ-орбиту элемента x. Через $V = \mathrm{lin}_F(x^\Phi)$ обозначим линейное подпространство алгебры KG над полем F, порождённое множеством x^Φ.

Базис пространства V строим последовательно. Сначала полагаем $L_0 = \{x\}$. Затем расширяем L_0 до максимального линейно независимого множества L_1 подпространства $V_1 = \mathrm{lin}_F(x^{B_1})$. Для этого рассматриваем последовательно в соответствии с лексикографическим порядком элементы $x^{\sigma^i \wedge \nu^j}, i + j = 1$, включая в L_1 те из них, которые не выражаются линейно через уже включенные до них элементы. Пусть уже построен базис L_p подпространства $V_p = \mathrm{lin}_F(x^{B_p})$. Рассматриваем последовательно только те элементы вида $x^{\sigma^i \wedge \nu^j}, i + j = p + 1$, множества $x^{S_{p+1}}$, которые имеют предшественников, т. е. $x^{\sigma^{i-1} \wedge \nu^j}$ или $x^{\sigma^i \wedge \nu^{j-1}}$ в L_p. Если предшественники не включены в базис, значит, соответствующие им элементы линейно выражаются через уже рассмотренные элементы. Но тогда рассмотрение элемента $x^{\sigma^i \wedge \nu^j}, i + j = p + 1$, не имеет смысла, так как он также линейно выражается через уже рассмотренные элементы. Перебираем элементы последовательно в соответствии с лексикографическим порядком, каждый раз проверяя, выражается ли элемент линейно через уже построенную часть базиса L_{p+1}. Если не выражается, то включаем его в L_{p+1}, если выражается, то нет. Так как размерность пространства V не превышает m, то через не более чем m включений

возникнет ситуация, когда $L_p = L_{p+1}$, то есть на очередном $(p+1)$-м шаге базис не увеличится. Очевидно, что в этом случае $L_p = L$. Процесс построения L закончен.

Пусть $L = \{x^{\sigma^{q_i}\wedge\nu^{t_i}} : i = 1, \ldots, s\}$. Вычисляем соответствующее разложение

$$x^{\psi} = \sum_{i=1}^{s} \alpha_i x^{\sigma^{q_i}\wedge\nu^{t_i}}, \alpha_i \in F, i = 1, \ldots, s. \tag{5}$$

Подставим в правую часть выражения (5) вместо x элемент x^{φ}. Поскольку φ является автоморфизмом алгебры (достаточно даже — линейного пространства) KG над F и перестановочен с любым автоморфизмом из Φ, получаем

$$\sum_{i=1}^{s} \alpha_i (x^{\varphi})^{\sigma^{q_i}\wedge\nu^{t_i}} = \left(\sum_{i=1}^{s} \alpha_i x^{\sigma^{q_i}\wedge\nu^{t_i}}\right)^{\varphi} = (x^{\psi})^{\varphi} = (x^{\varphi})^{\psi}.$$

Элемента $(x^{\varphi})^{\psi}$ достаточно для получения m.

Комментарий

В работе [22] есть примеры, в которых в качестве K выбирается кольцо матриц $M_2(F_p)$, p — простое. Такое кольцо может рассматриваться как алгебра над F_p размерности 4. Если выбрать в нём произвольную матрицу a, а затем применить к ней автоморфизм σ^i, то образ a^{σ^i} можно довольно легко записать в виде линейной комбинации над F_p единичной матрицы и матриц g^{σ^j} при $j = 1, 2, 3$. При этом i может быть очень большим. В общем случае предложенная атака будет эффективной, если кольцо K является алгеброй достаточно малой размерности над F_p. При этом порядок группы G должен быть сравнительно небольшим. Эти требования выглядят достаточно естественными. Действительно, работа в групповых кольцах групп большого порядка, да ещё с использованием автоморфизмов, затруднена уже при шифровании и расшифровании. Поэтому основные методы скрытия в подобных системах обычно связывают с достаточно большим кольцом коэффициентов.

Заметим, что в общем случае не обязательно пытаться получить базис L полностью. Можно организовать процесс параллельного построения базиса и проверки выразимости через уже построенную часть элемента x^{ψ}.

3. Протокол выработки общего секретного ключа Маркова, Михалева, Грибова, Золотых и Скаженика на платформе лупы Муфанг [23]

Напомним вкратце некоторые определения [24 – 26].

Группоид — непустое множество G с заданной бинарной операцией $\cdot$. *Квазигруппой* называется группоид, в котором для любой пары элементов $g, f \in G$ однозначно разрешимы уравнения $xg = f$ и $gx = f$. *Лупой* называется квазигруппа с единицей. Лупа называется *лупой Муфанг*, если на ней выполняется тождество $(xy)(zx) = (x(yz))x$.

Приведём некоторые свойства лупы Муфанг [24 – 26]:

1) в лупе Муфанг любые два элемента порождают подгруппу, в частности, лупа Муфанг является лупой с ассоциативными степенями;

2) если для элементов $a, b, c \in G$ выполнено равенство $a(bc) = (ab)c$, то эти элементы порождают в G подгруппу.

Описание

Установка

Пусть G — лупа Муфанг, $a, b, c \in G$ — её элементы. Эти данные считаются известными.

Алгоритм

1) Корреспондент А выбирает тройку случайных натуральных чисел (m, k, n), затем вычисляет и посылает Б сообщение вида

$$(u_1, u_2) = (a^m b^k, b^k c^n).$$

2) Корреспондент Б выбирает тройку случайных чисел (r, l, s), вычисляет и посылает А сообщение

$$(v_1, v_2) = (a^r b^l, b^l c^s).$$

3) Получив сообщение от Б, корреспондент А вычисляет элементы

$$(a^m v_1)b^k, (b^k v_2)c^n.$$

4) Подобным же образом Б получает элементы

$$(a^r u_1)b^l, (b^l u_2)c^s.$$

Общим ключом корреспондентов А и Б служит

$$K_{AB} = (a^{m+r} b^{k+l})(b^{k+l} c^{n+s}).$$

Объяснение

Утверждение 1 [23]. Если G — лупа Муфанг, $a, b \in G$, то для любых показателей $k, l, m, n, r, s \geqslant 0$ выполнено равенство

$$(a^m(a^r b^s))b^n = a^m((a^r b^s)b^n) = (a^r(a^m b^n))b^s = a^r((a^m b^m)b^s) = a^{m+r} b^{n+s}.$$

Корреспондент А получает ключ K_{AB} с помощью следующих вычислений:

$$(a^m v_1)b^k = a^{m+r} b^{k+l}, (b^k v_2)c^s = b^{k+l} c^{n+s}, K_{AB} = (a^{m+r} b^{k+l}) \cdot (b^{k+l} c^{n+s}).$$

Корреспондент Б получает ключ K_{AB} совершенно аналогично.

Криптографический анализ протокола выработки общего секретного ключа Маркова, Михалева, Грибова, Золотых и Скаженика на платформе лупы Муфанг

Предположим, что лупа Муфанг G содержится в конечномерной алгебре размерности m над полем F. В работе [23], например, рассматриваются в качестве возможных платформ для протокола неассоциативные, конечные и простые лупы Муфанг, которые называются лупами *Пейджа*. Они могут быть вложены в алгебры Цорна размерности 8 над конечным полем.

Возьмём элементы a, b, c, фигурирующие в протоколе. Пусть m, k, n, r, l, s — параметры из протокола. Достаточно по известным элементам $u_1 = a^m b^k$, $u_2 = b^k c^n$, $v_1 = a^r b^l$, $v_2 = b^l c^s$ вычислить $a^{m+r} b^{k+l}$ и $b^{k+l} c^{n+s}$.

Сначала определим базисы подпространств $V_1 = \lin_F(a^i b^j : i, j \geqslant 0)$ и $V_2 = \lin_F(b^p c^q : p, q \geqslant 0)$ соответственно. Опишем построение базиса пространства V_1. (Базис пространства V_2 строится аналогично.) Для этого на множестве $\{a^i b^j\}$ для произвольного $r \geqslant 0$ определим сферу радиуса r, полагая $S_r = \{a^i b^j : i + j = r\}$, и шар радиуса r формулой $B_r = \bigcup_{t=0}^{r} S_t$. По определению, $S_0 = B_0 = \{1\}$. Пусть $L_0 = \{1\}$. Далее расширяем L_0 до $L_1 = \lin_F B_1$, просматривая последовательно по лексикографическому порядку элементы из S_1, включая в L_1 те из них, которые не выражаются

линейно через уже включенные. Если базис L_i пространства $\mathrm{lin}_F(B_i)$ уже определён, просматриваем последовательно элементы S_{i+1}, имеющие уже включенных в базис предшественников. У элемента a^ib^j предшественниками считаются $a^{i-1}b^j$ (если $i \neq 0$) и a^ib^{j-1} (если $j \neq 0$). Включаем в базис L_{i+1} те из них, которые не выражаются линейно через уже включенные. Если на некотором этапе $L_i = L_{i+1}$, то $L_i = L$.

Пусть $L = \{a^{p_i}b^{q_i} : i = 1, \ldots, t\}$. Вычисляем соответствующее разложение

$$a^m b^k = \sum_{i=1}^{t} \alpha_i a^{p_i} b^{q_i}, \alpha_i \in F, i = 1, \ldots, t. \tag{6}$$

Используя правую часть (6), где все параметры известны, и элемент $a^r b^l$, получим

$$\sum_{i=1}^{t} \alpha_i (a^{p_i}(a^r b^l)) b^{q_i} = \left(a^r \left(\sum_{i=1}^{t} a^{p_i} b^{q_i} \right) \right) b^l = (a^r (a^m b^k)) b^l = a^{m+r} b^{k+l}.$$

Точно так же получаем элемент $b^{k+l}c^{n+s}$. Затем вычисляем искомое произведение $K_{AB} = (a^{m+r}b^{k+l})(b^{k+l}c^{n+s})$.

4. Криптографическая система Грибова, Золотых и Михалева [27]

Описание

Установка

Пусть K — ассоциативное кольцо с единицей 1, G — лупа, KG — луповое кольцо.

Корреспондент А выбирает автоморфизм σ кольца K большого порядка, а также автоморфизм ν лупы G также большого порядка. Через $C(\sigma)$ обозначим централизатор элемента σ в группе $\mathrm{Aut}\, K$, а через $C(\nu)$ — централизатор автоморфизма ν в $\mathrm{Aut}\, G$. Считаем, что оба этих централизатора строго больше, чем подгруппы $\mathrm{gp}(\sigma)$ и $\mathrm{gp}(\nu)$ соответственно.

Генерация ключей

Корреспондент А выбирает случайным образом автоморфизм $\tau \in C(\sigma)$, не принадлежащий $\mathrm{gp}(\sigma)$, и автоморфизм $\omega \in C(\nu)$, не принадлежащий $\mathrm{gp}(\nu)$. Затем он задаёт автоморфизм φ лупового кольца KG следующим образом: для любого $h \in KG$ вида $h = a_{g_1}g_1 + \ldots + a_{g_n}g_n$, где $g_i \in G, a_{g_i} \in K, i = 1, \ldots, n$, определяет значение h^φ формулой

$$h^\varphi = a_{g_1}^\tau g_1^\omega + \ldots + a_{g_n}^\tau g_n^\omega.$$

Далее А выбирает элементы $x, a \in KG$ и вычисляет $x^\varphi, a^\varphi \in KG$.

Открытым ключом для А служит $(\sigma, \nu, x, x^\varphi, a, a^\varphi)$.

Шифрование

Корреспондент Б для шифрования своего сообщения, закодированного в виде элемента m групповой алгебры KG, выбирает две упорядоченные пары случайных натуральных чисел (i, j) и (k, l), по которым определяет сессионные автоморфизмы ψ и χ группового кольца KG, полагая для любого элемента $h = a_{g_1}g_1 + \ldots + a_{g_n}g_n$, где $g_i \in G$, $a_{g_i} \in K$, $i = 1, \ldots, n$,

$$h^\psi = a_{g_i}^{\sigma^i} g_1^{\nu^j} + \ldots + a_{g_n}^{\sigma^i} g_n^{\nu^j}, \quad h^\chi = a_{g_i}^{\sigma^k} g_1^{\nu^l} + \ldots + a_{g_n}^{\sigma^k} g_n^{\nu^l}.$$

После этого Б вычисляет x^ψ, a^χ, используя открытый ключ корреспондента А и автоморфизмы ψ и χ. Набор параметров (i, j, k, l, ψ, χ) считается секретным сессионным ключом.

Зашифрованное сообщение m имеет вид

$$c = (a^{\chi}x^{\psi}, m((a^{\varphi})^{\chi}(x^{\varphi})^{\psi})). \tag{7}$$

Б вычисляет также левый аннулятор $\mathrm{Ann}((a^{\varphi})^{\chi}(x^{\varphi})^{\psi})$. Если полученный аннулятор ненулевой, то проводится новая сессия с выбором других элементов a и x или же выбираются новые сессионные автоморфизмы ψ, χ.

Расшифрование

Корреспондент А, получив зашифрованное сообщение (7), вычисляет, пользуясь перестановочностью автоморфизмов φ, ψ и χ, очевидной из их построения, элемент $(a^{\chi}x^{\psi})^{\varphi} = (a^{\varphi})^{\chi}(x^{\varphi})^{\psi}$.

Для получения сообщения m корреспонденту А достаточно решить систему линейных уравнений с коэффициентами из кольца K. Однозначность решения обеспечивается тривиальностью левого аннулятора элемента $(a^{\varphi})^{\chi}(x^{\varphi})^{\psi}$.

К р и п т о г р а ф и ч е с к и й а н а л и з к р и п т о г р а ф и ч е с к о й с и с т е м ы Г р и б о в а, З о л о т ы х и М и х а л е в а

Как и в криптографическом анализе протокола выработки общего секретного ключа Маркова, Михалева, Грибова, Золотых и Скаженика на платформе лупы Муфанг, обозначим через $\sigma^i \wedge \nu^j, i, j \geqslant 0$, автоморфизмы кольца KG, задаваемые указанным выше способом. Предположим, что KG — алгебра над конечным полем F конечной размерности m. Также предполагаем, что любой из рассматриваемых автоморфизм кольца K будет автоморфизмом K как алгебры над F. Это условие выполнено автоматически, если F — простое конечное поле. Поэтому достаточно требовать, чтобы K было алгеброй над простым конечным полем. Определим на группе Φ всех автоморфизмов вида $\sigma^i \wedge \nu^j, i, j \geqslant 0$, для произвольного $r \geqslant 0$ сферу S_r и шар B_r радиуса r, как это было сделано в криптоанализе протокола Росошека, описанном выше.

Пусть z — некоторый фиксированный ненулевой элемент алгебры KG. Обозначим через z^{Φ} множество всех элементов алгебры KG вида $z^{\eta}, \eta \in \Phi$, другими словами — Φ-орбиту элемента z. Пусть теперь a, x — элементы из протокола, a^{Φ}, x^{Φ} — их Φ-орбиты, $a^{\Phi} \cdot x^{\Phi}$ — произведение Φ-орбит.Через $V = \mathrm{lin}_F(a^{\Phi} \cdot x^{\Phi})$ обозначим линейное подпространство алгебры KG над полем F, порождённое множеством $a^{\Phi} \cdot x^{\Phi}$.

Базис L пространства V строим последовательно. Сначала полагаем $L_0 = \{a \cdot x\}$, затем расширяем L_0 до максимального линейно независимого множества L_1 подпространства $V_1 = \mathrm{lin}_F(a \cdot x^{B_1} \cup a^{B_1} \cdot x)$. Для этого рассматриваем последовательно в соответствии с лексикографическим порядком элементы $a \cdot x^{\sigma^i \wedge \nu^j}$, $i + j = 1$, и $a^{\sigma^i \wedge \nu^j} \cdot x$, $i + j = 1$, включая в L_1 те из них, которые не выражаются линейно через уже включенные до них элементы. Пусть уже построен базис L_r подпространства $V_r = \mathrm{lin}_F(a^{B_q} \cdot x^{B_p} : p + q = r)$. Рассматриваем последовательно только те элементы вида $a^{\sigma^k \wedge \nu^l} \cdot x^{\sigma^i \wedge \nu^j}$, $k + l + i + j = r + 1$, которые имеют предшественников в L_r, т. е. либо элемент $a^{\sigma^{k-1} \wedge \nu^l} \cdot x^{\sigma^i \wedge \nu^j}$, либо элементы указанного вида, отвечающие наборам индексов $(k, l-1, i, j)$, $(k, l, i-1, j)$ или $(k, l, i, j-1)$. Перебираем их последовательно в соответствии с лексикографическим порядком, каждый раз проверяя, выражается ли элемент линейно через уже построенную часть базиса L_{r+1}. Если не выражается, то включаем его в L_{r+1}, если выражается, то нет. Так как размерность пространства V не превышает m, то через не более чем m включений возникнет ситуация, когда $L_r = L_{r+1}$, то есть на очередном $(r+1)$-м шаге базис не увеличится. Очевидно, что в этом случае $L_r = L$. Процесс построения L закончен.

Пусть $L = \{a^{\sigma^{p_i}\wedge\nu^{r_i}} \cdot x^{\sigma^{q_i}\wedge\nu^{t_i}} : i = 1, \ldots, s\}$. Вычисляем соответствующее разложение

$$a^{\chi} \cdot x^{\psi} = \sum_{i=1}^{s} \alpha_i a^{\sigma^{p_i}\wedge\nu^{r_i}} \cdot x^{\sigma^{q_i}\wedge\nu^{t_i}}, \quad \alpha_i \in F, \quad i = 1, \ldots, s. \tag{8}$$

Подставим в правую часть выражения (8) a^{φ} вместо a и x^{φ} вместо x. Поскольку φ перестановочен с любым автоморфизмом из Φ, получаем

$$\sum_{i=1}^{s} \alpha_i (a^{\varphi})^{\sigma^{p_i}\wedge\nu^{r_i}} \cdot (x^{\varphi})^{\sigma^{q_i}\wedge\nu^{t_i}} = \left(\sum_{i=1}^{s} \alpha_i a^{\sigma^{p_i}\wedge\nu^{r_i}} \cdot x^{\sigma^{q_i}\wedge\nu^{t_i}}\right)^{\varphi} = (a^{\chi} \cdot x^{\psi})^{\varphi} = (a^{\varphi})^{\chi} \cdot (x^{\varphi})^{\psi}.$$

Остаётся, решив систему линейных уравнений, получить m.

Как отмечается в [27], «это нетрудно сделать, если в качестве K взять конечномерную алгебру над полем. Можно в качестве K брать и другие кольца, главное, чтобы можно было решать систему линейных уравнений с коэффициентами из этого кольца».

5. Криптографическая система Маркова, Михалева, Грибова, Золотых и Скаженика на платформе градуированного кольца с мультипликативным базисом [23]

Предварительные сведения

Пусть R — ассоциативное кольцо с единицей $1 \in R$, G — группа в мультипликативной записи с нейтральным элементом (единицей) $e \in G$. Кольцо R называется *G-градуированным*, если существует такое семейство аддитивных подгрупп $\{R_\sigma, \sigma \in G\}$ аддитивной группы R, что $R = \bigoplus\limits_{\sigma\in G} R_\sigma$, $R_\sigma R_\tau \subseteq R_{\sigma\tau}$ для всех $\sigma, \tau \in G$. Ясно, что R_e — подкольцо R, а произвольная подгруппа R_σ — бимодуль над R_e для любого $\sigma \in G$.

Мультипликативным базисом конечномерной алгебры называется такой её базис B, что $B \cup \{0\}$ замкнуто относительно умножения.

Описание

Установка

Корреспондент A выбирает градуированное кольцо R относительно конечной группы G с конечным мультипликативным базисом $B = \{b_1, \ldots, b_n\}$. Предполагается, что группы автоморфизмов $\operatorname{Aut} B$ и $\operatorname{Aut} R_e$ достаточно богаты некоммутирующими элементами большого порядка с нетривиальными централизаторами большого порядка. Все эти величины R, G, B, а также градуировка открыты.

Корреспондент A выбирает автоморфизм σ кольца R_e большого порядка, а также автоморфизм ν базиса B также большого порядка. Через $C(\sigma)$ обозначим централизатор элемента σ в группе $\operatorname{Aut} R_e$, а через $C(\nu)$ — централизатор автоморфизма ν в $\operatorname{Aut} B$. Считаем, что оба этих централизатора строго больше, чем подгруппы $\operatorname{gp}(\sigma)$ и $\operatorname{gp}(\nu)$ соответственно.

Генерация ключей

Корреспондент A выбирает случайным образом автоморфизм $\tau \in C(\sigma)$, не принадлежащий $\operatorname{gp}(\sigma)$, и автоморфизм $\omega \in C(\nu)$, не принадлежащий $\operatorname{gp}(\nu)$. Затем он задаёт автоморфизм φ алгебры R следующим образом: для любого $h \in R$ вида $h = a_{b_1}b_1 + \ldots + a_{b_n}b_n$, где $a_{b_i} \in R_e$, $i = 1, \ldots, n$, определяет

$$h^{\varphi} = a_{b_1}^{\tau} b_1^{\omega} + \ldots + a_{b_n}^{\tau} b_n^{\omega}.$$

Далее A выбирает элементы $x, a \in R$ с нулевыми левыми аннуляторами и вычисляет $x^{\varphi}, a^{\varphi} \in R$.

Открытым ключом для А служит $(\sigma, \nu, x, x^\varphi, a, a^\varphi)$.

Шифрование

Корреспондент Б для шифрования своего сообщения, закодированного в виде элемента m кольца R, выбирает две упорядоченные пары случайных натуральных чисел (i, j) и (k, l), по которым определяет сессионные автоморфизмы ψ и χ кольца R, полагая для любого элемента $h = a_{b_1}b_1 + \ldots + a_{b_n}b_n$, где $a_{b_i} \in R_e$, $i = 1, \ldots, n$,

$$h^\psi = a_{b_i}^{\sigma^i} b_1^{\nu^j} + \ldots + a_{b_n}^{\sigma^i} b_n^{\nu^j}, \quad h^\chi = a_{b_i}^{\sigma^k} b_1^{\nu^l} + \ldots + a_{b_n}^{\sigma^k} b_n^{\nu^l}.$$

После этого Б вычисляет x^ψ, a^χ, используя открытый ключ А. Набор параметров (i, j, k, l, ψ, χ) считается секретным сессионным ключом.

Зашифрованное сообщение m имеет вид

$$c = (a^\chi \cdot x^\psi, m \cdot ((a^\varphi)^\chi \cdot (x^\varphi)^\psi)). \tag{9}$$

Расшифрование

Корреспондент А, получив зашифрованное сообщение (9), вычисляет, пользуясь перестановочностью автоморфизмов φ, ψ и χ, очевидной из их построения, элемент $(a^\chi \cdot x^\psi)^\varphi = (a^\varphi)^\chi \cdot (x^\varphi)^\psi$.

Для прочтения сообщения m корреспонденту А достаточно решить систему линейных уравнений с коэффициентами из кольца R_e. Однозначность решения обеспечивается тривиальностью левого аннулятора элемента $(a^\varphi)^\chi \cdot (x^\varphi)^\psi$, вытекающей из тривиальности левых аннуляторов его сомножителей.

Криптографический анализ системы Маркова, Михалева, Грибова, Золотых и Скаженика

Обозначим через $\sigma^i \wedge \nu^j$, $i, j \geqslant 0$, автоморфизмы кольца R, задаваемые указанным выше способом. Предположим, что R — алгебра над конечным полем F конечной размерности m. Также предполагаем, что любой автоморфизм R будет автоморфизмом R как алгебры над F. Это условие выполнено автоматически, если F — простое конечное поле. Поэтому достаточно требовать, чтобы R было алгеброй над простым конечным полем.

Определим на группе Φ всех автоморфизмов вида $\sigma^i \wedge \nu^j, i, j \geqslant 0$, для произвольного $r \geqslant 0$ сферу и шар радиуса r, как это было сделано в криптоанализе протокола Росошека и системы Грибова, Золотых и Михалева, описанных выше.

Дальнейший анализ буквально повторяет рассуждения из криптоанализа системы Грибова, Золотых и Михалева. Для элементов $a, x \in R$ вычисляется базис подпространства $V = \mathrm{lin}_F(a^\Phi \cdot x^\Phi)$: $L = \{a^{\sigma^{p_i} \wedge \nu^{r_i}} \cdot x^{\sigma^{q_i} \wedge \nu^{t_i}} : i = 1, \ldots, s\}$. Далее вычисляем соответствующее разложение вида (8). После подстановки в правую часть этого разложения элементов x^φ вместо x и a^φ вместо a и аналогичных вычислений получаем элемент $(a^\varphi)^\chi \cdot (x^\varphi)^\psi$. Затем решаем систему линейных уравнений, получая в итоге m.

6. Протоколы обмена ключом Махалабониса [28]

6.1. Протокол обмена ключом Махалабониса 1

Описание

Установка

Пусть G — группа, g — элемент в G. Пусть Φ и Ψ — две подгруппы группы автоморфизмов $\mathrm{Aut}\,(G)$ группы G, элементы которых попарно коммутируют друг с другом, т. е. для любых $\varphi \in \Phi$, $\psi \in \Psi$ выполняется $\varphi \cdot \psi = \psi \cdot \varphi$. Эти данные являются открытыми.

Генерация ключей

1) Корреспондент А случайным образом выбирает автоморфизм $\varphi \in \Phi$. Затем вычисляет g^{φ} и посылает этот результат по незащищённому каналу связи корреспонденту Б.

2) Корреспондент Б случайным образом выбирает автоморфизм $\psi \in \Psi$. Затем вычисляет g^{ψ} и посылает результат по незащищённому каналу связи корреспонденту А.

Распределение ключей

Корреспондент А вычисляет $K_A = (g^{\psi})^{\varphi}$. Корреспондент Б вычисляет $K_B = (g^{\varphi})^{\psi} = (g^{\psi})^{\varphi}$.

Общий ключ есть $K = K_A = K_B = (g^{\psi})^{\varphi}$.

Криптографический анализ протокола обмена ключом Махалабониса 1

Пусть G — подгруппа группы всех обратимых матриц $\mathrm{GL}_n(A)$ над алгеброй A конечной размерности l над полем F. Тогда размерность алгебры $\mathrm{M}_n(A)$ над полем F равна $m = l \cdot n$.

Для простоты считаем, что подгруппы Φ и Ψ группы автоморфизмов $\mathrm{Aut}(G)$, фигурирующие в протоколе, конечно порождены. Пусть $\Phi = \mathrm{gp}(\varphi_1, \ldots, \varphi_k)$ и $\Psi = \mathrm{gp}(\psi_1, \ldots, \psi_l)$. Предположим также, что автоморфизмы подгруппы Ψ естественно продолжаются до линейных преобразований линейного пространства $\mathrm{lin}_F(G)$, порождённого группой G в линейном пространстве алгебры $\mathrm{M}_n(A)$ над F.

Тогда для любого $r \in \mathbb{N}$ определим сферу $S_r(\Phi)$ радиуса r, состоящую из всех групповых слов от порождающих элементов подгруппы Φ длины r. Шар $B_r(\Phi)$ радиуса r определяется как $\bigcup\limits_{i=0}^{r} S_i(\Phi)$. Как и раньше, $S_0(\Phi) = \{1\}$, т. е. сфера радиуса 0 состоит из пустого слова, записывающего единицу группы. Аналогично определяются сферы $S_r(\Psi)$ и шары $B_r(\Psi)$ подгруппы Ψ.

Обозначим через g^{Φ} множество всех элементов группы G вида $g^{\eta}, \eta \in \Phi$, другими словами — это Φ-орбита элемента g. Через $V = \mathrm{lin}_F(g^{\Phi})$ обозначим линейное подпространство алгебры $\mathrm{M}_n(F)$ над полем F, порождённое множеством g^{Φ}.

Базис подпространства V строим последовательно. Сначала полагаем $L_0 = \{g\}$. Затем расширяем L_0 до базиса L_1 подпространства $V_1 = \mathrm{lin}_F(g^{B_1})$. Для этого рассматриваем последовательно в соответствии с лексикографическим порядком элементы g^{λ}, $\lambda \in S_1(\Phi)$, включая в L_1 те из них, которые не выражаются линейно через уже включенные до них элементы. Пусть уже построен базис L_p подпространства $V_p = \mathrm{lin}_F(g^{B_p})$. Рассматриваем последовательно только те элементы вида g^{λ}, $\lambda \in S_{p+1}(\Phi)$, которые имеют предшественников в L_p, т. е. если в λ подслово (начиная со второй буквы) λ_1 длины p определяет элемент $g^{\lambda_1} \in L_p$. Перебираем их последовательно в соответствии с лексикографическим порядком, каждый раз проверяя, выражается ли элемент линейно через уже построенную часть базиса L_{p+1}. Если не выражается, то включаем его в L_{p+1}, если выражается, то нет. Так как размерность пространства V не превышает m, то через не более чем m включений возникнет ситуация, когда $L_p = L_{p+1}$, то есть на очередном $(p+1)$-м шаге базис не увеличится. Очевидно, что в этом случае $L_p = L$. Процесс построения L закончен.

Пусть $L = \{g^{\lambda_i}, \lambda_i \in \Phi, i = 1, \ldots, s\}$. Вычисляем соответствующее разложение

$$g^{\varphi} = \sum_{i=1}^{s} \alpha_i g^{\lambda_i}, \quad \alpha_i \in F, \quad i = 1, \ldots, s. \tag{10}$$

Подставим в правую часть выражения (10) вместо g элемент g^{ψ}. Поскольку ψ перестановочен с любым автоморфизмом из Φ и по предположению продолжается до линейного преобразования пространства $\mathrm{lin}_F(G)$, получаем

$$\sum_{i=1}^{s} \alpha_i (g^{\psi})^{\lambda_i} = \left(\sum_{i=1}^{s} \alpha_i g^{\lambda_i} \right)^{\psi} = (g^{\varphi})^{\psi} = (g^{\psi})^{\varphi}.$$

Таким образом получен общий ключ протокола.

Комментарий

Предлагаемый криптоанализ, более точно — атака на протокол возможна при двух условиях: точной представимости группы G матрицами над конечномерной алгеброй над полем и возможности расширения хотя бы одной из групп Φ или Ψ до группы линейных преобразований подпространства $\mathrm{lin}_F(G)$ пространства $\mathrm{M}_n(A)$. Это условие не является ограничительным, если G — конечная группа. Действительно, тогда голоморф $\mathrm{Hol}(G)$ (полупрямое расширение группы G с помощью её группы автоморфизмов $\mathrm{Aut}\,(G)$) также конечен и поэтому допускает точное представление матрицами над конечным полем. Все автоморфизмы группы G индуцируются внутренними автоморфизмами группы $\mathrm{Hol}(G)$, т. е. сопряжениями. Но любое сопряжение определяет не только линейное преобразование, но и автоморфизм соответствующей алгебры матриц.

Автор [28] предлагает использовать в качестве G конечно порождённую нильпотентную группу, ограничиваясь, впрочем, конечными группами в более детальных рекомендациях. Однако, если G — конечно порождённая нильпотентная (или даже более общо — полициклическая) группа, то ее голоморф $\mathrm{Hol}(G)$ представим матрицами над кольцом целых чисел $\mathbb{Z}$ (значит, и над полем рациональных чисел $\mathbb{Q}$) по теореме Мерзлякова [29]. Это также снимает ограничение на возможность использования описанной атаки.

6.2. Протокол обмена ключом Махалабониса 2

Описание

Установка

Пусть G — группа, g — элемент в G. Пусть Φ и Ψ — две подгруппы группы автоморфизмов $\mathrm{Aut}\,(G)$ группы G, элементы которых попарно коммутируют друг с другом. Данные G, Φ, Ψ являются открытыми. Элемент g — секретный.

Алгоритм

1) Корреспондент А случайным образом выбирает автоморфизм $\varphi \in \Phi$. Затем вычисляет g^{φ} и посылает этот результат по незащищённому каналу связи корреспонденту Б.

2) Корреспондент Б случайным образом выбирает автоморфизм $\psi \in \Psi$. Затем вычисляет $(g^{\varphi})^{\psi}$ и посылает результат корреспонденту А.

3) А вычисляет обратный автоморфизм φ^{-1} и применяет его к последнему сообщению, получая $((g^{\varphi})^{\psi})^{\varphi^{-1}} = g^{\psi}$. Затем А берёт другой автоморфизм $\xi \in \Phi$, вычисляет $(g^{\psi})^{\xi}$ и посылает результат корреспонденту Б.

Распределение ключа

Корреспондент Б вычисляет ψ^{-1} и применяет его к последнему сообщению, получая в итоге $((g^{\psi})^{\xi})^{\psi^{-1}} = g^{\xi}$.

Это и есть общий ключ.

К р и п т о г р а ф и ч е с к и й а н а л и з п р о т о к о л а о б м е н а к л ю ч о м М а х а л а б о н и с а 2

Пусть G — подгруппа группы всех обратимых матриц $\mathrm{GL}_n(A)$ над алгеброй A конечной размерности l над полем F. Тогда размерность алгебры $\mathrm{M}_n(A)$ над полем F равна $m = l \cdot n$.

Для простоты считаем, что подгруппы Φ и Ψ группы автоморфизмов $\mathrm{Aut}(G)$, фигурирующие в протоколе, конечно порождены. Пусть $\Phi = \mathrm{gp}(\varphi_1, \ldots, \varphi_k)$ и $\Psi = \mathrm{gp}(\psi_1, \ldots, \psi_l)$. Предположим также, что автоморфизмы подгруппы Ψ естественно продолжаются до линейных преобразований линейного пространства $\mathrm{lin}_F(G)$, порождённого группой G в линейном пространстве алгебры $\mathrm{M}_n(A)$ над F.

Строим базис L подпространства $\mathrm{lin}_F((g^\varphi)^\psi)^\Phi$ точно так же, как в криптоанализе предыдущего протокола. Пусть $L = \{((g^\varphi)^\psi)^{\lambda_i} : \lambda_i \in \Phi, i = 1, \ldots, s\}$. Получим выражение

$$(g^\psi)^\xi = \sum_{i=1}^{s} \alpha_i ((g^\varphi)^\psi)^{\lambda_i} = \left(\sum_{i=1}^{s} \alpha_i (g^\varphi)^{\lambda_i} \right)^\psi, \quad \alpha_i \in F. \tag{11}$$

Отсюда следует равенство

$$g^\xi = \sum_{i=1}^{t} \alpha_i (g^\varphi)^{\lambda_i}.$$

Значит, общий ключ g^ξ получается подстановкой элемента g^φ вместо $(g^\varphi)^\psi$ в правую часть выражения (11).

Заключение

Итак, в работе представлен подход, позволяющий находить передаваемое сообщение или общий ключ в целом ряде криптографических протоколов, базирующихся на конечномерных алгебрах. В ряде случаев протокол, не использующий конечномерной алгебры, можно превратить в протокол, базирующийся на конечномерной алгебре. Например, протокол, основанный на группе кос, можно с помощью известного представления группы кос матрицами превратить в протокол, базирующийся на матричной алгебре над полем. Подобный перевод на матричную платформу почти всегда возможен, если используются конечные алгебраические структуры.

Отличительной особенностью подхода является то, что в нём не вычисляются некоторые ключевые параметры протокола, не решаются соответствующие задачи поиска. Обычное представление о необходимости, а не только достаточности их решения оказывается в целом ряде случаев неверным.

В некоторых достаточно простых случаях эта идея уже высказывалась. Так, анализируя протокол распределения ключей, предложенный У. Романчук и В. Устименко [30], авторы [31] предложили атаку, похожую на описанные выше, а именно: в протоколе из [30] берётся в качестве платформы группа $\mathrm{GL}_n(F)$ над конечным полем F. Затем выбираются две коммутирующие матрицы $C, D \in \mathrm{GL}_n(F)$. Пусть $g \in F^n$ — фиксированный вектор. Эти данные открыты.

Корреспондент А выбирает многочлен $P = P(C, D) \in F[x, y]$, вычисляет и посылает вектор gP корреспонденту Б, который в свою очередь выбирает многочлен $Q = Q(C, D) \in F[x, y]$, вычисляет вектор gQ и посылает его А. Корреспондент А вычисляет ключ $K_A = (gQ)P = gQP$, Б делает то же самое, получая $K_B = (gP)Q = gPQ$. Так как C и D коммутируют, их общим ключом будет вектор $K = K_A = K_B$.

Потенциальный взломщик, подсмотрев по открытой сети gP, gQ и открытые данные C, D и g, вычисляет матрицу X, такую, что X коммутирует с A и D и, кроме

этого, выполняется равенство $gQ = gX$. Так как условия на X линейны, такая матрица легко вычислима. Далее легко получить ключ: $(gP)X = gXP = gQP = K$.

ЛИТЕРАТУРА

1. *Diffie W. and Hellman M. E.* New directions in cryptography // IEEE Trans. Inform. Theory. 1976. V. 22. P. 644–654.
2. *Hellman M. E.* An overview of public key cryptography // IEEE Communication Magazine. 2002. Iss. 50. P. 42–49.
3. *Menezes A. and Vanstone S.* A note on cyclic groups, finite fields, and the discrete logarithm problem // Applicable algebra in Engineering, Communication and Computing. 1992. V. 3. P. 67–74.
4. *Menezes A. J. and Wu Y.-H.* The discrete logarithm problem in $\mathrm{GL}(n, q)$ // Ars Combinatoria. 1997. V. 47. P. 23–32.
5. *Романьков В. А.* Алгебраическая криптография. Омск: Изд-во Ом. ун-та, 2013. 207 с.
6. *Myasnikov A., Shpilrain V., and Ushakov A.* Group-based cryptography. (Advances courses in Math., CRM, Barselona). Basel, Berlin, New York: Birkhäuser Verlag, 2008. 183 p.
7. *Myasnikov A., Shpilrain V., and Ushakov A.* Non-commutative cryptography and complexity of group-theoretic problems. (Amer. Math. Soc. Surveys and Monographs). Providence, RI: Amer. Math. Soc., 2011. 385 p.
8. *ElGamal T.* A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE Trans. Inform. Theory. 1985. V. IT-31. No. 4. P. 469–472.
9. *Menezes A. J., van Oorschot P. C., and Vanstone S. A.* Handbook of Applied Cryptography. CRC Press, 1996. 816 p.
10. *Koblitz N.* A Course in Number Theory and Cryptology. New York, Heidelberg, Berlin: Springer Verlag, 1994.
11. *Романьков В. А.* Введение в криптографию. Курс лекций. М.: Форум, 2012. 240 с.
12. *Krammer D.* Braid groups are linear // Ann. Math. 2002. V. 151. P. 131–156.
13. *Dehornoy P.* Braid-based cryptography // Contemp. Math. 2004. V. 360. P. 5–33.
14. *Garber D.* Braid group cryptography. Lecture notes of Tutorials given at Braids PRIMA Summer School at Singapore, June 2007. `arXivmath.:0711.3941v2[cs.CR]` 27 Sep. 2008. P. 1–39.
15. *Mahlburg K.* An overview of braid groups cryptography // `www.math.wisc.edu/~boston/mahlburg.pdf`, 2004.
16. *Каргаполов М. И., Мерзляков Ю. И.* Основы теории групп. М.: Наука, 1972.
17. *Lennox J. C. and Robinson D. J. S.* The Theory of Infinite Soluble Groups. Oxford Math. Monographs. Oxford: Oxford Science Publications, 2004.
18. *Мегрелишвили Р. П., Джинджихадзе М. В.* Однонаправленная матричная функция для обмена криптографическими ключами, метод генерации мультипликативных матричных групп // Proc. Intern. Conf. SAIT 2011, May 23–28, Kyiv, Ukraine. P. 472.
19. *Megrelishvili R., Chelidze M., and Chelidze K.* On the construction of secret and public-key cryptosystems // Appl. Math., Inform. Mech. 2006. V. 11. No. 2. P. 29–36.
20. *Megrelishvili R., Chelidze M., and Besiashvili G.* One-way matrix function — analogy of Diffie — Hellman protocol // Proc. Seventh Intern. Conf. IES-2010, 28 Sept.–3 Oct., Vinnytsia, Ukraine, 2010. P. 341–344.
21. *Росошек С. К.* Криптосистемы групповых колец // Вестник Томского госуниверситета. Приложение. 2003. № 6. С. 57–62.
22. *Росошек С. К.* Криптосистемы в группах автоморфизмов групповых колец абелевых групп // Фундаментальная и прикладная математика. 2007. Т. 13. № 3. С. 157–164.

23. *Марков В. Т., Михалев А. В., Грибов А. В. и др.* Квазигруппы и кольца в кодировании и построении криптосхем // Прикладная дискретная математика. 2012. № 4(18). С. 32–52.
24. *Белоусов В. Д.* Основы теории квазигрупп и луп. М.: Наука, 1967.
25. *Pflugfelder H. O.* Quasigroups and Loops: Introduction. Berlin: Heldermann Verlag, 1990.
26. *Smith J. D. H.* An Introduction to Quasigroups and their representations. Boca Raton, FL: Chapman & Hall/CRC, 2007.
27. *Грибов А. В., Золотых П. А., Михалев А. В.* Построение алгебраической криптосистемы над квазигрупповым кольцом // Математические вопросы криптографии. 2010. Т. 1. № 4. С. 23–33.
28. *Mahalanobis A.* The Diffie-Hellman key exchange protocol and non-abelian nilpotent groups // Israel J. Math. 2008. V. 165. P. 161–187.
29. *Мерзляков Ю. И.* Целочисленное представление голоморфов полициклических групп // Алгебра и логика. 1970. Т. 9. № 5. С. 539–558.
30. *Romanczuk U. and Ustimenko V.* On the $PSL2(q)$, Ramanujan graphs and key exchange protocols // `http://aca2010.info/index.php/aca2010/aca2010/paper/viewFile/80/3`.
31. *Blackburn S. R., Cid C., and Mullan C.* Cryptanalysis of three matrix-based key establishment protocols // J. Mathematical Cryptology. 2011. V. 5. P. 159–168.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.94

ОСНОВНЫЕ ЭЛЕМЕНТЫ МАНДАТНОЙ СУЩНОСТНО-РОЛЕВОЙ ДП-МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ В СУБД POSTGRESQL ОС СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ ASTRA LINUX SPECIAL EDITION

А. В. Шумилин

ОАО «НПО РусБИТех», г. Москва, Россия

E-mail: a.shumilin@rusbitech.ru

Рассмотрен подход к разработке на основе мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в операционной системе (ОС) семейства Linux (МРОСЛ ДП-модели) новой ДП-модели, адаптированной для реализации в СУБД PostgreSQL, применяемой в ОС специального назначения Astra Linux Special Edition (PostgreSQL ДП-модели). Описываются её новые элементы, отличия от МРОСЛ ДП-модели и от известных автору моделей управления доступом в СУБД, в том числе учитывающие особенности управления доступом к данным, хранящимся и обрабатываемым в исследуемой СУБД.

Ключевые слова: *компьютерная безопасность, системы управления базами данных, ролевая ДП-модель, Astra Linux.*

1. Введение. Необходимость адаптации модели. Этапы разработки

При разработке автоматизированных систем в защищённом исполнении необходимо обеспечивать соответствие требованиям по защите информации от несанкционированного доступа (НСД). Эти требования зависят от конфиденциальности информации и характера доступа к ней пользователей и приведены в руководящих документах ФСТЭК России [1, 2]. Выполнение требований должно, в том числе, подтверждаться сертификатом соответствующей системы сертификации на программное обеспечение. Для обеспечения выполнений указанных требований разработчики защищённых программных комплексов (ЗПК) применяют те или иные из существующих классических моделей управления доступом зачастую без учёта всех условий и особенностей функционирования современных программно-аппаратных средств. При этом больше внимания уделяется созданию непосредственно механизмов защиты, чем построению непротиворечивых формальных моделей логического управления доступом в разрабатываемых системах. Применение моделей, позволяющих провести анализ этих механизмов безопасности, при условии обеспечения гарантий проектирования и реализации, повышает доверие к безопасности создаваемых ЗПК. Следует отметить, что в современных условиях уже недостаточно обеспечения управления доступом в соответствии с дискреционными и мандатными моделями управления доступом без учёта условий возникновения запрещённых информационных потоков.

В настоящее время осуществляются попытки построения современных формальных моделей, объединяющих все аспекты дискреционного, мандатного и ролевого

управления доступом с учётом безопасности информационных потоков, для применения в реальных защищённых ОС. Например, при создании перспективной операционной системы специального назначения (ОС СН) Astra Linux Special Edition используется мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в операционных системах семейства Linux [3 – 5].

Поскольку неотъемлемой частью системы обработки данных в автоматизированных системах являются системы управления базами данных (СУБД), они также должны отвечать требованиям по защите информации от НСД, применяемым к создаваемой системе в целом. При построении автоматизированных систем в защищённом исполнении, как правило, используются СУБД, функционирующие под управлением применяемых ОС. В этом случае для реализации механизмов защиты в СУБД руководствуются моделью управления доступом, используемой в ОС.

Таким образом, в условиях постоянно возрастающих требований к обеспечению защиты информации при её хранении и обработке реализация механизмов управления доступом в СУБД не только не теряет свою актуальность, но и требует новых подходов, согласующихся с теми моделями управления доступом, которые реализуются в среде функционирования СУБД, которой является ОС.

В настоящее время для обеспечения единого подхода к управлению доступом в СУБД и ОС СН Astra Linux Special Edition автором осуществляется адаптация МРОСЛ ДП-модели с учетом специфики реляционных баз данных. В ходе работы использовались как классические базовые модели управления доступом семейства RBAC [6, 7], так и современные результаты теоретического моделирования управления доступом в СУБД [8, 9]. Кроме того, учитывались подходы к реализации управления доступом в СУБД Oracle, SQL Server и PostgreSQL, базирующиеся на требованиях стандарта SQL [10 – 13].

Целесообразность адаптации МРОСЛ ДП-модели и построения на её основе новой PostgreSQL ДП-модели связана со следующими особенностями управления доступом к данным, хранящимся и обрабатываемым в СУБД:

— первичная субъект-сессия СУБД создается субъект-сессией ОС;
— в качестве множества сущностей рассматриваются объекты реляционных баз данных;
— доступ к данным осуществляется с помощью языка запросов SQL;
— множество видов доступа, прав доступа и правила распространения прав доступа регламентируются стандартом SQL;
— в СУБД содержится не только пользовательская, но и служебная информация, описывающая структуру пользовательских данных.

Применение специальных моделей управления доступом в СУБД наряду с реализацией выполнения заданных требований по защите информации должно обеспечивать без потери производительности сохранение неотъемлемых качеств СУБД, таких, как:

— целостность и согласованность данных;
— управление данными и их структурой с помощью языка запросов;
— резервное копирование и восстановление.

Для обеспечения перечисленных качеств язык запросов должен быть расширен соответствующими командами санкционированного изменения правил разграничения доступа (ПРД), позволяющими не только гибко управлять ими, но и поддерживать резервное копирование и восстановление баз данных без потери информации о ПРД.

Создание эффективных механизмов управления доступом в реляционных СУБД требует не только хорошего знания реализуемых формальных моделей, но и детального анализа архитектуры и особенностей функционирования реальных систем обработки данных [14 – 17].

В связи со сложностью решение поставленной задачи выполняется поэтапно.

На первом этапе произведён анализ существующих механизмов управления доступом в рассматриваемой СУБД и возможности использования элементов МРОСЛ ДП-модели.

На втором этапе строится PostgreSQL ДП-модель, при этом формулируются определения и положения, необходимые для описания специфики СУБД ОС СН, разрабатываются правила преобразования состояния системы в рамках модели для нескольких существенных операций SQL. Для оценки возможности использования модели на её основе реализуется механизм управления доступом в реальной СУБД, что требует решения многих практических задач [18].

На третьем этапе предполагается расширение PostgreSQL ДП-модели правилами преобразования для большинства команд SQL, формулирование и обоснование в рамках модели, как минимум, достаточных условий безопасности механизма управления доступом и информационными потоками в рассматриваемых СУБД и ОС СН.

2. Элементы модели (описание состояния системы)

При разработке PostgreSQL ДП-модели были взяты за основу элементы МРОСЛ ДП-модели, задающие мандатное и ролевое управление доступом и информационными потоками. При этом для упрощения не рассматривались системы ограничений ролевого управления доступом и мандатный контроль целостности, которому соответствуют некоторые механизмы современных ОС (MIC — Mandatory Integrity Control и UAC — User Account Control в ОС семейства Microsoft Windows).

В отличие от МРОСЛ ДП-модели, для ОС в разрабатываемой модели роли трактуются не как сущности-объекты файловой системы, а как сущности-объекты СУБД. Множество ролей СУБД не является подмножеством ролей ОС, то же относится и к множеству административных ролей. Особые административные роли МРОСЛ ДП-модели для ОС не являются ролями СУБД и не могут быть получены пользователем СУБД непосредственно. Для обеспечения поддержки этих административных функций в PostgreSQL ДП-модели вводятся соответствующие параметры субъект-сессии СУБД, которые наследуются при создании субъект-сессии СУБД от субъект-сессии ОС.

В качестве множества сущностей (объектов и контейнеров) с заданной на нём иерархической структурой рассматриваются носящие подобный характер объекты реляционных баз данных (БД), применяемые в СУБД PostgreSQL. Следует отметить, что поскольку записи базы данных содержат в своём составе мандатный уровень конфиденциальности, то они рассматриваются в модели в качестве объектов, а содержащие их таблицы — соответственно в качестве контейнеров.

Системный каталог (метаданные) рассматривается как самостоятельная БД, реализованная с помощью средств СУБД. При этом все операции с этой БД осуществляются с помощью специальных конструкций языка запросов SQL или привилегированным пользователем в специальном режиме. В модели рассматривается управление доступом к пользовательским данным. Доступ к таблицам, записям таблиц и остальным объектам системного каталога осуществляется аналогично, за исключением того, что они считаются принадлежащими привилегированному пользователю.

В отличие от файловой системы ОС, иерархия объектов в СУБД носит более детерминированный характер, так как имеет ограниченную степень вложенности разнотипных объектов (например, база данных/схема/таблица/запись). Это позволяет привнести в модель элементы типизированной матрицы доступа, так как каждая операция SQL содержит информацию о типе объекта. Таким образом, PostgreSQL ДП-модель была расширена соответствующими определениями иерархии типов сущностей. Используем следующие обозначения (аналогичные применяемым в исходной МРОСЛ ДП-модели и работах [3, 8, 9]):

- $E = O \cup C$ — множество сущностей, где O — множество объектов, а C — множество контейнеров и $O \cap C = \varnothing$;
- S — множество субъект-сессий пользователей;
- T — множество типов сущностей;
- $type : E \to T$ — функция типов сущностей (по аналогии с [8]);
- R — множество ролей;
- AR — множество административных ролей, при этом по определению $AR \cap R = \varnothing$;
- $H_R : R \cup AR \to 2^R \cup 2^{AR}$ — функция иерархии ролей и административных ролей;
- U — множество учётных записей пользователей, при этом учётные записи в PostgreSQL являются подмножеством ролей, т. е. $U \subseteq R \cup AR$;
- $user : S \to U$ — функция принадлежности субъект-сессии учётной записи пользователя, задающая для каждой субъект-сессии учётную запись пользователя, от имени которой она активизирована;
- $R_a = \{read_a, write_a, append_a\}$ — множество видов доступа;
- $R_f = \{write_m, write_t\}$ — множество видов информационных потоков (по памяти и по времени);
- R_r — множество видов прав доступа, элементы которого с учётом специфики рассматриваемой СУБД будут определены далее;
- $R_{raf} = R_r \cup R_a \cup R_f$ — множество видов прав доступа, видов доступа и видов информационных потоков, при этом множества R_r, R_a и R_f попарно не пересекаются;
- $P \subseteq E \times R_r$ — множество прав доступа к сущностям;
- $PA : R \cup AR \to 2^P$ — функция прав доступа к сущностям ролей и административных ролей;
- $F \subseteq (E \cup R \cup AR) \times (E \cup R \cup AR) \times R_f$ — множество информационных потоков;
- $A \subseteq S \times E \times R_a$ — множество доступов субъект-сессий к сущностям;
- $AA \subseteq S \times (R \cup AR) \times R_a$ — множество доступов субъект-сессий к ролям или административным ролям;
- L_U — множество учётных записей доверенных пользователей;
- N_U — множество учётных записей недоверенных пользователей, при этом по определению справедливы равенства $L_U \cup N_U = U$, $L_U \cap N_U = \varnothing$;
- L_S — множество доверенных субъект-сессий;
- N_S — множество недоверенных субъект-сессий, при этом по определению справедливо равенство $L_S \cap N_S = \varnothing$.

Сохранены все определения исходной модели, касающиеся мандатного контроля конфиденциальности:

- $(LC, \leqslant)$ — решётка многоуровневой безопасности уровней конфиденциальности;

— $(f_u, f_e, f_r, f_s) \in FC$ — четвёрка функций уровней конфиденциальности:
  — $f_u : U \to LC$ — функция, задающая для каждой учётной записи пользователя её уровень доступа — максимальный разрешённый уровень доступа субъект-сессий, функционирующих от её имени;
  — $f_e : E \to LC$ — функция, задающая уровень конфиденциальности для каждой сущности;
  — $f_r : R \cup AR \to LC$ — функция, задающая для каждой роли или административной роли её уровень конфиденциальности;
  — $f_s : S \to LC$ — функция, задающая для каждой субъект-сессии её текущий уровень доступа;
— $CCR : E \cup R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая способ доступа к сущностям внутри контейнеров или ролям в иерархии ролей (с учётом их мандатных уровней конфиденциальности).

Для учёта специфики PostgreSQL дополнительно детализируем состав сущностей, их типов, а также соответствующие функции иерархии:

— $O = O_d \cup O_{col} \cup O_{proc} \cup O_{trg} \cup O_{cnstr} \cup O_{typ} \cup O_{seq} \cup O_{blob} \cup O_{rule} \cup O_{spc} \cup O_{glob}$,
где O_d — множество сущностей данных, соответствующих записям таблиц; O_{col} — множество столбцов; O_{proc} — множество хранимых процедур; O_{trg} — множество триггеров; O_{cnstr} — множество ограничений целостности; O_{typ} — множество используемых в СУБД типов данных; O_{seq} — множество последовательностей; O_{blob} — множество больших двоичных объектов; O_{rule} — множество правил преобразования (*RULE*); O_{spc} — множество табличных пространств (*TABLESPACE*); O_{glob} — множество глобальных объектов (*CAST*, *EXTENSION*, *FOREIGN DATA WRAPPER*, *FOREIGN SERVER*, *LANGUAGE*);
— $coltype : O_{col} \to O_{typ}$ — функция типов столбцов, сопоставляющая каждому столбцу его тип данных (домен);
— $C = C_{cl} \cup C_{db} \cup C_{nsp} \cup C_{rel}$,
где C_{db} — множество контейнеров, соответствующих базам данных; C_{nsp} — множество схем; C_{rel} — множество отношений, включающих таблицы и представления (далее — таблицы); C_{cl} — множество кластеров СУБД PostgreSQL. Кластер является в модели корневым контейнером, в котором располагаются глобальные объекты, такие, как базы данных, табличные пространства, глобальные описания и роли (субъекты);
— $T = \{cl, spc, db, nsp, rel, col, proc, trg, cnstr, typ, seq, blob, rule, glob, d\}$ — множество типов сущностей СУБД (соотносятся с одноимёнными индексами сущностей);
— $H_T : T \to 2^T$ — функция иерархии типов сущностей СУБД (сопоставляющая каждому типу сущности множество типов сущностей $H_T(t) \subset T$, которые в ней могут содержаться согласно особенностям реализации СУБД PostgreSQL), такая, что:
  — если $t \notin \{cl, db, nsp, rel\}$, то $H_T(t) = \varnothing$;
  — если $t = cl$, то $H_T(t) = \{db, spc\}$;
  — если $t = db$, то $H_T(t) = \{nsp, blob, glob\}$;
  — если $t = nsp$, то $H_T(t) = \{rel, proc, type, seq\}$;
  — если $t = rel$, то $H_T(t) = \{d, col, trg, cnstr, type, rule\}$;
— $H_E : E \to 2^E$ — функция иерархии сущностей (сопоставляющая каждой сущности $e \in E$ множество сущностей $H_E(e) \subset E$, непосредственно в ней содержащихся), удовлетворяющая условиям:

- если сущность $e \in H_E(c)$, то $e < c$ и не существует сущности-контейнера $d \in C$, такой, что $e < d$, $d < c$;
- для любых сущностей $e1, e2 \in E$, $e1 < e2$, выполняется $H_E(e1) \cap H_E(e2) = \varnothing$;
- для любых сущностей $e1, e2 \in E$, $e1 < e2$, выполняется $type(e1) \in H_T(type(e2))$ (контроль по типу сущностей);
- если $o \in O$, то справедливо равенство $H_E(o) = \varnothing$.

В исследуемой СУБД применяются дискреционные и ролевые механизмы управления доступом, отличные от механизмов, используемых в ОС. Определено больше видов прав доступа, и они могут применяться по-разному для разных типов сущностей. Язык запросов SQL позволяет гибко управлять правами доступа к объектам и к ролям. Но эта гибкость с точки зрения безопасности информационных потоков является недостатком: множество ролей SQL не разделяется по административному признаку, что в сочетании с возможностью владельца сущности предоставлять доступ к ней произвольным ролям приводит к неконтролируемому распространению прав доступа и возможности создания запрещённых информационных потоков. Для устранения этого было предложено ввести в модель (по аналогии с МРОСЛ ДП-моделью) разделение ролей SQL на административные и неадминистративные с введением ограничений на исполнение команд и изменение правил управления доступом для неадминистративных ролей.

Отличие предлагаемой модели заключается в том, что в ней, опираясь на существующие модели ролевого управления доступом, делается попытка сочетать исходную МРОСЛ ДП-модель с требованиями стандарта на язык запросов SQL [12, 13]. Несмотря на введённое разделение ролей на административные и неадминистративные, управление правами доступа и ролями сохраняется согласно реализации PostgreSQL.

В классической модели управления доступом в СУБД присутствует роль привилегированного пользователя (суперпользователя), которому разрешены все операции в СУБД. В СУБД PostgreSQL существуют особая учётная запись postgres, которая носит такой характер, и особая привилегия роли *SUPERUSER* (*CREATEUSER*), предоставляющая роли аналогичные права.

С учётом специфики СУБД PostgreSQL зададим множество видов прав доступа:

- $R_r = \{select_r, update_r, insert_r, delete_r, truncate_r, references_r, trigger_r, create_r, usage_r, connect_r, execute_r, createdb_r, login_r, createrole_r, superuser_r, own_r\}$.

  В отличие от [9], вместо функции *owner* для задания владельца сущности используется принятое в ДП-моделях [7] право доступа владения own_r, так как оно точнее отражает специфику работы механизмов разграничения доступа в СУБД PostgreSQL. При этом выполняются следующие условия:
  - в каждый отдельный момент времени только одна роль может обладать правом владения к конкретной сущности;
  - роль, создавшая сущность, автоматически получает право владения к ней;
  - принадлежащие таблице объекты (доступ к которым явно не управляется) считаются принадлежащими владельцу таблицы, т. е. для $t \in C_{rel}$, $e \in H_E(t)$, $r \in R \cup AR$ выполняется условие: если $(t, own_r) \in PA(r)$, то $(e, own_r) \in PA(r)$;
  - для глобальных объектов, не имеющих владельца (кластер, *CAST* и т. п.) наличие права доступа $superuser_r$ эквивалентно наличию права доступа владения own_r;
- $R_{ck} = \{createdb_r, login_r, createrole_r, superuser_r\}$ — множество видов прав доступа, применимых к кластеру;

— $R_{db} = \{create_r, usage_r, connect_r, own_r\}$ — множество видов прав доступа, применимых к базе данных;
— $R_{spc} = \{create_r, own_r\}$ — множество видов прав доступа, применимых к табличному пространству;
— $R_{nsp} = \{create_r, usage_r, own_r\}$ — множество видов прав доступа, применимых к схеме;
— $R_{rel} = \{select_r, update_r, insert_r, delete_r, truncate_r, references_r, trigger_r, own_r\}$ — множество видов прав доступа, применимых к таблицам;
— $R_{col} = \{select_r, update_r, insert_r, references_r\}$ — множество видов прав доступа, применимых к столбцам таблиц;
— $R_{seq} = \{select_r, update_r, usage_r, own_r\}$ — множество видов прав доступа, применимых к последовательностям;
— $R_{type} = \{usage_r, own_r\}$ — множество видов прав доступа, применимых к типам и доменам;
— $R_{proc} = \{execute_r, own_r\}$ — множество видов прав доступа, применимых к процедурам;
— $R_{blob} = \{select_r, update_r, own_r\}$ — множество видов прав доступа, применимых к большим двоичным объектам;
— $ALL_r : E \to 2^{R_r}$ — функция применимых к сущности видов прав доступа, такая, что для каждой $e \in E$ по определению справедливо $ALL_r(e) = R_\alpha \alpha \in type(e)$.

В СУБД PostgreSQL в каждый момент времени субъект-сессия может исполняться только с одной активной ролью. Активация другой роли (*SET ROLE*) в модели рассматривается как инициация новой субъект-сессии. Согласно стандарту SQL, сессия в случае наличия активной роли исполняется от её имени, в противном случае — от имени роли учётной записи пользователя, её активизировавшего (функция *user*). В отличие от [9], вводится функция *role*:

— $role : S \to R \cup AR$ — функция принадлежности субъект-сессии роли, задающая для каждой субъект-сессии активную роль, с которой она выполняется.

Функция *role* может меняться при переходе системы из одного состояния в другое, так как в зависимости от режима выполнения кода сущности-процедуры возможно применение прав, отличных от прав вызвавшей такую сущность роли.

В СУБД существуют следующие особенности применения доступов субъект-сессии к сущностям как в отношении пользовательских данных, так и в отношении системных объектов (метаданных):

— для ролей рассматриваются только доступы на чтение и запись, так как в СУБД PostgreSQL отсутствует понятие владельца роли, при этом доступом на запись к роли может обладать только доверенная субъект-сессия;
— для сущностей-контейнеров доступ на чтение позволяет получить список содержащихся в них сущностей, на добавление — возможность создать в них новую сущность, на запись — изменить или удалить сущность в их составе;
— операции по модификации свойств сущностей, в том числе состава таблицы (столбцы, ограничения и т. п.), требуют у субъект-сессии наличия доступного по иерархии права доступа владения;
— операции с записями таблиц приводятся к видам доступа общепринятым образом: *SELECT* — чтение, *INSERT* — добавление, *UPDATE*, *DELETE* — запись.

По аналогии с СУБД ДП-моделью [9] используем обозначения:

— $execute\_as : O_{proc} \to \{as\_caller, as\_owner\}$ — функция, задающая режим выполнения кода сущности-процедуры p субъект-сессией s при следующих условиях:
  — если $p \in O_{proc}$ и $execute\_as(p) = as\_caller$, то код исполняется от имени роли $role(s)$;
  — если $p \in O_{proc}$ и $execute\_as(p) = as\_owner$, то код исполняется от имени роли, непосредственно имеющей право доступа владения к p, т. е. роли $r \in R \cup AR$, для которой выполняется $(p, own_r) \in PA(r)$;
— $operations : O_p \to OP^*$ — функция, задающая для каждого объекта-процедуры конечную последовательность де-юре правил преобразования состояний модели, выполняющихся при её активизации, где OP — множество правил преобразования состояний модели;
— $triggers : C_{rel} \times \{update_r, insert_r, delete_r, truncate_r\} \to O^*_{trg}$ — функция, задающая для таблицы конечную последовательность триггеров, выполняющихся при реализации к нему права доступа определённого вида.

Один и тот же триггер в СУБД PostgreSQL может вызываться при реализации любого из заданных при его создании вида права доступа. При этом последовательность вызова триггеров для одной таблицы определяется в алфавитном порядке их имён. С триггером ассоциирована реализующая его функция:

— $trigger\_proc : O_{trg} \to O_{proc}$ — функция, задающая для триггера реализующую его процедуру.

В СУБД PostgreSQL существует возможность задавать для таблицы специальное правило, которое вызывается при реализации любого из заданных при его создании вида права доступа (аналогично триггеру). При этом последовательность вызова правил для одной таблицы определяется в алфавитном порядке их имён. Обозначим:

— $rules : C_{rel} \times \{select_r, update_r, insert_r, delete_r\} \to O^*_{rule}$ — функция, задающая для таблицы конечную последовательность правил, выполняющихся при реализации к ней права доступа определённого вида;
— $rule\_op : O_{rule} \to \{access\_select, access\_insert, access\_update, access\_delete\}$ — функция, задающая для правила таблицы соответствующее правило преобразования состояния модели из набора правил работы с данными в таблице.

Для обеспечения поддержки специальных административных ролей МРОСЛ ДП-модели в PostgreSQL ДП-модели вводятся соответствующие параметры субъект-сессии СУБД, которые наследуются при создании субъект-сессии СУБД от субъект-сессии ОС:

— $sop : S \to \{downgrade\_admin\_sop, \ldots\}$ — функция получения параметров новой субъект-сессии СУБД из субъект-сессии ОС, такая, что при наличии у субъект-сессии ОС доступа на чтение к административным ролям ОС (например, *downgrade_admin_role* в МРОСЛ ДП-модели) она отображает их в соответствующие параметры (например, *downgrade_admin_sop*);
— $session\_options : S \to \{downgrade\_admin\_sop, \ldots\}$ — функция, определяющая для каждой субъект-сессии её параметры.

Для описания процесса делегирования прав доступа аналогично [9] введено множество $Gr \subseteq (R \cup AR) \times P$ — множество прав доступа на сущности, которые могут быть переданы ролью другой роли. Могут быть переданы только права доступа, которыми роль уже обладает к сущностям, которыми владеет, либо полученные к другим сущностям с опцией *WITH_GRANT_OPTION*, т. е. $Gr(r) \subseteq PA(r)$.

Опция *WITH_ GRANT_ OPTION* даёт возможность получающей право роли передавать полученное право другим ролям. Владелец сущности по умолчанию получает все права на неё и может передавать их другим ролям. При отборе прав доступа возможен отбор опции *WITH_ GRANT_ OPTION*. Право владения не может быть делегировано. В СУБД используется ключевое слово *PUBLIC*, обозначающее права доступа всех существующих ролей — аналог прав для всех в ОС. Данный элемент не является сущностью (ролью или группой, хотя и может рассматриваться как встроенная группа, в которую входят все существующие роли и те, что будут созданы в будущем) и соответственно не может получить опцию делегирования прав доступа *WITH_ GRANT_ OPTION*. Для отражения этой особенности СУБД в модели задаётся следующая роль:

— $public \in R$ — особая наименьшая в иерархии роль, используемая для задания общих для всех прав доступа, такая, что
  — для любой $r \in R \cup AR$ выполняется $public \in H_R(r)$;
  — не существует такой роли $r \in R \cup AR$, что $r < public$.

При создании некоторых сущностей ряд видов прав доступа предоставляется всем ролям с помощью *PUBLIC* ($connect_r$ — для баз данных, $execute_r$ — для процедур и $usage_r$ — для некоторых глобальных объектов). Состав прав доступа по умолчанию может быть изменён администратором.

В отличие от МРОСЛ ДП-модели, осуществляется возврат к следующим обозначениям:

— $UA : U \to 2^R$ — функция авторизованных ролей учётной записи пользователя, задающая для каждой учётной записи пользователя множество ролей, на которые она может быть авторизована;
— $AUA : U \to 2^{AR}$ — функция авторизованных административных ролей учётной записи пользователя, задающая для каждой учётной записи пользователя множество административных ролей, на которые она может быть авторизована.

Аналогично передаче прав доступа осуществляется и управление иерархией ролей. Для описания этого процесса дополнительно введено множество Adm:

— $Adm \subseteq (R \cup AR) \times (R \cup AR)$ — множество ролей, доступных для делегирования членства, при этом без административного права $createrole_r$ может быть передано только членство в ролях, полученное с опцией *WITH_ ADMIN_ OPTION*.

Управление ролями разрешено только роли, имеющей административное право доступа $createrole_r$ или получившей членство с опцией *WITH_ ADMIN_ OPTION*. Управление ролями, имеющими право доступа $superuser_r$, разрешено только ролям, в свою очередь имеющим это право доступа. При отборе членства возможен отбор опции *WITH_ ADMIN_ OPTION*.

Как было сказано ранее, для предотвращения неконтролируемого распространения прав доступа предлагается разделить роли SQL на административные и неадминистративные. При этом административные роли могут быть использованы только доверенными субъект-сессиями. Это отличие от требований SQL влечёт необходимость доработки соответствующих механизмов управления ролями в СУБД.

Особенностью реализации наследования прав доступа в иерархии ролей в СУБД PostgreSQL является наличие признака, разрешающего или запрещающего подобное наследование. Права доступа наследуются только в случае наличия у роли свойства *INHERIT*. Для описания подобного поведения введена функция $role\_inherit$:

— $role\_inherit : R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция наследования прав доступа, задающая для каждой роли режим их наследования; если значение функции равно `true`, то роль, стоящая в иерархии выше текущей, наследует её права доступа, иначе нет. При этом $role\_inherit(public) = \texttt{true}$.

Для упрощения выражений в правилах преобразования предлагается ввести следующие обозначения:

— $H_R^{\sim} : R \cup AR \to 2^R \cup 2^{AR}$ — рекурсивная функция доступных ролей с учётом наследования прав доступа, удовлетворяющая для каждой роли $r \in R \cup AR$ условию: если значение функции $role\_inherit(r) = \texttt{true}$, то $H_R^{\sim}(r) = \{r\} \cup \bigcup\limits_{r' \in H_R(r)} H_R^{\sim}(r')$, иначе $H_R^{\sim}(r) = \{r\}$;

— $PA^{\sim} : R \cup AR \to 2^P$ — рекурсивная функция прав доступа к сущностям ролей и административных ролей с учётом иерархии, задающая множество всех доступных роли прав доступа, удовлетворяющая для $r \in R \cup AR$ условию: если значение функции $role\_inherit(r) = \texttt{true}$, то $PA^{\sim}(r) = PA(r) \cup \bigcup\limits_{r' \in H_R(r)} PA^{\sim}(r')$, иначе $PA^{\sim}(r) = PA(r)$.

Функция $PA^{\sim}$ позволяет получить всё множество доступных заданной роли прав доступа.

По аналогии с МРОСЛ ДП-моделью, для краткости и удобства описания элементов модели с учётом мандатного управления доступом вводится функция доступа субъект-сессии к сущности в контейнере с учётом прав доступа к сущностям-контейнерам, её содержащим. Но, в отличие от ОС, в СУБД у контейнеров отсутствует право доступа *execute*. Таким образом, задана следующая функция:

— $lookup : S \times E \setminus O_d \to \{\texttt{true}, \texttt{false}\}$ — функция доступа субъект-сессии к сущности в контейнере с учётом прав доступа к сущностям-контейнерам, её содержащим, такая, что по определению для субъект-сессии $s \in S$ и сущности $e \in E \setminus O_d$ справедливо равенство $lookup(s, e) = \texttt{true}$ тогда и только тогда, когда $e \in C_{cl}$ (доступ субъект-сессии к кластеру по определению есть) или если $e \in E \setminus O_d$ и существует последовательность сущностей $e_0, \ldots, e_n \in E$, где $n \geqslant 1$, $e_0 \in C_{cl}$, $e = e_n$, удовлетворяющих следующим условиям:
  — $e_i \in H_E(e_{i-1})$, где $1 \leqslant i \leqslant n$;
  — существует $r_i \in R \cup AR$, такая, что $(s, r_i, read_a) \in AA$ и
  если $e_i \in C_{db}$, то $(e_i, connect_r) \in PA\ (r_i)$,
  если $e_i \in C_{nsp}$, то $(e_i, usage_r) \in PA\ (r_i)$,
  для других типов контейнеров дополнительных условий не требуется;
  — $f_e(e_i) \leqslant f_s(s)$, или $CCR(e_i) = \texttt{false}$, или $downgrade\_admin\_sop \in session\_option(x)$.

Объекты-данные O_d исключаются, так как доступ к ним регламентируется отдельными правами доступа и осуществляется только через другие объекты.

Определим $G = (UA, AUA, PA, user, role, FC, CCR, A, AA, F, H_R, H_E, L_U, L_S)$ — состояние системы; используем обозначения: $\sum(G^*, OP)$ — система, при этом:

— G^* — множество всех возможных состояний;
— OP — множество правил преобразования состояний;
— $G \vdash_{op} G'$ — переход системы $\sum(G^*, OP)$ из состояния G в состояние G' с использованием правила преобразования состояний $op \in OP$.

В начальном состоянии $G_0 = (UA_0, AUA_0, PA_0, user_0, role_0, FC_0, CCR_0, A_0, AA_0, F_0, H_{R_0}, H_{E_0}, L_{U_0}, L_{S_0})$ по определению справедливо $A_0 = \varnothing$, $AA_0 = \varnothing$, $F_0 = \varnothing$.

3. Правила преобразования состояний

Далее приведено описание основных правил преобразования состояний системы, заданных в рамках PostgreSQL ДП-модели, в которых по существу используются её новые элементы. Кроме того, на реализации именно этих правил сконцентрированы усилия автора при внедрении модели в рассматриваемую СУБД ОС СН.

Для упрощения описания правил преобразования будем записывать только те составляющие состояния G', которые меняются относительно состояния G.

Наличие права $superuser_r$ разрешает любые действия и для упрощения без необходимости не указывается.

Де-юре правила преобразования состояний PostgreSQL ДП-модели

Правило	Исходное состояние G	Результирующее состояние G'
1	2	3
$access\_read(x, y)$	$x \in S$, $y \in (E \setminus O_d) \cup R \cup AR$, [$y \in E \setminus O_d$, (либо $lookup(x, y) =$ `true` и $f_e(y) \leqslant f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)], [$y \in R \cup AR$, (либо $f_r(y) \leqslant f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)]	если $y \in E \setminus O_d$, то $A' = A \cup \{(x, y, read_a)\}, AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, read_a)\}, A' = A$
$access\_write(x, y)$	$x \in S$, $y \in (E \setminus O_d) \cup R \cup AR$, [$y \in E \setminus O_d$, (если $y \in C_{cl}$, то $(y, superuser_r) \in PA^{\sim}(role(x))$, если $y \in C_{db} \cup C_{nsp}$, то $(y, create_r) \in PA^{\sim}(role(x))$), (либо $lookup(x, y) =$ `true` и ($f_e(y) = f_s(x)$ или ($f_s(x) < f_e(y)$ и $CCR(y) =$ `false`)), либо $downgrade\_admin\_sop \in session\_option(x)$)], [$y \in R \cup AR$, $(y, createrole_r) \in PA^{\sim}(role(x))$, (либо $f_r(y) = f_s(x)$, либо $downgrade\_admin\_sop \in session\_option(x)$)]	если $y \in E \setminus O_d$, то $A' = A \cup \{(x, y, write_a)\}, AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, write_a)\}, A' = A$
$access\_append(x, y)$	$x \in S$, $y \in (E \setminus O_d)$, (если $y \in C_{cl}$, то $(y, superuser_r) \in PA^{\sim}(role(x))$, если $y \in C_{db} \cup C_{nsp}$, то $(y, create_r) \in PA^{\sim}(role(x))$], (либо $f_s(x) < f_e(y)$, либо $downgrade\_admin\_sop \in session\_option(x)$))	$A' = A \cup \{(x, y, read_a)\}, AA' = AA$
$delete\_access(x, y, \alpha_a)$	$x \in S, y \in E \cup R \cup AR$, $(x, y, \alpha_a) \in A \cup AA$	если $y \in E$, то $A' = A \setminus (x, y, \alpha_a), AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \setminus (x, y, \alpha_a), A' = A$
$access\_select(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{out})$	$x \in S$, $y \in C_{rel}$, $(x, y, read_a) \in A$, $\{col_j : 1 \leqslant j \leqslant k\} \in O_{col} \cap H_E(y)$, $D_{out} \subseteq O_d \cap H_E(y)$, (либо $(y, select_r) \in PA^{\sim}(role(x))$, либо $(col_j, select_r) \in PA^{\sim}(role(x))$, $1 \leqslant j \leqslant k$), (либо $\forall t \in D_{out}(f_e(t) \leqslant f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$A' = A \cup \{(x, t, read_a) : t \in D_{out}\}$, $\forall er \in rules(y, select_r)$ выполняется $execute\_rule(er)$

Продолжение таблицы

1	2	3
$access\_insert(x, y, D_{in})$	$x \in S, y \in C_{rel}, (x, y, append_a) \in A$, $D_{in} \not\subseteq O$, (либо $(y, insert_r) \in PA^{\sim}(role(x))$, либо $(c, insert_r) \in PA^{\sim}(role(x))$, $c \in \{c : c \in H_E(y), type(c) = col\}$)	$A' = A \cup \{(x, t, append_a) : t \in D_{in}\}$, $\forall t \in D_{in}\ f'_e(t) = f_s(x)$, $O'_d = O_d \cup D_{in}, H'_E(y) = H_E(y) \cup D_{in}$, $\forall er \in rules(y, insert_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, insert_r)$ выполняется $execute\_procedure(trigger\_proc(et))$
$access\_update(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{mod})$	$x \in S$, $y \in C_{rel}$, $(x, y, write_a) \in A$, $\{col_j : 1 \leqslant j \leqslant k\} \in O_{col} \cap H_E(y)$, $D_{mod} \subseteq O_d \cap H_E(y)$, (либо $(y, update_r) \in PA^{\sim}(role(x))$, либо $(col_j, update_r) \in PA^{\sim}(role(x))$, $1 \leqslant j \leqslant k$), (либо $\forall t \in D_{mod}(f_e(t) = f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$A' = A \cup \{(x, t, write_a) : t \in D_{mod}\}$, $\forall er \in rules(y, update_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, update_r)$ выполняется $execute\_procedure(trigger\_proc(et))$
$access\_delete(x, y, D_{del})$	$x \in S$, $y \in C_{rel}$, $(x, y, write_a) \in A$, $D_{del} \subseteq O_d \cap H_E(y)$, (либо $(y, delete_r) \in PA^{\sim}(role(x))$, либо $(c, delete_r) \in PA^{\sim}(role(x))$, $c \in \{c : c \in H_E(y), type(c) = col\}$), (либо $\forall t \in D_{del}(f_e(t) = f_s(x))$, либо $downgrade\_admin\_sop \in session\_option(x)$)	$O'_d = O_d \setminus D_{del}$, $H'_E(y) = H_E(y) \setminus D_{del}$, $\forall er \in rules(y, delete_r)$ выполняется $execute\_rule(er)$, $\forall et \in triggers(y, delete_r)$ выполняется $execute\_procedure(trigger\_proc(et))$, $A' = A \setminus \{(s, t, \alpha_a) : s \in S, t \in D_{del}, \alpha_a \in R_a\}$
$create\_rel(x, y, z, Q_{col}, Q_{cnstr})$	$x \in S, y \notin E, z \in C_{nsp}, type(y) = rel$, $\forall e \in Q_\alpha\ type(e) = \alpha$ и $e \notin E$, $(x, z, append_a) \in A$, $\forall e' \in Q_{col}\ (x, coltype(e'), read_a) \in A$	$E' = E \cup \{y\} \cup Q_{col} \cup Q_{cnstr}$, $(C' = C \cup \{y\}, O' = O \cup Q_{col} \cup Q_{cnstr})$, $H'_E(z) = H_E(z) \cup \{y\}$, $H'_E(y) = Q_{col} \cup Q_{cnstr}$, $f'_e(y) = f_s(x), CCR'(y) = \texttt{true}$, $PA'(role(x)) = PA(role(x)) \cup ALL_r(y)$, $Gr'(role(x)) = Gr(role(x)) \cup ALL_{r(y)}$
$create\_procedure(x, y, z, as\_option, \{op_j : 1 \leqslant j \leqslant k\})$	$x \in S, y \notin E, z \in C_{nsp}$, $as\_option \in \{as\_caller, as\_owner\}$, $\{op_j : 1 \leqslant j \leqslant k\} \subseteq OP$, $(x, z, append_a) \in A$	$O' = O \cup \{y\}$, $(O'_{proc} = O_{proc} \cup \{y\}$, $C' = C)$, $H'_E(z) = H_E(z) \cup \{y\}$, $operations'(y) = \{op_j : 1 \leqslant j \leqslant k\}$, $f'_e(y) = f_s(x)$, $execute\_as'(y) = as\_option$, $PA'(role(x)) = PA(role(x)) \cup ALL_r(y)$, $Gr'(role(x)) = Gr(role(x)) \cup ALL_r(y)$, $PA'(public) = PA(public) \cup \{(y, execute_r)\}$
$execute\_procedure(x, y)$	$x \in S, y \in O_{proc}, (x, y, read_a) \in A$, $(y, execute_r) \in PA^{\sim}(role(x))$	если $execute\_as(y) = as\_owner$, то: 1) $\exists r \in R \cup AR$ $(y, own_r) \in PA(r)\ role'(x) = r$; 2) выполняется $G \vdash_{op_1} \ldots \vdash_{op_n} G_n = G'$, $op_i \in operations(x), i = 1, \ldots, n$; 3) $role'(x) = role(x)$, иначе выполняется $G \vdash_{op_1} \ldots \vdash_{op_n} G_n = G'$, $op_i \in operations(x), 1 \leqslant i \leqslant n$

Окончание таблицы

1	2	3
$create\_trigger(x, y, z, p, \{r_j : 1 \leqslant j \leqslant k\})$	$x \in S$, $y \notin E$, $z \in C_{rel}$, $p \in O_{proc}$, $\{r_j : 1 \leqslant j \leqslant k\} \subseteq \{insert_r, update_r, delete_r, truncate_r\}$, $(x, z, write_a) \in A, (x, p, read_a) \in A$, $(z, trigger_r) \in PA^{\sim}(role(x))$	$O' = O \cup \{y\}, (O'_{trg} = O_{trg} \cup \{y\}$, $C' = C), H'_E(z) = H_E(z) \cup \{y\}$, $f'_e(y) = f_s(x), trigger\_proc'(y) = p$, $triggers'(z, r_j) = triggers(z, r_j) \cup \{y\}$, $1 \leqslant j \leqslant k$
$create\_rule(x, y, z, op, r)$	$x \in S$, $y \notin E$, $z \in C_{rel}$, $op \in \{access\_select, access\_insert, access\_update, access\_delete\}$, $r \in \{select_r, insert_r, update_r, delete_r\}$, $(x, z, write_a) \in A$, $(z, own_r) \in PA^{\sim}(role(x))$	$O' = O \cup \{y\}, (O'_{rule} = O_{rule} \cup \{y\}$, $C' = C), H'_E(z) = H_E(z) \cup \{y\}$, $f'_e(y) = f_s(x), rule\_op'(y) = op$, $rules'(z, r) = rules(z, r) \cup \{y\}$
$execute\_rule(x, y, z, r)$	$x \in S$, $y \in O_{rule}$, $y \in H_E(z)$, $r \in \{select_r, insert_r, update_r, delete_r\}$, $(x, z, \alpha) \in A$, ($\alpha = read_a$ для $r = select_r$, $\alpha = append_a$ для $r = insert_r$, $\alpha \in \{read_a, write_a\}$ для $r \in \{update_r, delete_r\}$)	выполняется $G \vdash_{op(y)} G_{op(y)} = G'$
$grant\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$	$x \in S$, $y \in E \setminus (O_d \cup C_{cl})$, $r \in R \cup AR$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq ALL_r(y)$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq Gr(role(x))$, $grant\_option \in \{\texttt{true}, \texttt{false}\}$, $(x, r, write_a) \in AA$	$PA'(r) = PA(r) \cup \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$, (если $r \neq public$ и $grant\_option = \texttt{true}$, то $Gr'(r) = Gr(r) \cup \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$)
$revoke\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$	$x \in S$, $y \in E \setminus (O_d \cup C_{cl})$, $r \in R \cup AR$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq ALL_r(y)$, $\{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\} \subseteq Gr(role(x))$, $grant\_option \in \{\texttt{true}, \texttt{false}\}$, $(x, r, write_a) \in AA$	[если $r \neq public$, то $Gr'(r) = Gr(r) \setminus \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\})$], [если $r = public$ или $grant\_option = \texttt{false}$, то $PA'(r) = PA(r) \setminus \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}$]

Де-юре правила $access\_read(x, y)$, $access\_write(x, y)$ и $access\_append(x, y)$ аналогичны одноименным правилам МРОСЛ ДП-модели и позволяют субъект-сессии x получить соответствующий доступ к сущности y с определёнными условиями на уровень конфиденциальности субъект-сессии. Отличие заключается в замене функции $execute\_container(x, y)$ на функцию $lookup(x, y)$. Кроме того, при доступе к кластеру необходимо наличие у роли субъект-сессии доступного по иерархии права $superuser_r$, а при доступе к базам данных и схемам — наличие права $create_r$ к ним. Для нарушения требований к уровню конфиденциальности субъект-сессии требуется наличие у субъект-сессии параметра $downgrade\_admin\_sop$. Де-юре правило $delete\_access(x, y, \alpha_a)$ позволяет субъект-сессии x, обладающей доступом α_a к сущности, роли или административной роли y, удалить этот доступ.

Де-юре правила $access\_select(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{out})$, $access\_insert(x, y, D_{in})$, $access\_update(x, y, \{col_j : 1 \leqslant j \leqslant k\}, D_{mod})$ и $access\_delete(x, y, D_{del})$ отражают специфику работы с данными в таблицах БД и позволяют субъект-сессии x получить доступ на выборку, вставку, изменение и удаление данных D сущности-таблицы y. Для выполнения операции необходимо наличие у роли субъект-сессии x доступных по иерархии необходимых доступов к таблице и соответствующего операции права доступа. Операции затрагивают только доступные по мандатным атрибутам записи таблицы (для нарушения указанного поведения требуется наличие у субъект-сессии параметра $downgrade\_admin\_sop$). При выполнении операции вызываются на исполнение соответствующие реализованному праву доступа триггеры и правила, заданные для указанной таблицы функциями $triggers$ и $rules$ соответственно.

Правила создания контейнеров рассмотрим на примере де-юре правила создания таблицы $create\_rel(x, y, z, Q_{col}, Q_{cntsr})$, которое позволяет субъект-сессии СУБД x создать новую контейнер-таблицу y в схеме z с указанным множеством столбцов Q_{col} и ограничений Q_{cnstr}. Для этого требуется наличие у субъект-сессии x доступа на добавление к схеме z и на чтение к сущностям-типам, указанным для столбцов из Q_{col}. При этом новая сущность-контейнер приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, и мандатный атрибут CCR, равный `true`. Роль субъект-сессии получает все права доступа к новой сущности с опцией их делегирования (множество Gr).

Де-юре правило $create\_procedure(x, y, z, as\_option, \{op_j : 1 \leqslant j \leqslant k\})$ позволяет субъект-сессии x создать процедуру y в контейнере $z \in C_{nsp}$ при наличии к нему доступа на добавление. Для создаваемой процедуры указывается режим выполнения кода $as\_option \in \{as\_caller, as\_owner\}$ и упорядоченный конечный набор операций $\{op_j : 1 \leqslant j \leqslant k\}$, входящих в множество OP правил преобразования состояния модели. При этом новая сущность приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, указанный режим исполнения кода, а в качестве кода — состав указанных операций. Роль субъект-сессии получает все права доступа к новой сущности с опцией их делегирования (множество Gr). Для процедур по умолчанию предоставляется право на исполнение $execute_r$ группе $PUBLIC$.

Де-юре правило $execute\_procedure(x, y)$ позволяет субъект-сессии x выполнить процедуру y при наличии к ней доступа на чтение и доступного по иерархии права исполнения $execute_r$ для указанной процедуры у роли субъект-сессии. В ходе исполнения процедуры согласно упорядоченному набору операций, составляющих тело процедуры $operations$, последовательно выполняются преобразования состояния модели. При этом если для процедуры указан режим исполнения кода as_owner, перед выполнением процедуры текущая роль субъект-сессии меняется на роль, обладающую правом владения к процедуре; после исполнения исходная роль субъект-сессии восстанавливается.

Де-юре правила $create\_trigger(x, y, z, p, \{r_j : 1 \leqslant j \leqslant k\})$ и $create\_rule(x, y, z, op, r)$ позволяют субъект-сессии x создать для таблицы $z \in C_{rel}$ новый триггер или правило, выполняющиеся при реализации прав доступа $\{r_j : 1 \leqslant j \leqslant k\} \subseteq \{insert_r, update_r, delete_r, truncate_r\}$ или $r \in \{select_r, insert_r, update_r, delete_r\}$ соответственно. При создании триггера указывается активируемая процедура $p \in O_{proc}$, причём необходимо наличие у субъект-сессии z доступов на запись к таблице z и чтение к процедуре p и доступного по иерархии права создания триггера $trigger_r$ для указанной таблицы у роли субъект-сессии. При создании правила указывается операция op из перечисленных $\{access\_select, access\_insert, access\_update, access\_delete\} \in OP$ правил преобразования состояния модели, причём необходимо наличие у субъект-сессии x доступного по иерархии права владения к таблице z. Новая сущность приобретает уровень конфиденциальности, равный уровню доступа субъект-сессии x, помещается в иерархию таблицы z, обновляются функции $triggers$ и $rules$ соответственно; устанавливается связь между триггером и активируемой процедурой или правилом таблицы и правилом преобразования состояния модели.

Де-юре правило $execute\_rule(x, y, z, r)$ позволяет субъект-сессии x выполнить правило y таблицы z при реализации права доступа r. Данное правило вызывается автоматически при реализации заданного права доступа к таблице. Для этого необходимо наличие соответствующего доступа к таблице. При выполнении этого правила реально исполняется указанное при создании правило op преобразования состояния модели.

Де-юре правила $grant\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$ и $revoke\_rights(x, r, \{(y, \alpha_{rj}) : 1 \leqslant j \leqslant k\}, grant\_option)$ позволяют субъект-сессии x добавить или удалить соответственно права доступа к сущности y из множества прав доступа роли или административной роли r. Для применения правил необходимо наличие у субъект-сессии x доступа на запись к роли r и перечисленных прав доступа в списке делегирования Gr роли субъект-сессии. Параметр $grant\_option$ соответствует опции *SQL WITH_ GRANT_ OPTION*, предоставляющей роли r возможность делегировать полученные права. Данная опция не может быть применена к встроенной группе *PUBLIC*. При удалении, если параметр $grant\_option$ установлен в `true`, отбирается только возможность делегирования.

Де-факто правила, используемые для отражения факта получения субъект-сессией де-факто владения субъект-сессиями или факта реализации информационного потока по памяти или по времени, схожи с аналогичными правилами МРОСЛ ДП-модели.

Заключение

Основной целью адаптации МРОСЛ ДП-модели для использования в реляционной СУБД PostgreSQL и разработки новой PostgreSQL ДП-модели является обеспечение единого подхода к управлению доступом в СУБД и ОС СН Astra Linux Special Edition.

Проводимое исследование еще не закончено, однако уже в настоящее время в рамках PostgreSQL ДП-модели удалось описать специфичные для исследуемой СУБД элементы, отличные от заданных в известных автору моделях управления доступом в ОС или СУБД. При этом наибольшее внимание уделено учёту специфики реляционных баз данных и сохранению совместимости с традиционными требованиями стандарта SQL.

Создание модели позволит не только сформулировать и обосновать условия нарушения безопасности в СУБД, но и проверить их корректность в реальной системе. В дальнейшем интеграция СУБД с реализацией PostgreSQL ДП-модели в разрабатываемую версию ОС СН позволит повысить защищённость информации в автоматизированных системах, создаваемых на их основе.

ЛИТЕРАТУРА

1. *Гостехкомиссия России.* Руководящий документ. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищённости от несанкционированного доступа к информации. М.: Военное издательство, 1992.
2. *Гостехкомиссия России.* Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. М.: Военное издательство, 1992.
3. *Девянин П. Н.* О разработке мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в операционных системах семейства Linux // Методы и технические средства обеспечения безопасности информации: материалы 21-й науч.-технич. конф. 24–29 июня 2012 г. СПб.: Изд-во Политехн. ун-та, 2012. С. 91–94.
4. *Девянин П. Н.* Об опыте внедрения мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в защищенную ОС Astra Linux Special Edition // Методы и технические средства обеспечения безопасности информации: материалы 22-й науч.-технич. конф. 08–11 августа 2013 г. СПб.: Изд-во Политехн. ун-та, 2013. С. 78–80.
5. Операционные системы Astra Linux [Электронный ресурс]. `http://www.astra-linux.ru/`.
6. *Sandhu R.* Rationale for the RBAC96 family of access control models // Proc. 1st ACM Workshop on Role-Based Access Control. ACM, 1997.

7. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
8. *Колегов Д. Н.* О построении иерархического ролевого управления доступом // Прикладная дискретная математика. 2012. № 5. С. 69–71.
9. *Смольянинов В. Ю.* О достаточных условиях похищения прав доступа в СУБД ДП-модели // Прикладная дискретная математика. 2012. № 5. С. 75–76.
10. *Смирнов С. Н.* Безопасность систем баз данных. М.: Гелиос АРВ, 2007. 352 с.
11. PostgreSQL 9.2.1 Documentation. The PostgreSQL Global Development Group, 2012. 2605 p.
12. Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework). ISO/IEC 9075:1999, 1999.
13. Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation). ISO/IEC 9075:1999, 1999.
14. *Sumathi S. and Esakkirajan S.* Fundamentals of Relational Database Management Systems. Springer, 2007. 776 p.
15. *Lane T.* A Tour of PostgreSQL Internals. Great Bridge, LLC, 2000. 25 p. [Электронный ресурс.] `http://www.postgresql.org/files/developer/tour.pdf`.
16. *Шумилин А. В.* Перспективный подход к обеспечению защиты информации от несанкционированного доступа в СУБД // Новые технологии. Программная инженерия. 2012. № 1. С. 35–40.
17. *Шумилин А. В.* Подход к оцениванию влияния средств разграничения доступа к данным на производительность реляционных СУБД // Новые технологии. Программная инженерия. 2013. № 4. С. 29–33.
18. *Шумилин А. В.* Применение мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в СУБД операционной системы специального назначения Astra Linux Special Edition // Методы и технические средства обеспечения безопасности информации: материалы 22-й науч.-технич. конф. 08–11 августа 2013 г. СПб.: Изд-во Политехн. ун-та, 2013. С. 87–89.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.17

ХАРАКТЕРИЗАЦИЯ ОРГРАФОВ С ТРЕМЯ ДОПОЛНИТЕЛЬНЫМИ ДУГАМИ В МИНИМАЛЬНОМ ВЕРШИННОМ 1-РАСШИРЕНИИ

М. Б. Абросимов, О. В. Моденова

Саратовский государственный университет им. Н. Г. Чернышевского, г. Саратов, Россия

E-mail: mic@rambler.ru

Описаны все орграфы, минимальное вершинное 1-расширение которых имеет в точности три дополнительных ребра.

Ключевые слова: *граф, минимальное вершинное 1-расширение, точное вершинное 1-расширение, оптимальная отказоустойчивая реализация.*

Введение

Ориентированным графом (далее — *орграфом*) называется пара $\overrightarrow{G} = (V, \alpha)$, где V — конечное непустое множество, называемое множеством вершин, а α — отношение на множестве вершин V, называемое отношением смежности. Граф с симметричным и антирефлексивным отношением смежности называется *неориентированным графом*. Далее используются основные понятия преимущественно в соответствии с работой [1].

Симметризацией орграфа $\overrightarrow{G} = (V, \alpha)$ называется неорграф $G = (V, (\alpha \cup \alpha^{-1})\backslash\Delta)$, то есть симметризация орграфа получается заменой дуг рёбрами и удалением петель.

Граф $G^* = (V^*, \alpha^*)$ называется *минимальным вершинным k-расширением* (МВ-kР) n-вершинного графа $G = (V, \alpha)$, если выполняются следующие условия:

1) граф G^* является вершинным k-расширением G, то есть граф G вложим в каждый подграф графа G^*, получающийся удалением любых его k вершин;

2) граф G^* содержит $n + k$ вершин, то есть $|V^*| = |V| + k$;

3) α^* имеет минимальную мощность при выполнении условий 1 и 2.

Построение МВ-kР можно представить как добавление к исходному графу k дополнительных вершин и некоторого числа дополнительных рёбер. Это число дополнительных рёбер иногда называют *рёберной стоимостью* (edge cost) и обозначают $\mathrm{ec}(G, k)$.

Граф G^* называется *точным вершинным k-расширением* (ТВ-kР) графа G, если граф G изоморфен каждому подграфу графа G^*, получающемуся из него удалением любых его k вершин и всех связанных с ними дуг (рёбер).

В данной работе рассматривается случай $k = 1$.

Понятие МВ-kР введено на основе понятия оптимальной k-отказоустойчивой реализации, которое предложено Хейзом в работе [2]. Оказалось, что задача является вычислительно сложной [3]. Исследования данной проблемы развивались в двух направлениях. Основное направление — поиск минимальных расширений для интересных классов графов: цепей, циклов, деревьев. Второе — описание графов, минимальные расширения которых имеют заданное число дополнительных дуг или рёбер. В работе [4] исследована задача описания неориентированных графов с $\mathrm{ec}(G, 1) \leqslant 3$. Получены следующие результаты (теоремы 1–5).

Теорема 1. МВ-kР, причём единственное с точностью до изоморфизма, вполне несвязного n-вершинного графа O_n есть вполне несвязный $(n + k)$-вершинный граф O_{n+k}. Никакие другие графы не могут иметь МВ-kР с нулевым числом дополнительных рёбер.

Теорема 2. Графы со степенным множеством $\{1, 0\}$, и только они, имеют МВ-1Р с одним дополнительным ребром, причём это расширение единственно с точностью до изоморфизма.

Теорема 3. Среди связных графов только цепи имеют МВ-1Р с двумя дополнительными рёбрами, причём это расширение единственно с точностью до изоморфизма.

Теорема 4. Среди несвязных графов без изолированных вершин только графы вида $P_n \cup C_{n+1} \cup \ldots \cup C_{n+1}$ при $n > 1$ имеют МВ-1Р с двумя дополнительными рёбрами, причём это расширение имеет вид $C_{n+1} \cup C_{n+1} \cup \ldots \cup C_{n+1}$, и оно единственно с точностью до изоморфизма.

Теорема 5. Связные графы, имеющие МВ-1Р с тремя дополнительными рёбрами, могут иметь только следующий вид:

1) полный граф K_3;
2) графы с вектором степеней вида $(3, \ldots, 3, 2, 2, 2)$, имеющие ТВ-1Р;
3) графы с вектором степеней $(3, 3, 3, \ldots, 3, 2, \ldots, 2, 1)$ особого вида.

В данной работе изучается аналогичный вопрос для ориентированных графов. В [5] показана связь между МВ-1Р неориентированных и ориентированных графов.

Лемма 1. Пусть орграф G^* есть МВ-kР орграфа G. Тогда симметризация орграфа G^* является вершинным k-расширением симметризации орграфа G.

Следствие. Число дополнительных дуг МВ-kР орграфа G не меньше числа дополнительных рёбер МВ-kР симметризации орграфа G.

В работе [6] удалось полностью описать орграфы с $\mathrm{ec}(G, 1) \leqslant 2$. Далее рассмотрим случай $\mathrm{ec}(G, 1) = 3$.

Основные результаты

Рассмотрим только связные орграфы без петель.

Лемма 2. В симметризации МВ-1Р ориентаций цепей вершины степени 3 не могут быть смежны с тремя вершинами степени 2.

Доказательство. Допустим, от противного, что есть МВ-1Р некоторой ориентации цепи и его симметризация имеет вершину степени 3, смежную с тремя вершинами степени 2. Тогда, удалив эту вершину степени 3, получим три вершины степени 1. Очевидно, что цепь сюда вложить нельзя. ■

Теорема 6. Не существует ориентаций цепей с числом вершин $n > 4$, таких, что их МВ-1Р имеет три дополнительных дуги.

Доказательство. Рассматривать гамильтоновы цепи здесь не будем, потому что выше доказано, что такие цепи имеют МВ-1Р с двумя дополнительными дугами.

В симметризации МВ-1Р цепи не может быть вершин со степенью меньше 2. По условию теоремы, МВ-1Р должно иметь три дополнительные дуги. Построить МВ-1Р с тремя дополнительными дугами можно по одной из схем на рис. 1 (на иллюстрации ориентации дуг не указаны).

По лемме 2 схемы d и e не являются МВ-1Р цепи.

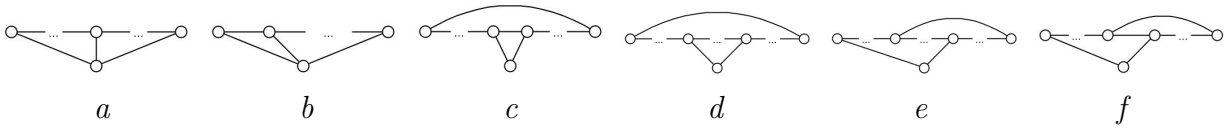

Рис. 1. Схемы построения МВ-1Р для цепи

Будем считать, что в цепи 7 или более вершин. Для 5- и 6-вершинных цепей непосредственной проверкой можно убедиться, что для них МВ-1Р с тремя дополнительными дугами не существует.

Рассмотрим первую схему. Выберем две вершины в этой схеме (v_1 и v_2, рис. 2), суммы полустепеней исхода и захода у которых равны 2.

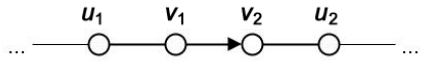

Рис. 2. Схема a, дуга из v_1 в v_2

Пусть дуга между v_1 и v_2 направлена так, как изображено на рис. 2. Ориентации остальных дуг не имеют значения. Тогда при удалении вершины u_1 получим, что один конец цепи имеет полустепени (1,0). А если удалим u_2, то получим, что второй конец цепи имеет полустепень (0,1). Аналогично и для других схем.

Рассмотрим схему a. Пусть концы этой цепи имеют следующие полустепени: u_1 — (1,0), u_n — (0,1). Рассмотрим дугу между u_2 и u_3. Предположим, что она идёт из u_2 в u_3. Удалим вершину u_{n-2}. Тогда u_{n-1} становится концом (1,0) и, следовательно, дуга из u_n направлена в u_{n+1} (рис. 3).

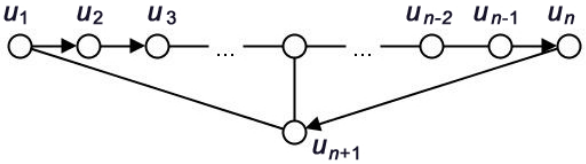

Рис. 3. Схема a, дуга из u_n в u_{n+1}

И так далее — с помощью удаления вершин определяем, как должны быть ориентированы дуги цепи. В итоге получим, что цепь гамильтонова. Гамильтоновы цепи имеют МВ-1Р с двумя дополнительными дугами, поэтому данная схема не подходит.

Теперь предположим, что дуга идет из u_3 в u_2 (рис. 4). Тогда в цепи вторая вершина — сток. Удалим u_{n-3}; вершина u_{n-2} теперь не может быть концом (1,0), потому что с ней должна быть смежна вершина-сток, а из u_{n-1} уже выходит одна дуга. Значит, вершина u_{n-2} после удаления u_{n-3} имеет полустепени (0,1).

Удалим вершину u_{n-2}. Тогда u_{n-1} станет концом (1,0), а u_n должна быть стоком, то есть дуга направлена из u_{n+1} в u_n. И так далее, удаляя вершины, будем определять направления дуг. В результате получим, что все вершины цепи — источники или стоки. Отсюда следует, что в цепи чётное число вершин, потому что концы цепи разные.

Определим направление дуги между u_{n+1} и u_1. Для этого удалим, например, u_{n-1}. Тогда получаем, что дуга направлена из u_{n+1} в u_1.

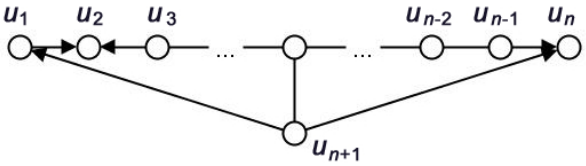

Рис. 4. Схема a, дуга из u_{n+1} в u_1

В данном случае цепь будет строиться как $u_n - u_{n+1} - u_1 - u_2 -$ и т. д., но в такой цепи вершина u_1 не сток и не источник. Получили противоречие. Следовательно, данная схема не подходит.

Р а с с м о т р и м с х е м у b. Пусть концы этой цепи имеют следующие полустепени: u_1 — (1,0), u_n — (0,1). Рассмотрим дугу между u_2 и u_3. Предположим, что она идёт из u_2 в u_3. Удалим вершину u_{n-2}. Тогда u_{n-1} становится концом (1,0) и, следовательно, дуга из u_n направлена в u_{n+1} (рис. 5).

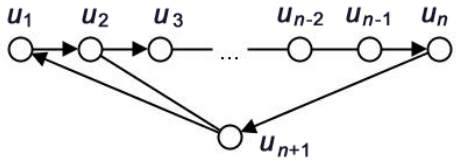

Рис. 5. Схема b, дуга из u_n в u_{n+1}

И так далее — с помощью удаления вершин определяем, как должны быть ориентированы дуги цепи. И в итоге получим, что цепь гамильтонова. Гамильтоновы цепи имеют МВ-1Р с двумя дополнительными дугами, поэтому данная схема не подходит.

Теперь предположим, что дуга идет из u_3 в u_2 (рис. 6). Тогда в цепи вторая вершина — сток. Удалим u_{n-3}. Вершина u_{n-2} теперь не может быть концом (1,0), потому что с ней должна быть смежна вершина-сток, а из u_{n-1} уже выходит одна дуга. Значит, вершина u_{n-2} после удаления u_{n-3} имеет полустепени (0,1).

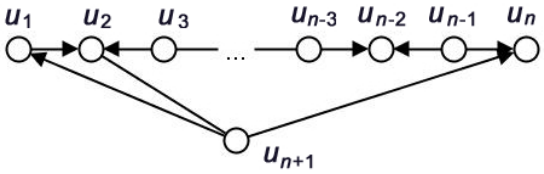

Рис. 6. Схема b, дуга из u_{n+1} в u_1

Теперь удалим вершину u_n, и так далее. В результате получим, что цепь должна состоять только из стоков и источников. У цепи концы разные, поэтому вершин в ней должно быть чётное количество.

Удалим вершину u_{n-2}. Тогда u_{n-1} станет концом (1,0), а u_n должна быть стоком, т. е. дуга направлена из u_{n+1} в u_n. Удалим u_{n-1} и получим направление ещё одной дуги — дуга из u_{n+1} направлена в u_1.

В данном случае цепь будет строиться как $u_n - u_{n+1} - u_1 - u_2 -$ и т. д., но в такой цепи вершина u_1 не сток и не источник. Получили противоречие. Следовательно, данная схема не подходит.

Рассмотрим схему c. Пусть концы этой цепи имеют следующие полустепени: $u_1 - (1,0)$, $u_n - (0,1)$. Удалим первую вершину u_1. Второй конец цепи останется прежним, значит, ребро $\{u_2, u_3\}$ направляем из u_2 в u_3: (u_2, u_3). И так далее. Получим в итоге, что цепь должна быть гамильтоновой, а гамильтоновы цепи имеют МВ-1Р с двумя дополнительными дугами. Значит, данная схема не подходит.

Рассмотрим схему f. Пусть концы этой цепи имеют следующие полустепени: $u_1 - (1,0)$, $u_n - (0,1)$. Рассмотрим дугу между u_2 и u_3. Предположим, что она идёт из u_2 в u_3 (рис. 7). Если удалить вершину u_{n-2}, то получим направление дуги между u_n и u_i: из u_n в u_i.

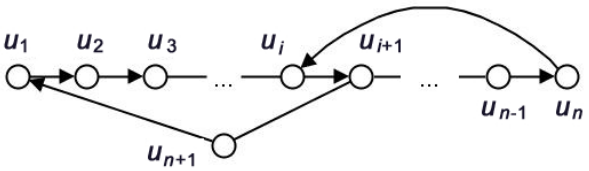

Рис. 7. Схема f, дуга из u_n в u_i

Если удалить вершину u_1, то u_2 станет концом (1,0) и дуга между u_3 и u_4 будет направлена из u_3 в u_4. И так продолжаем до тех пор, пока не дойдём до u_i. Имеем, что первые $i-1$ дуги направлены «в одну сторону». При удалении вершины u_{i-2} цепь будет строиться как $u_{i-1} - u_i - u_n -$ и т. д. Получаем противоречие, потому что дуга направлена из u_n в u_i, а по нашему предположению, она должна быть направлена в обратную сторону.

Теперь предположим, что дуга идет из u_3 в u_2 (рис. 8). Тогда в цепи вторая вершина — сток. Удалим вершину u_{n-2}. Тогда дуга между u_n и u_i должна быть направлена из u_i в u_n.

Удалим u_{n-3}. Тогда вершина u_{n-2} теперь не может быть концом (1,0), потому что с ней должна быть смежна вершина-сток, а из u_{n-1} уже выходит одна дуга. Значит, вершина u_{n-2} после удаления u_{n-3} имеет полустепени (0,1).

И так далее. В результате получаем, что все вершины цепи должны быть стоками или источниками. Вершин в цепи должно быть чётное число, так как концы цепи разные.

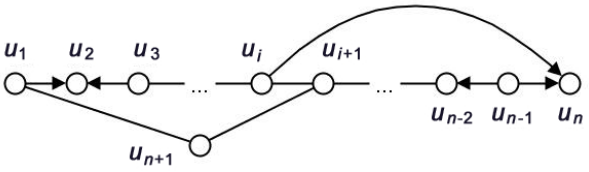

Рис. 8. Схема f, все вершины должны быть стоками или источниками

Рассмотрим дугу из u_i в u_n. При удалении вершины u_{i+1} эта дуга входит в цепь. Вершина u_n — сток, а вершина u_i должна быть источником, т. е. рассматриваемая дуга выходит из нее. Если u_i — сток, то построенный по схеме орграф не является МВ-1Р. Предположим, что u_i — источник. Тогда u_{i+1} — сток. Получается противоречие, если удалить u_i и рассмотреть дугу между u_{i+1} и u_n: обе вершины должны быть стоками, так как в них входят дуги. Но рассматриваемая дуга может иметь только одну ориентацию, поэтому можно сделать вывод, что такая схема тоже не подходит для построения МВ-1Р с тремя дополнительными дугами. ■

В работе [7] доказывается следующее утверждение:

Теорема 7. Если в точном вершинном 1-расширении связного графа есть сток или источник, то этот граф является транзитивным турниром.

Напомним, что граф называется *кубическим*, если все его вершины имеют степень 3. В п. 2 теоремы 5 упоминаются графы с вектором степеней вида $(3,\dots,3,2,2,2)$, имеющие точное вершинное 1-расширение. Эти точные вершинные 1-расширения имеют вектор степеней $(3,\dots,3)$, то есть являются кубическими графами.

Теорема 8. Среди всех ориентаций кубических графов только транзитивный 4-вершинный турнир является точным вершинным 1-расширением.

Доказательство. Пусть G — кубический граф, H — его ориентация, являющаяся точным вершинным 1-расширением. Рассмотрим, какие степени исхода и захода могут иметь вершины графа H: (0,3), (1,2), (2,1) и (3,0). С учётом теоремы 7 получаем, что степени (0,3) и (3,0) могут быть только у ориентации полного графа в транзитивный турнир. Однако из кубических графов только K_4 является полным.

Пусть G не изоморфен K_4. Тогда в H могут быть только вершины со степенями исхода и захода (1,2) и (2,1). Легко видеть, что их число должно быть одинаковым. В самом деле, если бы это было не так и число вершин вида (1,2) было k_1, а вершин вида (2,1) — k_2, то число исходящих дуг было бы $k_1 + 2k_2$, а входящих — $2k_1 + k_2$. Эти числа должны быть равны, то есть $k_1 + 2k_2 = 2k_1 + k_2$, откуда $k_1 = k_2$.

Итак, в графе H одинаковое число вершин вида (1,2) и (2,1). Так как H является точным вершинным 1-расширением, то все его максимальные подграфы изоморфны. Обозначим через u произвольную вершину вида (1,2). Рассмотрим граф, получающийся из H удалением вершины, в которую идет дуга из вершины u. В этом графе вершина u имеет полустепени исхода и захода (0,2). Таким образом, в любом максимальном подграфе графа H должна быть вершина вида (0,2). Но такая вершина может получиться только при удалении вершины, в которую идёт дуга из вершины вида (1,2) графа H. Следовательно, в каждую вершину графа H должна идти дуга из подходящей вершины вида (1,2). Но это невозможно, так как количество вершин графа H в 2 раза больше, чем количество исходящих дуг из вершин вида (1,2). Таким образом, единственная возможная ориентация кубического графа, являющаяся точным вершинным 1-расширением, — транзитивный 4-вершинный турнир. ■

Теорема 9. Среди ориентаций графов из теоремы 5 только граф K_3 имеет такую ориентацию (транзитивный турнир), МВ-1Р которой имеет три дополнительные дуги.

Доказательство. Граф из первого случая теоремы 5 — граф K_3. Известно, что у такого графа есть ориентация — транзитивный турнир, которая имеет МВ-1Р с тремя дополнительными дугами (4-вершинный транзитивный турнир) [8].

Выше доказано, что ориентации графов из второго случая теоремы 5 не имеют МВ-1Р с тремя дополнительными дугами.

Остаётся рассмотреть последний, третий, случай теоремы 5. Это графы с вектором степеней $(3,3,3,\dots,3,2,\dots,2,1)$ особого вида. В работе [4] даётся описание графов такого вида и их МВ-1Р. Рассмотрим граф G такого вида и его МВ-1Р — граф G^*. Граф G^* должен обладать следующими свойствами:

1) степенное множество $\{3,2\}$;
2) количество вершин степени 3 чётно и больше 3;

3) вершины степени 2 смежны либо с двумя вершинами степени 3, либо с одной вершиной степени 3 и одной вершиной степени 2, то есть вершины степени 3 соединяются цепью, состоящей не более чем из трёх ребер;
4) вершины степени 3 смежны с двумя другими вершинами степени 3, причем граф, индуцированный всеми вершинами степени 3, представляет собой цикл.

То есть граф G^* можно представить как цикл, образованный вершинами степени 3, с которыми соединены цепи длины 1 или 2 (рис. 9).

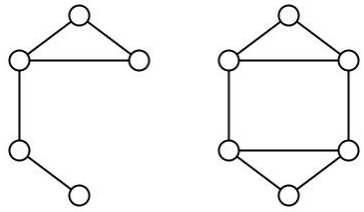

Рис. 9. Пример графа G и его МВ-1Р G^*

При удалении вершины степени 3 из графа G^* получим в точности исходный граф G. Предположим, что существует ориентация графа G^*, такая, что она является МВ-1Р подходящего графа G с тремя дополнительными дугами.

Рассмотрим случай, когда граф G^* представляет собой цикл, в которому присоединены цепи длины 1 (рис. 10).

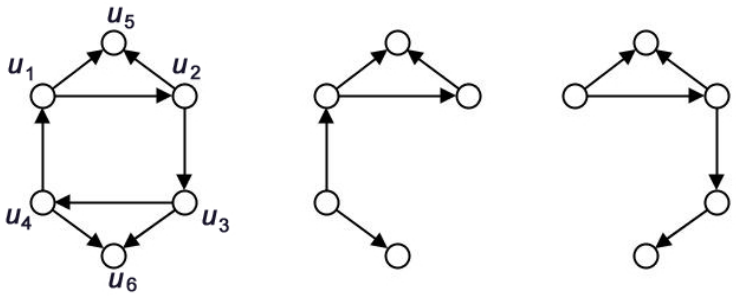

Рис. 10. Пример графа G^* и подграфы, получающиеся при удалении вершин степени 3

Дуги, инцидентные вершинам u_5 и u_6, должны быть ориентированы одинаковым образом. Предположим, что вершины u_5 и u_6 — стоки. Цикл может быть ориентирован как гамильтонов либо как негамильтонов. Если цикл гамильтонов, то при удалении u_1 и при удалении u_2 получим неизоморфные графы, что показано на рис. 10.

Если цикл не гамильтонов, то логично предположить, что он должен быть ориентирован таким образом, чтобы удаление u_1 и удаление u_2 давали изоморфные графы. Но по схеме построения МВ-1Р цикл состоит из вершин степени 3, то есть можно найти две такие вершины степени 3, смежные друг с другом, но соединённые с двумя разными цепями. В данном примере это u_1 и u_4. Если дуга направлена из u_4 в u_1, то при удалении u_2 и u_3 опять получаем неизоморфные графы.

Аналогично рассматривается предположение, что вершины u_5 и u_6 — источники.

Если вершины u_5 и u_6 имеют полустепени (1,1), то при удалении u_1 и удалении u_2 опять получим неизоморфные графы, так как в этих графах u_5 будет иметь разные полустепени — (0,1) и (1,0).

Аналогичным образом рассматривается случай, когда граф G^* выглядит как цикл, в которому присоединены цепи длины 2.

Таким образом, ориентации графов из третьего случая теоремы 5 также не имеют МВ-1Р с тремя дополнительными дугами. ■

Теорема 10. Среди связных орграфов только четыре имеют МВ-1Р с тремя дополнительными дугами: 3-вершинный транзитивный турнир, две 3-вершинные цепи и одна 4-вершинная цепь.

Доказательство. Используя доказанные выше теоремы, получаем, во-первых, что 3-вершинный транзитивный турнир имеет МВ-1Р с тремя дополнительными дугами. Во-вторых, непосредственной проверкой можно убедиться, что среди цепей с числом вершин меньше 5 есть три цепи, имеющие МВ-1Р с тремя дополнительными дугами. Эти графы и их МВ-1Р приведены на рис. 11.

Рис. 11. Графы из теоремы 10 и их МВ-1Р

Теорема доказана. ■

ЛИТЕРАТУРА

1. *Абросимов М. Б.* Графовые модели отказоустойчивости. Саратов: Изд-во Сарат. ун-та, 2012. 192 с.
2. *Hayes J. P.* A graph model for fault-tolerant computing system // IEEE Trans. Comput. 1976. V. C.25. No. 9. P. 875–884.
3. *Абросимов М. Б.* О сложности некоторых задач, связанных с расширениями графов // Матем. заметки. 2010. № 5(88). С. 643–650.
4. *Абросимов М. Б.* Характеризация графов с заданным числом дополнительных ребер минимального вершинного 1-расширения // Прикладная дискретная математика. 2012. № 1. С. 111–120.
5. *Абросимов М. Б.* Минимальные вершинные расширения направленных звезд // Дискретная математика. 2011. Т. 23. № 2. С. 93–102.
6. *Абросимов М. Б., Моденова О. В.* Характеризация орграфов с малым числом дополнительных дуг минимального вершинного 1-расширения // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2013. Т. 13. Вып. 2. Ч. 2. С. 3–9.
7. *Абросимов М. Б., Долгов А. А.* О бесконтурных точных расширениях // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2010. Т. 10. Вып. 1. С. 3–9.
8. *Абросимов М. Б.* Минимальные расширения транзитивных турниров // Вестник Томского государственного университета. Приложение. 2006. № 17. С. 187–190.

УДК 519.87

О ПОСТРОЕНИИ ЦИРКУЛЯНТНЫХ СЕТЕЙ РАЗМЕРНОСТИ ЧЕТЫРЕ С МАКСИМАЛЬНЫМ ЧИСЛОМ ВЕРШИН ПРИ ЛЮБОМ ДИАМЕТРЕ

Э. А. Монахова

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: emilia@rav.sscc.ru

Рассматривается задача оптимизации неориентированных циркулянтных сетей, состоящая в максимизации числа вершин при заданных степени и диаметре графа. Получена новая нижняя оценка достижимого числа вершин циркулянтных сетей размерности четыре при любых диаметрах. Построены новые бесконечные семейства циркулянтов, достигающих найденной оценки. Для графов найденных семейств получены различные аналитические описания.

Ключевые слова: *неориентированные циркулянтные сети, диаметр, максимальный порядок графа.*

Введение

Циркулянтные сети (графы) являются графами Кэли абелевых групп и широко изучаются в теории графов и дискретной математике, а также играют важную роль в разнообразных приложениях, в том числе в теории кодирования, распределённых вычислениях, моделировании химических реакций, при построении и анализе топологий сетей (оптических, клеточных, нейронных) и высокопроизводительных мультипроцессорных систем [1 – 11].

Пусть $s_1, s_2, \ldots, s_k, n$ — целые числа, такие, что $1 \leqslant s_1 < s_2 < \ldots < s_k < n$, и $S = (s_1, s_2, \ldots, s_k)$. Неориентированный граф $C(n; S)$ с множествами вершин $V = \{0, 1, \ldots, n-1\}$ и рёбер $E = \{(i, j) : |i - j| \equiv s_l \pmod n, l = 1, \ldots, k\}$ называется *циркулянтным*, числа из множества S — *образующими*, k — *размерностью*, n — *порядком* графа.

Диаметром графа G называется $\mathrm{d}(G) = \max\limits_{i,j \in V} \mathrm{d}(i, j)$, где $\mathrm{d}(i, j)$ — длина кратчайшего пути из вершины i в вершину j графа G.

Пусть $M(d, k)$ для любых натуральных d и k — максимально возможное (достижимое) натуральное n, такое, что существует множество образующих $S = (1, s_2, \ldots, s_k)$, при котором $\mathrm{d}(C(n; S)) \leqslant d$. Известно [11, 12], что

$$M(d, k) \leqslant P(d, k) = \sum_{i=0}^{k} C_k^i C_d^{k-i} 2^{k-i},$$

где верхняя граница $P(d, k)$ может рассматриваться как граница Мура для циркулянтных графов размерности k. Получение точных значений функции $M(d, k)$ для $k \geqslant 3$ достаточно трудоёмко и сводится к полному перебору параметрических описаний графа. Нижние оценки для $M(d, k)$ в каждом отдельном случае могут зависеть от рассматриваемого диаметра и, как правило, получаются посредством поиска бесконечных семейств графов, достигающих этих оценок [1, 13, 14].

Наиболее полный обзор результатов по оценкам диаметра и достижимого порядка k-мерных, $k \geqslant 2$, неориентированных циркулянтных графов можно найти в [3]. Более ранние обзоры представлены в работах [1, 2]. Для размерности 2 задача построения циркулянтов с единичной образующей и максимальным порядком $M(d,2)$, совпадающим с $P(d,2) = 2d(d+1)+1$ при любом диаметре d, решена (для ссылок см. [3]). Для размерности 3 нижняя оценка достижимого порядка циркулянтов последовательно улучшалась в [1, 11, 15]. В [16] найдена экстремальная функция $M(d,3)$ для любого диаметра d и построены бесконечные семейства трёхмерных циркулянтов с порядком, совпадающим с $M(d,3)$, вид которой, как оказалось, зависит от класса вычетов d по модулю 3. В [17] доказано, что существует граф Кэли абелевой группы с тремя образующими, который имеет диаметр d и размер, совпадающий с $M(d,3)$. Следует отметить, что открытые семейства двумерных и трёхмерных циркулянтных сетей успешно применяются в теории кодирования при построении совершенных кодов, исправляющих ошибки. В последнее время для построения таких кодов начали рассматривать четырёхмерные циркулянтные графы и циркулянты более высоких размерностей [5]. Наряду с двух- и трёхмерными циркулянтами, циркулянтные сети размерности 4 также используют как базовые элементы топологии при построении мультипроцессорных кластерных систем [8]. Важно подчеркнуть, что, взятые в качестве сетей связи мультипроцессорных систем, графы с максимальным (или близким к нему) порядком при заданных размерности и диаметре имеют высокие отказоустойчивость и скорость коммуникаций, минимальную задержку и максимальные связность и надёжность.

В данной работе будем исследовать максимально возможные циркулянтные сети размерности четыре с единичной образующей.

Для графов размерности $k \geqslant 4$ известны следующие результаты. В [11] доказано, что при $n = t^k$, t — нечётное число, графы $C(n; 1, t, \ldots, t^{k-1})$ имеют диаметр

$$d = \frac{k}{2}(t-1) = \frac{k}{2}n^{1/k} - \frac{k}{2}.$$

В частности, $n = d^4/16 + \mathrm{O}(d^3)$ для $k = 4$ и диаметра $d \equiv 0 \pmod 4$. В [9] изучены топологические свойства мультипликативных циркулянтов вида $C(n; 1, t, \ldots, t^{k-1})$ с чётным $t \geqslant 2$ и $n = t^k$ и получен их диаметр

$$d = k\frac{t}{2} - \left\lfloor \frac{k}{2} \right\rfloor.$$

Для $k = 4$ и диаметра $d \equiv 2 \pmod 4$ снова $n = d^4/16 + \mathrm{O}(d^3)$. Эти результаты улучшены в [15]: пусть d и k — любые положительные целые, такие, что $d \geqslant k \geqslant 3$, и пусть $t = \lfloor (d-k+3)/k \rfloor$, тогда

$$M(d,k) \geqslant 2t\sum_{i=0}^{k-1}(4t)^i = \frac{1}{2}\left(\frac{4}{k}\right)^k d^k + \mathrm{O}(d^{k-1}).$$

Отсюда для размерности четыре $M(d,4) \geqslant n = d^4/2 - (3/2)d^3 + \mathrm{O}(d^2)$. В [18] оценка функции $M(d,4)$ из [15] улучшена на $\mathrm{O}(3d^3/2)$ для любого нечётного диаметра. В [17] авторы, используя плотное покрытие решётками пространства $\mathbb{Z}^k$, при $k = 4$ нашли следующую оценку:

$$M(d,4) \geqslant n = \begin{cases} d^4/2 + d^3 & \text{при чётном } d, \\ 8\lfloor d/2 \rfloor \lceil d/2 \rceil^3 & \text{при нечётном } d. \end{cases}$$

В [19] оценка функции $M(d,4)$ из [17] улучшена для диаметров $d \equiv 0 \pmod 4$.

В данной работе в п. 1 получена новая нижняя оценка экстремальной функции $M(d,4)$ для любых диаметров $d > 1$ и построены бесконечные семейства циркулянтных сетей, достигающих найденной оценки, в п. 2 получены различные аналитические описания для графов найденных семейств, в п. 3 обсуждается проблема получения функции $M(d,4)$.

1. Новые семейства циркулянтных графов размерности четыре

Теоремы 1 и 2 позволяют улучшить нижнюю оценку функции $M(d,4)$ соответственно для чётных и нечётных диаметров $d > 1$. Это достигается путём построения бесконечных семейств циркулянтов $\{C(n(d); 1, s_2(d), s_3(d), s_4(d)) : d \geqslant 2\}$, достигающих найденной оценки.

Теорема 1. Пусть $d \geqslant 2$ — чётное число, $S = (1, s_2, s_3, s_4)$, где

$$s_2 = d+1, \quad s_3 = d^3/2 + d^2/2 + d, \quad s_4 = d^3/2 + 3d^2/2 + 3d + 2; \tag{1}$$

$$n = d^4/2 + d^3 + 5d^2/2 + 2d + 1. \tag{2}$$

Тогда $\mathrm{d}(C(n;S)) = d$.

Теорема 2. Пусть $d > 1$ — нечётное число, $S = (1, s_2, s_3, s_4)$, где

$$s_2 = d, \quad s_3 = d^3/2 + 3d/2, \quad s_4 = d^3/2 + d^2 + 3d/2 + 1, \tag{3}$$

$$n = d^4/2 + d^3 + 5d^2/2 + 2d + 1.$$

Тогда $\mathrm{d}(C(n;S)) = d$.

Пронумеруем вершины циркулянтного графа от 0 до $n-1$. Длину кратчайшего пути из вершины 0 в вершину x обозначим через $D(x) = \mathrm{d}(0,x)$, $0 \leqslant x < n$. Определим $+s_4$- и $-s_4$-связи из x, если идти по ним в вершину $x + s_4 \pmod n$ (в прямом направлении) или в $x - s_4 \pmod n$ (в обратном направлении) соответственно. Аналогично определим $+s_3$- и $-s_3$-связи, $+s_2$- и $-s_2$-связи и $+1$- и -1-связи. Пусть $\Delta = s_4 - s_3$. Кроме шагов длины 1 будем также использовать $\pm\Delta$-связи длины 2.

Предварительно докажем следующее утверждение.

Лемма 1. В циркулянтном графе $C(n; 1, s_2, s_3, s_4)$ с порядком, заданным (2), и образующими (1) для чётного $d \geqslant 2$ или (3) для нечётного $d > 1$ любой интервал вершин с номерами $[a,b]$ длины $b - a = \Delta/2$ с суммой $D(a) + D(b) = d$ для чётного d или $D(a) + D(b) = d+1$ для нечётного d имеет свойство

$$\max_{a \leqslant x \leqslant b} D(x) = d. \tag{4}$$

Доказательство. Пусть $D(a) = l$,

$$D(b) = \begin{cases} d - l, & l = 0, \ldots, d, \quad \text{при чётном } d, \\ d + 1 - l, & l = 1, \ldots, d, \quad \text{при нечётном } d. \end{cases}$$

Все вершины с номерами из $[a,b]$ можно достичь либо $+s_2$- и ±1-связями из a, либо $-s_2$- и ±1-связями из b. Номер первой вершины от a, для которой $D(x) = d+1$ (без учёта $-s_2$-связей из b), есть

$$x = \begin{cases} a + (d/2 - l)s_2 + d/2 + 1 & \text{при чётном } d, \\ a + (\lceil d/2 \rceil - l)s_2 + \lceil d/2 \rceil & \text{при нечётном } d. \end{cases}$$

С другой стороны, учитывая $-s_2$-связи из b, имеем

$$x = \begin{cases} b - ls_2 & \text{при чётном } d, \\ b - (l-1)s_2 & \text{при нечётном } d, \end{cases}$$

то есть существует путь длины d в данную вершину. Для остальных вершин с расстоянием до 0, превышающим d, использование $-s_2$- и ± 1-связей из b также уменьшает до значения d функцию расстояний до вершины 0. В частности, свойство (4) выполняется, когда $D(a) = D(b) = \lceil d/2 \rceil$. ■

Теперь можем написать общее доказательство к теоремам 1 и 2.

Доказательство теорем 1 и 2. Пусть $d \geqslant 2$ — целое число. Рассмотрим циркулянтные графы $C(n; 1, s_2, s_3, s_4)$ с порядками n, заданными (2), и образующими s_i, $i = 2, 3, 4$, заданными (1) при чётном d или (3) при нечётном d.

В силу (2) и (1) имеем при чётном d

$$n = (d/2)\sum_{i=1}^{4} s_i + 1, \quad n = (d-1)s_4 + \Delta + s_2, \quad s_4 = (d+1)\Delta/2 + s_2,$$

где $s_2 = d + 1$; $\Delta = (d+1)^2 + 1 = s_2^2 + 1$.

В силу (2) и (3) имеем при нечётном d

$$n = \lceil d/2 \rceil \sum_{i=1}^{4} s_i - s_2, \quad n = ds_4 + \Delta + s_2, \quad s_4 = (d+2)\Delta/2 + s_2,$$

где $s_2 = d$; $\Delta = d^2 + 1 = s_2^2 + 1$.

Заметим, что числовые значения s_2 (а также Δ) равны для двух взятых последовательно чётных и нечётных значений d.

Требуется доказать, что $D(x) \leqslant d$ для любой вершины $0 \leqslant x < n$. Поскольку циркулянтные графы являются вершинно-транзитивными, достаточно рассмотреть вершины с номерами $0 \leqslant x \leqslant (n-1)/2$. Удобнее рассмотреть вершины с номерами $0 \leqslant x \leqslant \lceil d/2 \rceil s_4 > (n-1)/2$.

В графе $C(n; 1, s_2, s_3, s_4)$ справедливо соотношение

$$[0, \lceil d/2 \rceil s_4] = \bigcup_{i=1}^{\lceil d/2 \rceil} (A_i \cup B_i \cup C_i),$$

где $A_i = [(i-1)s_4, (i-1)s_4 + s_2]$; $B_i = [(i-1)s_4 + s_2, is_3]$; $C_i = [is_3, is_4]$.

1. Пусть $x \in \bigcup\limits_{i=1}^{\lceil d/2 \rceil} C_i$ и $x_{ik} = is_3 + ks_4$, $y_{ik} = x_{ik} + \Delta/2$. Тогда

$$\bigcup_{i=1}^{\lceil d/2 \rceil} C_i = \bigcup_{i=1}^{\lceil d/2 \rceil} \bigcup_{k=0}^{\lceil d/2 \rceil - i} ([x_{ik}, y_{ik}] \cup [y_{ik}, x_{i-1,k+1}]).$$

Отсюда следует, что $D(x_{ik}) = D(x_{i-1,k+1}) = i + k \leqslant \lceil d/2 \rceil$ (равенство имеет место при $k = \lceil d/2 \rceil - i$). Так как $y_{ik} = n - (\lceil d/2 \rceil - i)s_3 - (\lceil d/2 \rceil - k)s_4$, имеем

$$D(y_{ik}) = \begin{cases} d - (i+k) & \text{при чётном } d, \\ d + 1 - (i+k) & \text{при нечётном } d. \end{cases}$$

Тем самым

$$D(x_{ik}) + D(y_{ik}) = D(y_{ik}) + D(x_{i-1,k+1}) = \begin{cases} d & \text{при чётном } d, \\ d+1 & \text{при нечётном } d \end{cases}$$

для всех $i = 1, 2, \ldots, \lceil d/2 \rceil$ и $k = 0, 1, \ldots, \lceil d/2 \rceil - i$. Таким образом, интервалы $[x_{ik}, y_{ik}]$ и $[y_{ik}, x_{i-1,k+1}]$ обладают свойством (4), а значит, $D(x) \leqslant d$ для любой вершины $x \in \bigcup\limits_{i=1}^{\lceil d/2 \rceil} C_i$.

2. Пусть $x_{ik} = (i-1)s_4 + s_2 + k\Delta$, $y_{ik} = x_{ik} + \Delta/2$ при $k = 0, 1, \ldots, \lceil d/2 \rceil - 1 - i$, $x_{i,\lceil d/2 \rceil - i} = is_3 - \Delta/2 = (\lceil d/2 \rceil - 1)s_4 - (\lceil d/2 \rceil - i)s_3 + s_2$, $y_{i,\lceil d/2 \rceil - i} = is_3$.

Тогда

$$\bigcup_{i=1}^{\lceil d/2 \rceil} B_i = \bigcup_{i=1}^{\lceil d/2 \rceil} \left\{ \bigcup_{k=0}^{\lceil d/2 \rceil - 1 - i} ([x_{ik}, y_{ik}] \cup [y_{ik}, x_{i,k+1}]) \cup [x_{i,\lceil d/2 \rceil - i}, y_{i,\lceil d/2 \rceil - i}] \right\}.$$

Так как для всех $x \in [x_{i,\lceil d/2 \rceil - i}, y_{i,\lceil d/2 \rceil - i}]$, $i = 1, \ldots, \lceil d/2 \rceil$, имеет место равенство

$$D(x_{i,\lceil d/2 \rceil - i}) + D(y_{i,\lceil d/2 \rceil - i}) = \begin{cases} d & \text{при чётном } d, \\ d+1 & \text{при нечётном } d, \end{cases}$$

данные интервалы вершин обладают свойством (4), а значит, $D(x) \leqslant d$ для любой вершины $x \in \bigcup\limits_{i=1}^{\lceil d/2 \rceil} [x_{i,\lceil d/2 \rceil - i}, y_{i,\lceil d/2 \rceil - i}]$.

Остаётся рассмотреть случай, когда $x \in \bigcup\limits_{i=1}^{\lceil d/2 \rceil - 1} B_i$ (при $d > 2$). Из определения x_{ik} следует, что $D(x_{ik}) = i + 2k$. Учитывая, что

$$s_2 + \Delta/2 = \lceil d/2 \rceil s_3 - (\lceil d/2 \rceil - 1)s_4,$$

получим $y_{ik} = (\lceil d/2 \rceil - k)s_3 - (\lceil d/2 \rceil - i - k)s_4$ и

$$D(y_{ik}) = \begin{cases} d - i - 2k & \text{при чётном } d, \\ d + 1 - i - 2k & \text{при нечётном } d. \end{cases}$$

Таким образом,

$$D(x_{ik}) + D(y_{ik}) = \begin{cases} d & \text{при чётном } d, \\ d+1 & \text{при нечётном } d \end{cases}$$

для интервалов $[x_{ik}, y_{ik}]$ длины $\Delta/2$. По свойству (4) $D(x) \leqslant d$ для всех $x_{ik} \leqslant x \leqslant y_{ik}$, $i = 1, 2, \ldots, \lceil d/2 \rceil - 1$, $k = 0, 1, \ldots, \lceil d/2 \rceil - 1 - i$.

Рассмотрим интервалы вершин $[y_{ik}, x_{i,k+1}]$. Имеем

$$x_{i,k+1} = (i-1)s_4 + (k+1)\Delta + s_2, \quad D(x_{i,k+1}) = i + 2k + 2.$$

Отсюда следует, что

$$D(y_{ik}) + D(x_{i,k+1}) = \begin{cases} d+2 & \text{при чётном } d, \\ d+3 & \text{при нечётном } d \end{cases}$$

для всех $i = 1, 2, \ldots, \lceil d/2 \rceil - 1$, $k = 0, 1, \ldots, \lceil d/2 \rceil - 1 - i$. Разобьём каждый интервал $[y_{ik}, x_{i,k+1}]$ на две части: $[y_{ik}, z_{ik}]$ и $[z_{ik}, x_{i,k+1}]$, где $z_{ik} = x_{i,k+1} - s_2 = (i-1)s_4 + (k+1)\Delta$. Из определения z_{ik} следует, что $D(z_{ik}) = i + 2k + 1$.

Интервал $[y_{ik}, z_{ik}]$ имеет длину, равную $\Delta/2 - s_2$, и

$$D(y_{ik}) + D(z_{ik}) = \begin{cases} d+1 & \text{при чётном } d, \\ d+2 & \text{при нечётном } d. \end{cases}$$

По этой причине для обеспечения условия $D(x) \leqslant d$ на интервале $[y_{ik}, z_{ik}]$ достаточно использовать $+s_2$- и ± 1-связи из y_{ik} и $-s_2$- и ± 1-связи из z_{ik}.

На интервале $[z_{ik}, x_{i,k+1}]$ длины s_2 максимум $D(x)$ при использовании ± 1-связей достигается в вершине $x_0 = x_{i,k+1} - \lfloor d/2 \rfloor$. Но существует путь в данном графе в вершину x_0 из $y_{i,k+1}$, содержащий следующее количество образующих s_2:

$$\begin{cases} d/2+1 & \text{при чётном } d, \\ \lceil d/2 \rceil & \text{при нечётном } d. \end{cases}$$

Поскольку

$$D(y_{i,k+1}) = \begin{cases} d-1-D(z_{ik}) & \text{при чётном } d, \\ d-D(z_{ik}) & \text{при нечётном } d, \end{cases}$$

имеем $D(x_0) = \lceil 3d/2 \rceil - D(z_{ik})$. Отсюда $D(x_0) \leqslant d$ при $D(z_{ik}) \geqslant \lceil d/2 \rceil$. Нетрудно определить, что $D(x_0) \leqslant d$ и при $D(z_{ik}) < \lceil d/2 \rceil$. Таким образом, $D(x) \leqslant d$ для всех $z_{ik} \leqslant x \leqslant x_{i,k+1}$.

3. Пусть $x \in A_i$, $i = 1, 2, \ldots, \lceil d/2 \rceil$, $A_i = [(i-1)s_4, (i-1)s_4 + s_2] = [x_i, z_i]$.

Так как $D(x_i) = i - 1$, $D(z_i) = i$, возьмём самое большое значение $i = \lceil d/2 \rceil$, для него $\max\limits_{x_i \leqslant x \leqslant z_i} D(x) = d$. Тогда $\max\limits_{x_i \leqslant x \leqslant z_i} D(x) \leqslant d$ для всех $i = 1, \ldots, \lceil d/2 \rceil$.

4. Строго говоря, там, где получено $D(x) = d$, может быть $D(x) \leqslant d$. Поэтому остаётся показать, что в данных графах существует хотя бы одна такая вершина x, что $D(x) = d$.

Пусть $d \equiv 0 \pmod 4$. В качестве искомой вершины возьмем $x = (n-1)/2 = \dfrac{d}{4}(s_1 + {} + s_2 + s_3 + s_4)$. Видно, что в данную вершину существует путь из 0 длины d. Он же является минимально возможным, так как $x - n = -\dfrac{d}{4}(s_1 + s_2 + s_3 + s_4) - 1$ даёт путь длины $d+1$ и длины всех других возможных путей из 0 в x не менее d. Следовательно, $D(x) = d$.

В остальных случаях в качестве искомых вершин берем $x = (n+1)/2 = \dfrac{d+3}{4}s_4 + {} + \dfrac{d-1}{4}s_3 - \dfrac{d-1}{4}s_2 - \dfrac{d-1}{4}$ при $d \equiv 1 \pmod 4$, $x = (n-1)/2 = \dfrac{d+2}{4}s_4 + \dfrac{d-2}{4}s_3 - {} - \dfrac{d+2}{4}s_2 + \dfrac{d-2}{4}$ при $d \equiv 2 \pmod 4$, $x = (n+1)/2 = \dfrac{d+1}{4}s_4 + \dfrac{d+1}{4}s_3 + \dfrac{d+1}{4}s_2 - {} - \dfrac{d-3}{4}$ при $d \equiv 3 \pmod 4$. Доказательства аналогичны случаю, когда $d \equiv 0 \pmod 4$. Теоремы 1 и 2 доказаны.

2. Другие аналитические описания для графов найденных семейств

Для полученного семейства графов с чётными диаметрами из теоремы 1 найдены аналитически другие виды образующих.

Утверждение 1. Пусть d — чётное число, $S^{(2)} = \left(1, s_2^{(2)}, s_3^{(2)}, s_4^{(2)}\right)$, где

$$s_2^{(2)} = d^2/2 + d + 2, \quad s_3^{(2)} = d^3/2 + d^2/2 + 2d, \quad s_4^{(2)} = d^3/2 + d^2 + 2d + 1, \qquad (5)$$

и $n = d^4/2 + d^3 + 5d^2/2 + 2d + 1$. Тогда $\mathrm{d}(C(n; S^{(2)})) = d$.

Доказательство. Достаточно показать, что образующие $S^{(2)}$ получаются из образующих S с помощью эквивалентного преобразования [3]. Для этого умножим каждую образующую из $S = (1, s_2, s_3, s_4)$ на $s_3^{(2)}$, поскольку применение алгоритма

Евклида к n и $s_3^{(2)}$ даёт $\left(n, s_3^{(2)}\right) = 1$. Образующая $s_1 = 1$ переходит в $s_3^{(2)}$. Образующая s_2 переходит в $s_1^{(2)} = 1$, поскольку $s_2 \cdot s_3^{(2)} = n - 1$. Образующая s_3 переходит в $s_4^{(2)}$, поскольку $s_3 \cdot s_3^{(2)} = (d^2/2+1)n - s_4^{(2)}$. Образующая s_4 переходит в $s_2^{(2)}$, поскольку $s_4 \cdot s_3^{(2)} = (d^2/2 + d + 2)n - s_2^{(2)}$. Таким образом, получили изоморфное отображение образующих $(1, s_2, s_3, s_4)$ соответственно в образующие $\left(s_3^{(2)}, 1, s_4^{(2)}, s_2^{(2)}\right)$. ■

Утверждение 2. Пусть d — чётное число, $S^{(3)} = \left(1, s_2^{(3)}, s_3^{(3)}, s_4^{(3)}\right)$, где

$$s_2^{(3)} = d^3 + d^2 + 3d + 1, \quad s_3^{(3)} = d^3 + d^2 + 4d + 1, \quad s_4^{(3)} = d^3 + 2d^2 + 4d + 3, \qquad (6)$$

и $n = d^4/2 + d^3 + 5d^2/2 + 2d + 1$. Тогда $\mathrm{d}(C(n; S^{(3)})) = d$.

Доказательство. Покажем, что образующие $S^{(3)}$ получаются из образующих S с помощью эквивалентного преобразования. Для этого умножим каждую образующую из $S = (1, s_2, s_3, s_4)$ на $s_4^{(3)}$ (применение алгоритма Евклида к n и $s_4^{(3)}$ даёт $\left(n, s_4^{(3)}\right) = 1$). Образующая $s_1 = 1$ переходит в $s_4^{(3)}$. Образующая s_2 переходит в $s_2^{(3)}$, так как $s_2 \cdot s_4^{(3)} = 2n + s_2^{(3)}$. Образующая s_3 переходит в 1, так как $s_3 \cdot s_4^{(3)} = (d^2 + d + 1)n - 1$. Образующая s_4 переходит в $s_3^{(3)}$, так как $s_4 \cdot s_4^{(3)} = (d^2 + 3d + 5)n + s_3^{(3)}$. Таким образом, получили изоморфное отображение $(1, s_2, s_3, s_4)$ соответственно в $\left(s_4^{(3)}, s_2^{(3)}, 1, s_3^{(3)}\right)$. ■

В табл. 1 даны примеры описаний циркулянтов размерности 4 порядка n из теоремы 1 с тремя видами образующих (1), (5), (6).

Таблица 1

Описания циркулянтов размерности 4 для чётных диаметров d

d	n	s_1,	s_2,	s_3,	s_4	$s_1^{(2)}$,	$s_2^{(2)}$,	$s_3^{(2)}$,	$s_4^{(2)}$	$s_1^{(3)}$,	$s_2^{(3)}$,	$s_3^{(3)}$,	$s_4^{(3)}$
2	31	1,	3,	8,	18	1,	6,	10,	13	1,	19,	21,	27
4	241	1,	5,	44,	70	1,	14,	48,	57	1,	93,	97,	115
6	967	1,	7,	132,	182	1,	26,	138,	157	1,	271,	277,	315
8	2737	1,	9,	296,	378	1,	42,	304,	337	1,	601,	609,	675
10	6271	1,	11,	560,	682	1,	62,	570,	621	1,	1131,	1141,	1243
12	12481	1,	13,	948,	1118	1,	86,	960,	1033	1,	1909,	1921,	2067
14	22471	1,	15,	1484,	1710	1,	114,	1438,	1597	1,	2983,	2997,	3195
16	37537	1,	17,	2192,	2482	1,	146,	2208,	2337	1,	4401,	4417,	4675
18	59167	1,	19,	3096,	3458	1,	182,	3114,	3277	1,	6211,	6229,	6555

Отметим, что значения Δ, $\Delta^{(2)} = s_4^{(2)} - s_3^{(2)}$ и $\Delta^{(3)} = s_4^{(3)} - s_3^{(3)}$ могут быть получены из формулы

$$n = 2(d^2/2 + 1)M(d/2, 2) + d^2/2 - 1,$$

из которой следует, что $\Delta = 2M(d/2, 2)$; $\lfloor n/\Delta \rfloor = d^2/2 + 1$; $\Delta^{(2)} = d^2/2 + 1$; $\lfloor n/\Delta^{(2)} \rfloor = 2M(d/2, 2)$; $\Delta^{(3)} = d^2 + 2$; $\lfloor n/\Delta^{(3)} \rfloor = M(d/2, 2)$.

Найденные образующие (1), (5), (6) представляют собой полиномы относительно d. Вторые образующие во всех трёх видах — полиномы первой, второй и третьей степеней для образующих (1), (5), (6) соответственно, а третьи и четвертые образующие — полиномы третьей степени. В следующем утверждении все четыре образующие графов рассматриваемого семейства — полиномы третьей степени.

Утверждение 3. Пусть $d > 2$ — чётное число, $S^{(4)} = \left(s_1^{(4)}, s_2^{(4)}, s_3^{(4)}, s_4^{(4)}\right)$, где $s_1^{(4)} = d^3/2 - 3d^2/2 - d - 3$, $s_2^{(4)} = d^3 + 2d^2 + 6d + 6$, $s_3^{(4)} = d^3 + 3d^2 + 8d + 4$, $s_4^{(4)} = 7d^3/2 + 13d^2/2 + 13d + 5$, и $n = d^4/2 + d^3 + 5d^2/2 + 2d + 1$. Тогда $\mathrm{d}(C(n; S^{(4)})) = d$.

Доказательство. Покажем, что образующие $S^{(4)}$ получаются из образующих S с помощью эквивалентного преобразования. Для этого умножим каждую образующую из $S = (1, s_2, s_3, s_4)$ на $s_2^{(4)}$. Образующая $s_1 = 1$ переходит в $s_2^{(4)}$. Образующая s_2 переходит в $s_3^{(4)}$, так как $s_2 \cdot s_2^{(4)} = 2n + s_3^{(4)}$. Образующая s_3 переходит в $s_1^{(4)}$, так как $s_3 \cdot s_2^{(4)} = (d^2 + d + 3)n + s_1^{(4)}$. Образующая s_4 переходит в $s_4^{(4)}$, так как $s_4 \cdot s_2^{(4)} = (d^2 + 3d + 7)n + s_4^{(4)}$. Таким образом, получили изоморфное отображение $(1, s_2, s_3, s_4)$ соответственно в $\left(s_2^{(4)}, s_3^{(4)}, s_1^{(4)}, s_4^{(4)}\right)$. ■

Отметим, что только при $d = 4$ образующая $s_1^{(4)}$ равна 1.

Для семейства графов с нечётными диаметрами из теоремы 2 также существуют другие аналитические описания образующих.

Утверждение 4. Пусть $d > 1$ — нечётное число, $S^{(2)} = \left(1, s_2^{(2)}, s_3^{(2)}, s_4^{(2)}\right)$, где

$$s_2^{(2)} = (d^2 + 3)/2, \quad s_3^{(2)} = (d^3 + d^2 + 3d + 1)/2, \quad s_4^{(2)} = d^3/2 + d^2 + 5d/2 + 2, \qquad (7)$$

и $n = d^4/2 + d^3 + 5d^2/2 + 2d + 1$. Тогда $\mathrm{d}(C(n; S^{(2)})) = d$.

Доказательство. Достаточно показать, что образующие $S^{(2)}$ получаются из образующих S с помощью эквивалентного преобразования. Для этого умножим каждую образующую из $S = (1, s_2, s_3, s_4)$ на $s_4^{(2)}$, поскольку применение алгоритма Евклида к n и $s_4^{(2)} = (n-1)/d$ даёт $\left(n, s_4^{(2)}\right) = 1$. Образующая s_1 переходит в $s_4^{(2)}$. Образующая s_2 переходит в $s_1^{(2)} = 1$, поскольку $s_2 \cdot s_4^{(2)} = n - 1$. Образующая s_3 переходит в $s_2^{(2)}$, поскольку $s_3 \cdot s_4^{(2)} = (d^2 + 3)/2 \cdot n - s_2^{(2)}$. Образующая s_4 переходит в $s_3^{(2)}$, поскольку $s_4 \cdot s_4^{(2)} = (d^2 + 2d + 3)/2 \cdot n + s_3^{(2)}$. Таким образом, получили изоморфное отображение образующих $(1, s_2, s_3, s_4)$ соответственно в образующие $\left(s_4^{(2)}, 1, s_2^{(2)}, s_3^{(2)}\right)$. ■

Утверждение 5. Пусть $d > 1$ — нечётное число, $S^{(3)} = \left(1, s_2^{(3)}, s_3^{(3)}, s_4^{(3)}\right)$, где

$$s_2^{(3)} = d^3 + d^2 + 3d, \quad s_3^{(3)} = d^3 + 2d^2 + 4d + 2, \quad s_4^{(3)} = d^3 + 2d^2 + 5d + 3, \qquad (8)$$

и $n = d^4/2 + d^3 + 5d^2/2 + 2d + 1$. Тогда $\mathrm{d}(C(n; S^{(3)})) = d$.

Доказательство. Покажем, что образующие $S^{(3)}$ получаются из образующих S с помощью эквивалентного преобразования. Умножим каждую образующую из $S = (1, s_2, s_3, s_4)$ на $s_2^{(3)}$ (применение алгоритма Евклида к n и $s_2^{(3)}$ даёт $\left(n, s_2^{(3)}\right) = 1$). Образующая $s_1 = 1$ переходит в $s_2^{(3)}$. Образующая s_2 переходит в $s_3^{(3)}$, так как $s_2 \cdot s_2^{(3)} = 2n - s_3^{(3)}$. Образующая s_3 переходит в $s_4^{(3)}$, так как $s_3 \cdot s_2^{(3)} = (d^2 - d + 3)n - s_4^{(3)}$. Образующая s_4 переходит в 1, так как $s_4 \cdot s_2^{(3)} = (d^2 + d + 1)n - 1$. Таким образом, получили изоморфное отображение $(1, s_2, s_3, s_4)$ соответственно в $\left(s_2^{(3)}, s_3^{(3)}, s_4^{(3)}, 1\right)$. ■

В табл. 2 приведены примеры описаний циркулянтов размерности 4 порядка n из теоремы 2 с тремя видами образующих (3), (7), (8).

Таблица 2

Описания циркулянтов размерности 4 для нечётных диаметров d

d	n	s_1,	s_2,	s_3,	s_4	$s_1^{(2)}$,	$s_2^{(2)}$,	$s_3^{(2)}$,	$s_4^{(2)}$	$s_1^{(3)}$,	$s_2^{(3)}$,	$s_3^{(3)}$,	$s_4^{(3)}$
3	97	1,	3,	18,	28	1,	6,	23,	32	1,	45,	59,	63
5	511	1,	5,	70,	96	1,	14,	83,	102	1,	165,	197,	203
7	1681	1,	7,	182,	232	1,	26,	207,	240	1,	413,	471,	479
9	4231	1,	9,	378,	460	1,	42,	419,	470	1,	837,	929,	939
11	8977	1,	11,	682,	804	1,	62,	743,	816	1,	1485,	1619,	1631
13	16927	1,	13,	1118,	1288	1,	86,	1203,	1302	1,	2405,	2589,	2603
15	29281	1,	15,	1710,	1936	1,	114,	1823,	1952	1,	3645,	3887,	3903
17	47431	1,	17,	2482,	2772	1,	146,	2627,	2790	1,	5253,	5561,	5579

3. К вопросу об экстремальной функции $M(d, 4)$

Из теорем 1 и 2 следует, что найденная нижняя оценка (2) достижимого числа вершин циркулянтных сетей размерности 4 и любого диаметра $d > 1$ лучше на $\mathrm{O}(5d^3/2)$ оценки из [15] и на $\mathrm{O}(5d^2/2)$ — оценки из [17]. В табл. 3 приведены результаты сравнения порядков графов найденных семейств и графов из [15, 17] для диаметров $2 \leqslant d \leqslant 17$.

Таблица 3

Порядки известных графов и графов новых семейств при равных диаметрах

d	n [15]	n [17]	n	d	n [15]	n [17]	n
2		16	31	3		64	97
4	170	192	241	5	170	432	511
6	170	864	967	7	170	1536	1681
8	170	2560	2737	9	2340	4000	4231
10	2340	6000	6271	11	2340	8640	8977
12	2340	12096	12481	13	11310	16464	16927
14	11310	21952	22471	15	11310	28672	29281
16	11310	36864	37537	17	34952	46656	47431

С помощью программы перебора параметрических описаний циркулянтных графов показано при $k = 4$, что граф порядка $n = 241$ — максимально возможный циркулянтный граф диаметра $d = 4$. Представленные в табл. 4 (табл. 2 из [19]) наборы образующих для данного экстремального графа соответствуют образующим из теоремы 1, утверждений 1, 2 и 3, кроме образующих $(1, 31, 82, 86)$, для которых не удалось найти обобщающего аналитического описания.

В табл. 4 также даны максимально возможные циркулянты для других диаметров, полученные с помощью программы перебора.

Таблица 4

Описания максимальных циркулянтов диаметра d и размерности 4

d	$P(d, 4)$	$n = M(d, 4)$	$S = (s_1, s_2, s_3, s_4)$
1	9	9	(1, 2, 3, 4)
2	41	35	(1, 6, 7, 10), (1, 7, 11, 16)
3	129	99	(1, 24, 39, 44)
4	321	241	(1, 5, 44, 70), (1, 14, 48, 57), (1, 31, 82, 86), (1, 93, 97, 115)
5	681	511	(1, 5, 70, 96), (1, 14, 83, 102), (1, 165, 197, 203)

Формула (2) для порядков циркулянтных графов при $d \equiv 0 \pmod 4$, как отмечалось в выдвинутой в [19] гипотезе, задаёт значения экстремальной функции $M(d, 4)$ при $d \equiv 0 \pmod 4$. Вопрос, насколько близко подошли найденные семейства к значениям функции $M(d, 4)$ при других диаметрах, остаётся открытым.

ЛИТЕРАТУРА

1. *Bermond J.-C., Comellas F., and Hsu D. F.* Distributed loop computer networks: a survey // J. Parallel Distributed Comput. 1995. V. 24. P. 2–10.
2. *Hwang F. K.* A survey on multi-loop networks // Theor. Comp. Sci. 2003. No. 299. P. 107–121.
3. *Монахова Э. А.* Структурные и коммуникативные свойства циркулянтных сетей // Прикладная дискретная математика. 2011. № 3(13). С. 92–115.
4. *Нестеренко Б. Б., Новотарский М. А.* Клеточные нейронные сети на циркулянтных графах // Искусственный интеллект. 2009. № 3. С. 132–138.
5. *Martinez C., Beivide R., and Gabidulin E. M.* Perfect codes from Cayley graphs over Lipschitz integers // IEEE Trans. Inform. Theory. 2009. V. 55. No. 8. P. 3552–3562.
6. *Comellas F., Mitjana M., and Peters J. G.* Broadcasting in small-world communication networks // 9th Inter. Coll. on Structural Information and Communication Complexity (SIROCCO 9), 2002. Proc. Informatics. 2002. No. 13. P. 73–85.
7. *Narayanan L., Opatrny J., and Sotteau D.* All-to-all optical routing in chordal rings of degree four // Algorithmica. 2001. V. 31. No. 2. P. 155–178.
8. *Muga F. P., Saldana R. P., and Yu W. E. S.* Building graph-based symmetric cluster // NECTEC Techn. J. 2001. V. 11. No. 9. P. 195–199.
9. *Stojmenovic I.* Multiplicative circulant networks. Topological properties and communication algorithms // Discrete Appl. Math. 1997. No. 77. P. 281–305.
10. *Воробьев В. А.* Простейшие структуры однородных вычислительных систем // Вычислительные системы. Вопросы теории и построения ВС. Новосибирск, 1974. № 60. С. 35–49.
11. *Wong C. K. and Coppersmith D.* A combinatorial problem related to multimodule memory organizations // J. Assoc. Comp. Mach. 1974. No. 21. P. 392–402.
12. *Корнеев В. В.* О макроструктуре однородных вычислительных систем // Вычислительные системы. Вопросы теории и построения ВС. Новосибирск, 1974. № 60. С. 17–34.
13. *Macbeth H., Siagiova J., and Siran J.* Cayley graphs of given degree and diameter for cyclic, Abelian, and metacyclic groups // Discrete Math. 2012. V. 312. No. 1. P. 94–99.
14. *Jia X.-D. and Su W.* Triple loop networks with minimal transmission delay // Int. J. Found. Comp. Sci. 1997. V. 8. No. 3. P. 305–328.
15. *Chen S. and Jia X.-D.* Undirected loop networks // Networks. 1993. No. 23. P. 257–260.
16. *Monakhova E. A.* Optimal triple loop networks with given transmission delay: topological design and routing // Inter. Network Optimization Conf. (INOC'2003), Evry/Paris, France. 2003. P. 410–415.
17. *Dougherty R. and Faber V.* The degree-diameter problem for several varieties of Cayley graphs, 1: the Abelian case // SIAM J. Discrete Math. 2004. V. 17. No. 3. P. 478–519.
18. *Монахова Э. А.* Оптимизация циркулянтных сетей связи размерности четыре // Дискретный анализ и исследование операций. 2008. Т. 15. № 3. С. 58–64.
19. *Монахова Э. А.* Новая достижимая нижняя оценка числа вершин в циркулянтных сетях размерности четыре // Дискретный анализ и исследование операций. 2013. Т. 20. № 1. С. 37–44.

УДК 519.176

О ДВУХ ЗАДАЧАХ АППРОКСИМАЦИИ ВЗВЕШЕННЫХ ГРАФОВ И АЛГОРИТМАХ ИХ РЕШЕНИЯ

А. Р. Ураков, Т. В. Тимеряев

Уфимский государственный авиационный технический университет, г. Уфа, Россия

E-mail: urakov@ufanet.ru, timeryaev@yandex.ru

Рассматриваются две связанные задачи аппроксимации взвешенных графов. В качестве погрешности аппроксимации рассматривается абсолютная величина максимума разностей расстояний между вершинами исходного графа и соответствующими им вершинами графа аппроксимирующего. В первой задаче требуется минимизировать погрешность аппроксимации при заданной размерности аппроксимирующего графа. Во второй — минимизировать размерность аппроксимирующего графа при заданной погрешности. Для задач приводится постановка в терминах задачи разбиения графа, описываются эвристические алгоритмы решения и производится сравнение с решением известным алгоритмом разбиения графа.

Ключевые слова: *аппроксимация графа, разбиение графа, уменьшение сложности задач.*

Введение

Под аппроксимацией графа в общем случае понимают процесс приближения некоторой исходной информации о исследуемом графе (возможно, не обладающей полнотой) графом с нужными исследователю характеристиками. В качестве исходной информации может выступать либо граф, обладающий неудобными с точки зрения рассматриваемой задачи свойствами, либо набор данных, не представленный в виде графа, но требующий такого представления для возможности формулирования и решения задач на нём с использованием аппарата теории графов.

Известные задачи аппроксимации графа можно условно разделить на следующие категории: 1) восстановление исследуемой структуры по ограниченному объёму информации в виде графа с последующим изучением её свойств и механизмов посредством решения задач на получившемся приближённом графе [1]; 2) приближение исходного графа с одним числом рёбер (его матрицы связности) графом на тех же вершинах с другим числом рёбер (другой матрицей связности), обладающим определёнными свойствами [2 – 4]; 3) различного рода задачи на графах специального вида [5, 6].

В одной из постановок задачи аппроксимации требуется приблизить рассматриваемый граф графом, состоящим из k компонент связности [7]. В таком виде задача аппроксимации имеет много общего с задачей разбиения. В задаче разбиения графа требуется найти представление множества вершин исходного графа V в виде некоторого числа попарно непересекающихся подмножеств. Как правило, задачи разбиения графа содержат размер получаемого «разреза» множества вершин (*cut size*) как одну из составляющих целевой функции. В соответствии с этим существуют различные постановки задачи (min-cut, ratio cut, balanced cut и др.) [8 – 10].

В данной работе рассматривается не упоминавшаяся ранее в литературе задача аппроксимации взвешенного графа большой размерности графом меньшей размерности с целевой функцией на максимум разностей расстояний между вершинами исходного

графа и соответствующими им вершинами полученного малого графа. Рассматриваются два варианта задачи, приводятся их формулировки в терминах задач аппроксимации и разбиения графа. Для задач предлагаются алгоритмы решения, приводится их сравнение на данных из открытого доступа.

1. Постановка задачи и определения

Рассматривается простой связный ненаправленный взвешенный граф $G = (V, E, w)$, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин; $E = \{e_1, e_2, \ldots, e_m\}$ — множество рёбер; $w : E \to [0, \infty)$ — неотрицательная вещественная весовая функция на рёбрах. Введём ряд обозначений и определений.

Вес ребра из вершины v_i в вершину v_j будем обозначать $w(i, j)$ (отсутствие ребра — $w(i, j) = \infty$). *Длина пути* — сумма весов входящих в него рёбер. *Расстояние* m_{ij} из вершины v_i в вершину v_j — длина кратчайшего пути из v_i в v_j.

Рассматриваемые задачи аппроксимации состоят в сопоставлении исходному графу G аппроксимирующего графа $G_a = (V_a, E_a, w_a)$ меньшей размерности: $|V_a| = n_a$, где $1 \leqslant n_a \leqslant n$. Будем называть вершины $\{v_i^a : i = 1, \ldots, n_a\}$ аппроксимирующего графа *метавершинами*. Множество метавершин может быть подмножеством исходного множества вершин ($V_a \subset V$) или как частично, так и полностью не содержаться в V. Вес ребра между метавершинами v_i^a и v_j^a графа G_a будем обозначать $w_a(i, j)$, расстояние между этими же метавершинами — m_{ij}^a. Будем обозначать $v_{c(i)}^a$ метавершину графа G_a, аппроксимирующую вершину v_i графа G. *Погрешностью аппроксимации* будем называть величину $A_e = \max\limits_{i,j=1,\ldots,n} \left(|m_{ij} - m_{c(i)c(j)}^a| \right)$.

В работе рассматриваются две задачи аппроксимации: 1) уменьшение погрешности A_e при заданной размерности G_a и 2) уменьшение размерности G_a так, чтобы погрешность A_e не превосходила заданной величины.

Задача 1. Даны простой связный ненаправленный взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на рёбрах $w : E \to [0, \infty)$ и вещественное число $e_{\max} > 0$. Найти граф $G_a = (V_a, E_a, w_a)$ и функцию $c : \{1, \ldots, n\} \to \to \{1, \ldots, n_a\}$, такие, что подграфы $G_j(V_j)$, где $V_j = \{v_i \in V : c(i) = j\}$, $j = 1, \ldots, n_a$, связны, $A_e \leqslant e_{\max}$ и $|V_a| \to \min$.

Задача 2. Даны простой связный ненаправленный взвешенный граф $G = (V, E, w)$, $|V| = n$, с неотрицательной вещественной весовой функцией на рёбрах $w : E \to [0, \infty)$ и целое число k, $n > k > 0$. Найти граф $G_a = (V_a, E_a, w_a)$, $|V_a| = k$, и функцию соответствия $c : \{1 \ldots, n\} \to \{1, \ldots, k\}$, такие, что подграфы $G_j(V_j)$, где $V_j = \{v_i \in V : c(i) = j\}$, связны и $A_e \to \min$.

Обе рассматриваемые задачи являются NP-трудными. Эти задачи при некоторых допущениях могут быть сформулированы в терминах задачи разбиения графа. Введём обозначения: *диаметр множества вершин* $\mathrm{d}(V_i) = \max\limits_{j,k:v_j,v_k \in V_i} m_{jk}$; *радиус множества вершин* $\mathrm{r}(V_i) = \min\limits_{j:v_j \in V_i} \left(\max\limits_{k:v_k \in V_i} m_{jk} \right)$; вершину, на которой достигается $\mathrm{r}(V_i)$, будем называть *центром множества вершин* V_i и обозначать $\mathrm{c}(V_i)$.

Будем аппроксимировать вершины подмножеств разбиения V_i центрами этих подмножеств $v_{c_i} = \mathrm{c}(V_i)$. Веса рёбер в графе G_a между этими центрами будем полагать равными расстояниям между центрами в исходном графе G: $w_a(c_i, c_j) = m_{c_i c_j}$. Добавим для каждой вершины G_a петлю с весом, равным половине диаметра соответствующего множества разбиения: $w_a(c_i, c_i) = \mathrm{d}(V_i)/2$. Дополнительно будем требовать связность всех подграфов разбиения $G_i(V_i)$.

При выполнении всех этих условий для погрешности аппроксимации справедливо $d_{\max}/2 \leqslant A_e \leqslant r_{m_1} + r_{m_2}$. Здесь $d_{\max} = \max\limits_i \mathrm{d}(V_i)$, r_{m_1} — максимальный, а r_{m_2} — второй по величине радиусы множеств разбиения V_i. Действительно, в этом случае погрешность аппроксимации внутри подграфов не превосходит $d_{\max}/2 \leqslant r_{m_1}$, а погрешность аппроксимации между вершинами v_i и v_j из разных подграфов не превосходит суммы расстояний от этих вершин до центров соответствующих подграфов, т. е. $A_e \leqslant r_{m_1} + r_{m_2}$.

При такой оценке для A_e задачи разбиения графа, соответствующие поставленным задачам аппроксимации, выглядят следующим образом.

Задача 1*. Даны простой связный ненаправленный взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на рёбрах $w : E \to [0, \infty)$ и вещественное число $e_{\max} > 0$. Найти разбиение множества вершин графа на попарно непересекающиеся подмножества $V_1, V_2, \ldots, V_k$, такое, что все подграфы $G_i(V_i)$, порождённые этими множествами, связны, $r_{m_1} + r_{m_2} \leqslant e_{\max}$ и $k \to \min$.

Задача 2*. Даны простой связный ненаправленный взвешенный граф $G = (V, E, w)$, $|V| = n$, с неотрицательной вещественной весовой функцией на ребрах $w : E \to$ $\to [0, \infty)$ и целое число k, $n > k > 0$. Найти разбиение множества вершин графа на k попарно непересекающихся подмножеств $V_1, V_2, \ldots, V_k$, такое, что все подграфы $G_i(V_i)$, порождённые этими множествами, связны и $r_{m_1} + r_{m_2} \to \min$.

Предлагаются эвристические алгоритмы решения задач 1 и 2.

2. Алгоритм для задачи 1 ($A1$)

Алгоритм состоит в последовательном «удалении» вершин графа. Удаляемым вершинам ставится в соответствие в качестве метавершины одна из смежных вершин и выполняется пересчёт весов рёбер получившегося графа меньшей размерности.

Пусть на возможность удаления первой рассматривается вершина v_i степени m, смежная с вершинами v_{i_z}, $z = 1, \ldots, m$. Удаление v_i влечёт необходимость изменения весов рёбер только между смежными с ней вершинами. Пересчёт весов данных рёбер будем производить по формуле

$$w(i_j, i_l) = \begin{cases} w(i_j, i) + w(i, i_l), & \text{если } w(i_j, i) + w(i, i_l) < w(i_j, i_l), \\ w(i_j, i_l) & \text{иначе.} \end{cases}$$

После такого пересчёта погрешность аппроксимации между всеми неудалёнными вершинами останется равной нулю. Поэтому для минимизации общей погрешности аппроксимации необходимо поставить в соответствие удалённой вершине v_i такую метавершину $v_{c(i)}$, которая сделает как можно меньшим максимум абсолютной разности расстояний между v_i, $v_{c(i)}$ и вершинами исходного графа: $\max\limits_j \left(\left|m_{ij} - m_{c(i)j}\right|\right) \to \min$. В исходном графе расстояния от v_i до любой вершины графа v_j определяются формулой $m_{ij} = \min\limits_z (w(i, i_z) + m_{i_z j})$; после удаления v_i расстояния между неудалёнными вершинами $m_{i_z j}$ не изменились. В качестве $v_{c(i)}$ будем выбирать ближайшую смежную с v_i вершину $v_{c(i)} = v_{i_j} : w(i, i_j) = \min\limits_z w(i, i_z)$, при этом погрешность аппроксимации увеличится не более чем на вес ребра $w(i, i_j)$.

В задаче 1 требуется, чтобы погрешность аппроксимации не превосходила некоторого заданного числа $A_e \leqslant e_{\max}$. Для достижения этого необходимо хранить информацию о погрешности, вносимой каждой из оставшихся вершин v_l. Пусть A_e^l — максимально возможная погрешность аппроксимации между вершинами, аппроксимируемыми вершиной v_l; сначала все $A_e^l = 0$. Удаление вершины v_i изменяет погрешность

аппроксимирующей её вершины $v_{c(i)}$ по правилу $A_e^{c(i)} = A_e^{c(i)} + A_e^i + w\,(i, c(i))$. В силу сохранения расстояний между оставшимися вершинами для выполнения $A_e \leqslant e_{\max}$ необходимо, чтобы для всех пар оставшихся вершин (v_l, v_j) выполнялось $A_e^l + A_e^j \leqslant$ $\leqslant e_{\max}$, это справедливо, как только $A_e^j \leqslant e_{\max}/2$ для всех j. Согласно этому, вершину v_i можно удалять, когда для аппроксимирующей её вершины $v_{c(i)}$ выполняется $A_e^{c(i)} + A_e^i + w\,(i, c(i)) \leqslant e_{\max}/2$.

Рассмотрение и удаление вершин может производиться различными способами. Одной из очевидных стратегий является удаление вершин в порядке роста добавляемой этими вершинами погрешности. В п. 4 приводятся результаты тестирования алгоритма для реализации, в которой вершины рассматриваются в порядке роста их степеней. Такая стратегия показала наилучшие результаты на тестовых графах.

Полученная таким образом аппроксимация не гарантирует связности подграфов $G_j\,(V_j)$, $V_j = \{v_i \in V : c(i) = j\}$, исходного графа, вершины которых аппроксимируются одной метавершиной. Существует правило переопределения функции c, гарантирующее связность таких подграфов и не увеличивающее погрешность аппроксимации.

Пусть для всех множеств V_j найдены радиусы $\mathrm{r}(V_j)$ и центры $\mathrm{c}(V_j)$ (быстрый поиск этих величин можно осуществить, используя алгоритм [11]). Если для вершины $v_i \in V_k$ отсутствует путь до центра $\mathrm{c}(V_k)$ в графе $G_k\,(V_k)$, то в пути от v_i до $\mathrm{c}(V_k)$ в исходном графе G существует вершина $v_j \in V_l$. Если принадлежность вершины v_i множеству V_l не увеличивает $\mathrm{r}(V_l)$ и у вершины v_i принадлежность множеству не изменялась, полагаем $c(i) = l$. В противном случае все вершины v_s пути от v_i до v_j в исходном графе G делаем принадлежащими множеству V_k, полагая $c(s) = k$. После каждого изменения принадлежности множеству какой-либо из вершин центры и радиусы изменившихся множеств V_k, V_l пересчитываются и процесс повторяется снова до тех пор, пока не останется несвязных подграфов. После этого полагаем $w\,(\mathrm{c}\,(V_j)\,, \mathrm{c}\,(V_j)) = \mathrm{r}\,(V_j)$, $c(i) = \mathrm{c}\,(V_j)$ для всех i, таких, что $c(i) = j$.

В худшем случае, когда для полного графа выполняется «удаление» $n-2$ вершин, сложность алгоритма составляет $\mathrm{O}(n^3)$.

3. Алгоритм для задачи 2 ($A2$)

Для решения задачи 2 предлагается эвристический итерационный алгоритм, в основе которого лежит алгоритм $A1$. Итеративно ищется такое значение величины $e_{\max}$ (и аппроксимация при этом значении), при котором решение задачи 1 алгоритмом $A1$ обладает свойством $|V_a| \leqslant k$, где k — ограничение из задачи 2. Для поиска такого $e_{\max}$ используется интерполяция многочленом Лагранжа и предположение о зависимости $e_{\max} = f\,(|V_a|)$. При этом считаем, что $f\,(|V|) = 0$, а для получения ещё одной точки интерполяции полагаем $e_{\max}^1 = \overline{w}(n/k)$, где $\overline{w}$ — средний вес ребра в графе.

Если в результате такого поиска аппроксимирующий граф содержит вершин меньше нужного ($|V_a| < k$), то полагаем $c(i) = \mathrm{c}\,(V_j)$ для всех i, $c(i) = j$, а $k - |V_a|$ вершин v_i с максимальным удалением от центров своих множеств $\mathrm{c}(V_j)$ переносим в отдельные множества, полагая $c(i) = i$.

4. Результаты

В качестве тестовых данных использовались взвешенные графы дорожных сетей США из открытого доступа [12]. Из них были получены связные подграфы размерностью от 10^3 до $5 \cdot 10^3$, по 10 для каждой размерности. Характеристики тестовых графов представлены в табл. 1.

Таблица 1

Характеристики тестовых графов

Группа графов	Среднее кол-во вершин	Средняя степень вершин	Средний диаметр	Среднее расстояние	Кол-во графов
$G1$	10^3	2,48	$3{,}64 \cdot 10^5$	$1{,}39 \cdot 10^5$	10
$G2$	$2 \cdot 10^3$	2,6	$3{,}17 \cdot 10^5$	$1{,}07 \cdot 10^5$	
$G3$	$3 \cdot 10^3$	2,62	$5{,}63 \cdot 10^5$	$1{,}9 \cdot 10^5$	
$G4$	$4 \cdot 10^3$	2,68	$4{,}09 \cdot 10^5$	$1{,}45 \cdot 10^5$	
$G5$	$5 \cdot 10^3$	2,66	$4{,}88 \cdot 10^5$	$1{,}69 \cdot 10^5$	

Для сравнения с алгоритмом $A2$ использовалась модификация эвристического алгоритма Fiduccia — Mattheyses биразбиения графа ($A2^*$ [13], сложность одной итерации $O(|E|)$), решающая задачу в постановке 2*. Описание сущности процесса разбиения данным методом приводится в [14]. Здесь укажем лишь особенности адаптации алгоритма к рассматриваемой задаче.

В задаче требуется минимизировать сумму максимального r_{m_1} и второго по величине r_{m_2} радиусов множества вершин получившихся подграфов разбиения — $r_{m_1} + r_{m_2} \to \min$. После начального разбиения выбор пары подграфов (G_i, G_j) для его улучшения производится так, чтобы радиус множества вершин G_i был равен r_{m_1} или r_{m_2}. При этом G_j выбирается как подграф, имеющий минимальный радиус множества вершин среди подграфов, связанных с G_i хотя бы одним ребром и изменённых со времени последнего совместного рассмотрения с G_i. Число вершин — претендентов на перемещение — ограничивается требованием сохранения связности подграфов после перемещения. Улучшение пары (G_i, G_j) останавливается, если $\max\limits_i g_i < 0$ (g_i — разница целевых функций до и после возможного перемещения v_i) или радиусы множеств вершин обоих подграфов меньше r_{m_2}.

После того как дальнейшее улучшение разбиения становится невозможным, для каждого подграфа G_i находится центр v_{c_i} и диаметр $\mathrm{d}(G_i)$. Функция соответствия c задается как $c(j) = i$ для всех j, таких, что $v_j \in V_i$. Подграф G_a строится из k вершин, веса его рёбер задаются формулами $w_a(i,j) = m\,(c_i, c_j)$ и $w_a(i,i) = \mathrm{d}\,(G_i)\,/2$.

Результаты экспериментов приведены в табл. 2 и 3.

Заключение

Рассмотрены две задачи аппроксимации на уменьшение размерности графа при ограничениях на величину изменения расстояний между исходными вершинами и вершинами аппроксимирующего графа. Полезность постановки этих задач обнаруживается в практических приложениях: транспортной логистике, мониторинге вычислительных сетей и т. д. Полученный малый аппроксимирующий граф может использоваться при решении NP-трудных задач на графах, позволяя: а) решать задачи быстрее, используя методы для графов большой размерности, б) получать более точные результаты, применяя более вычислительно сложные алгоритмы.

Рассматриваемые задачи аппроксимации являются частным случаем задачи разбиения графа. Функцию c, ставящую в соответствие вершинам исходного графа вершины аппроксимирующего графа, можно рассматривать как способ распределения вершин по множествам разбиения V_i. Такое отношение между задачами аппроксимации и разбиения графа позволяет использовать методы решения одной задачи при решении другой, и наоборот.

Таблица 2

Результаты решения задачи 1 алгоритмом $A1$

Группа графов	$\overline{V_n}$	$\overline{A_e}$ фактическая	$\overline{\text{Время, с}}$
$e_{\max} = 10^3$			
$G1$	897	975	1,4
$G2$	$1{,}8 \cdot 10^3$	987	5,8
$G3$	$2{,}7 \cdot 10^3$	991	14
$G4$	$3{,}5 \cdot 10^3$	995	25
$G5$	$4{,}4 \cdot 10^3$	996	41
$e_{\max} = 2 \cdot 10^3$			
$G1$	765	$2 \cdot 10^3$	1,4
$G2$	$1{,}4 \cdot 10^3$	$2 \cdot 10^3$	5,8
$G3$	$2{,}2 \cdot 10^3$	$2 \cdot 10^3$	14
$G4$	$2{,}8 \cdot 10^3$	$2 \cdot 10^3$	25
$G5$	$3{,}4 \cdot 10^3$	$2 \cdot 10^3$	41
$e_{\max} = 5 \cdot 10^3$			
$G1$	558	$4{,}9 \cdot 10^3$	1,3
$G2$	945	$4{,}9 \cdot 10^3$	5,4
$G3$	$1{,}5 \cdot 10^3$	$4{,}9 \cdot 10^3$	13
$G4$	$1{,}8 \cdot 10^3$	$5 \cdot 10^3$	24
$G5$	$2{,}2 \cdot 10^3$	$5 \cdot 10^3$	38

Таблица 3

Результаты решения задачи 2 (Δ_1 — отношение A_e к диаметру графа, Δ_2 — отношение A_e к среднему расстоянию в графе)

Группа графов	$A2$				$A2^*$			
	$\overline{A_e}$	$\overline{\Delta_1}$	$\overline{\Delta_2}$	$\overline{\text{Время, с}}$	$\overline{A_e}$	$\overline{\Delta_1}$	$\overline{\Delta_2}$	$\overline{\text{Время, с}}$
$k = \lfloor 0{,}1 \cdot \lvert V \rvert \rfloor$								
$G1$	$4{,}6 \cdot 10^4$	0,14	0,35	0,52	$1{,}3 \cdot 10^5$	0,35	0,92	1,7
$G2$	$3 \cdot 10^4$	0,1	0,29	2,3	$9{,}6 \cdot 10^4$	0,31	0,91	8,4
$G3$	$4{,}3 \cdot 10^4$	0,09	0,26	5,4	$1{,}8 \cdot 10^5$	0,32	0,95	23
$G4$	$3{,}2 \cdot 10^4$	0,08	0,24	9,9	$1{,}4 \cdot 10^5$	0,33	0,97	44
$G5$	$2{,}8 \cdot 10^4$	0,06	0,19	16	$1{,}7 \cdot 10^5$	0,32	0,95	81
$k = \lfloor 0{,}2 \cdot \lvert V \rvert \rfloor$								
$G1$	$2{,}4 \cdot 10^4$	0,08	0,2	0,83	$1{,}2 \cdot 10^5$	0,31	0,82	1,9
$G2$	$1{,}7 \cdot 10^4$	0,06	0,16	3,6	$8{,}1 \cdot 10^4$	0,26	0,76	11
$G3$	$2{,}3 \cdot 10^4$	0,05	0,13	8,5	$1{,}6 \cdot 10^5$	0,28	0,83	30
$G4$	$1{,}6 \cdot 10^4$	0,04	0,12	15	$1{,}2 \cdot 10^5$	0,29	0,85	62
$G5$	$1{,}6 \cdot 10^4$	0,04	0,11	26	$1{,}5 \cdot 10^5$	0,28	0,82	112
$k = \lfloor 0{,}5 \cdot \lvert V \rvert \rfloor$								
$G1$	$7 \cdot 10^3$	0,02	0,06	1,3	$8 \cdot 10^4$	0,23	0,58	2,7
$G2$	$4{,}9 \cdot 10^3$	0,02	0,05	5,5	$5{,}8 \cdot 10^4$	0,18	0,53	17
$G3$	$5{,}9 \cdot 10^3$	0,01	0,04	13	$1 \cdot 10^5$	0,19	0,55	50
$G4$	$4{,}3 \cdot 10^3$	0,01	0,03	24	$8{,}7 \cdot 10^4$	0,19	0,55	108
$G5$	$4{,}3 \cdot 10^3$	0,01	0,03	39	$8{,}4 \cdot 10^4$	0,18	0,53	199

Необходимо отметить, что в практической деятельности связность подграфов $G_j\,(V_j)$, приводимая в постановках рассматриваемых задач, часто не является необходимой. Это связано с тем, что граф разбивается лишь логически, связность его не нарушается. С исключением этого условия в приведённых алгоритмах отпадает надобность в процедуре изменения принадлежности вершин множествам V_j и вычислении $\mathrm{r}(V_j)$ и $\mathrm{c}(V_j)$, что значительно уменьшает время аппроксимации данными методами.

ЛИТЕРАТУРА

1. *Bonneau J., Anderson J., et al.* Eight friends are enough: social graph approximation via public listings // Proc. Second ACM EuroSys Workshop on Social Network Systems. Nuremberg, 2009. P. 13–18.
2. *Koutis I., Levin A., et al.* Improved spectral sparsification and numerical algorithms for SDD matrices // Proc. STACS. Paris, 2012. P. 266–277.
3. *Long B., Xiaoyun X., et al.* Community learning by graph approximation // Proc. Seventh IEEE Intern. Conf. on Data Mining. Omaha, 2007. P. 232–241.
4. *Jacob J., Jentsch M., et al.* Detecting hierarchical structure in molecular characteristics of disease using transitive approximations of directed graphs // Bioinformatics. 2008. V. 24(7). P. 995–1001.
5. *Kuhn F., Moscibroda T., et al.* Unit disk graph approximation // Proc. Joint Workshop on Foundations of Mobile Computing. Philadelphia, 2004. P. 17–23.
6. *Fertin G., Hermelin D., et al.* Common structured patterns in linear graphs: approximation and combinatorics // Proc. 18th Annual Conf. on Combinatorial Pattern Matching. Berlin, 2007. P. 241–252.
7. *Ильев В. П., Ильева С. Д. и др.* Приближенные алгоритмы для задач аппроксимации графов // Дискретный анализ и исследование операций. 2007. Т. 18(1). С. 41–60.
8. *Kernighan B. and Lin S.* An efficient heuristic procedure for partitioning graphs // The Bell System Techn. J. 1970. V. 40(1). P. 291–307.
9. *Hagen L. and Kahng A.* New spectral methods for ratio cut partitioning and clustering // IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems. 1992. V. 11(9). P. 1074–1085.
10. *Andreev K. and Racke H.* Balanced graph partitioning // Proc. sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures. Barcelona, 2004. P. 120–124.
11. *Ураков А. Р., Тимеряев Т. В.* Алгоритмы быстрого поиска для двух задач о метрических характеристиках взвешенных графов // Управление большими системами. 2013. Вып. 42. С. 153–172.
12. `http://www.dis.uniroma1.it/challenge9/download.shtml` — 9th DIMACS Implementation Challenge — Shortest Paths (дата обращения: май 2013).
13. *Fiduccia C. and Mattheyses R.* A linear time heuristic for improving network partitions // DAC'82. Proc. 19th Design Automation Conf. Las Vegas, 1982. P. 175–181.
14. *Ураков А. Р., Тимеряев Т. В.* Многоуровневый алгоритм разбиения графов по критерию средней длины // Информационные технологии. 2012. № 4. С. 22–25.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 004.43, 004.056

О КРИПТОГРАФИЧЕСКОМ РАСШИРЕНИИ И ЕГО РЕАЛИЗАЦИИ ДЛЯ РУССКОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Г. П. Агибалов, В. Б. Липский, И. А. Панкратова

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: agibalov@isc.tsu.ru, lipsky@mail.tsu.ru, pank@isc.tsu.ru

Представлено расширение русского языка программирования ЛЯПАС, получившее название ЛЯПАС-Т и заключающееся в увеличении длины операндов и расширении множества элементарных операций над ними. Необходимость в нём продиктована, в первую очередь, потребностями страны в доверенных и эффективных программной и аппаратной реализациях современных криптографических алгоритмов в безопасных компьютерных системах логического управления критически важными объектами, такими, как космические системы, энергетические установки, ядерное оружие, подводные лодки, беспилотники и т. п. Представлены также компилятор ЛЯПАСа-Т, генерирующий его загрузочный модуль для операционной системы Linux, и проекты процессора, реализующего ЛЯПАС-Т аппаратно, и препроцессора, конвертирующего программы на ЛЯПАСе-Т в исполняемый код процессора. Сообщается о процессоре для подмножества ЛЯПАСа-Т без подпрограмм, операций над комплексами и длинных операндов, описанном на VHDL, протестированном средствами компьютерного моделирования и реализованном на ПЛИС с помощью системы автоматизированного проектирования.

Ключевые слова: *Русский язык программирования, ЛЯПАС-Т, компилятор, препроцессор, процессор, аппаратная реализация.*

Введение

Здесь под Русским языком программирования подразумевается алгоритмический язык ЛЯПАС, разработанный в начале 1960-х годов в Томском государственном университете под руководством А. Д. Закревского и предназначенный для представления логико-комбинаторных алгоритмов решения задач прикладной дискретной математики, встречающихся в синтезе дискретных автоматов [1, 2]. Имя Русского языка программирования ему дали американские учёные [3]. До 1990-х годов ЛЯПАС интенсивно использовался в Советском Союзе [2], США [4], Германии, Польше, Чехословакии и многих других странах. В настоящее время ЛЯПАС успешно возрождается силами кафедры защиты информации и криптографии Томского государственного университета, главным образом, с целью разработки доверенного системного и прикладного программного обеспечения для автоматизированного проектирования безопасных компьютерных систем логического управления и для безопасной и эффективной реализации криптографических алгоритмов [5]. Среди многочисленных языков програм-

мирования, известных сегодня, ЛЯПАС предстаёт как наиболее подходящий для этих целей.

Вместе с тем есть один существенный и, возможно, единственный недостаток ЛЯПАСа — отсутствие в нём ряда элементарных операций, которые широко используются в современных криптографических алгоритмах: для арифметики длинных чисел, вычислений в многомерных пространствах над конечными полями и кольцами, решения комбинаторных задач над большими множествами и др. Кстати сказать, этот недостаток присущ всем современным языкам программирования, включая и те, что моложе ЛЯПАСа. В некоторых из них он преодолевается путём написания классов длинных чисел, больших дискретных функций и т. п. Что касается ЛЯПАСа, этот его недостаток более эффективно преодолевается расширением самого языка путём распространения элементарных операций в ЛЯПАСе на логические комплексы, которые, как известно, допускают арифметическую интерпретацию, и добавлением к нему некоторых новых элементарных операций, определённых над переменными и логическими комплексами. Последняя версия ЛЯПАСа — язык ЛЯПАС-М [6, 7], подвергнутый предварительно некоторой ревизии и затем расширенный (exTended) указанным образом, — называется ЛЯПАС-Т.

Ревизия ЛЯПАСа-М касается символики языка и арифметических операций умножения и деления целых чисел. Её результат назван vЛЯПАСом (от vLYaPAS, или reVised LYaPAS). В нём вместо больших русских букв используются большие латинские буквы, символы некоторых операций заменены другими, более подходящими знаками, и операции умножения и деления определены с сохранением переполнения и остатка соответственно. Для хранения последних в язык введена специальная переменная Z.

Цель данной работы — представить информацию о некоторых из первых результатов, полученных в процессе возрождения ЛЯПАСа, а именно: о расширении ЛЯПАС-Т, обусловленном, главным образом, требованиями криптографических алгоритмов; о компиляторе ЛЯПАСа-Т, генерирующем код в формате исполняемого файла операционной системы (ОС) Linux; о процессоре, реализующем программы на ЛЯПАСе-Т аппаратно; о препроцессоре, транслирующем эти программы в исполняемый код процессора; о воплощении процессора в ПЛИС для подмножества vЛЯПАСа.

Начнём изложение с краткого обзора языка vЛЯПАС.

1. vЛЯПАС

Программа на vЛЯПАСе представляет собой последовательность предложений, каждое из которых (кроме, возможно, первого) начинается с §s, где s — целое неотрицательное число, и которые, в свою очередь, являются последовательностями операций над операндами языка.

1.1. Операнды

Операндами в vЛЯПАСе являются константы, переменные, комплексы и элементы комплексов. Они используются для представления неотрицательных целых чисел, булевых векторов, символов Unicode и последовательностей из них. Компоненты в булевом векторе нумеруются, начиная с 0 в направлении справа налево. В vЛЯПАСе неотрицательные целые ограничиваются числом $2^{32} - 1$, а длина булева вектора — числом 32. Булев вектор длины 32 называется словом и рассматривается также как неотрицательное целое, представленное в двоичном позиционном коде. Булев вектор любой длины $n \geqslant 1$ с одной единичной компонентой называется далее единичным вектором.

В vЛЯПАСе имеются натуральные, единичные и символьные константы. Натуральные константы записываются как десятичные, шестнадцатеричные, восьмеричные и двоичные числа. Единичная константа — это единичный вектор. Она обозначается Ii, если номер компоненты 1 в ней равен i. Символьная константа — это последовательность символов Unicode. В ней символы $'$ и \ записываются как пары \$'$ и \\ соответственно.

Переменные в vЛЯПАСе принимают значения булевых векторов длины 32. Их количество равно 27. Они обозначаются буквами a, b, ..., z, Z и, как сказано выше, Z используется в специальных целях. Кроме того, есть ещё виртуальная переменная, называемая собственной или внутренней переменной языка. В отличие от остальных, реальных, переменных, она не участвует в записи программ на ЛЯПАСе, но появляется в них неявно как результат любой элементарной операции и может быть использована любой последующей операцией в программе. Для удобства изложения эту переменную принято называть τ.

Комплекс есть конечное линейно упорядоченное множество элементов, которые в символьном комплексе суть коды символов (булевы векторы длины 8, или байты), а в логическом — булевы векторы длины 32, или слова. Каждый комплекс имеет уникальный номер из ряда 0, 1, 2, ... Логический или символьный комплекс, имеющий номер i, обозначается Li или Fi соответственно. Действительное и максимально возможное количества элементов в комплексе являются параметрами комплекса и называются его мощностью и ёмкостью соответственно.

Далее все эти понятия вводятся формально:

— переменная υ ::=a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z;
— особая переменная Z;
— десятичная цифра ϱ ::= 0|1|2|3|4|5|6|7|8|9, десятичая константа ∂ ::= $\varrho\{\varrho\}$;
— шестнадцатеричная цифра η ::= ϱ|A|B|C|D|E|F, шестнадцатеричная константа $\hbar$ ::= $\eta\{\eta\}$h;
— восьмеричная цифра ω ::= 0|1|2|3|4|5|6|7, восьмеричная константа ϖ ::= $\omega\{\omega\}$o;
— двоичная цифра ε ::= 0|1, двоичная константа β ::= $\varepsilon\{\varepsilon\}$b ;
— натуральная константа ι ::= $\partial|\hbar|\varpi|\beta$;
— символ σ ::= любой символ Unicode, символьная константа $\Sigma ::='\ \sigma\{\sigma\}'$;
— единичная константа $\mathcal{I}$ ::= I∂|Iυ; константа ::= $\iota|\Sigma|\mathcal{I}$;
— символьный комплекс $\mathcal{F}$::= F∂, логический комплекс $\mathcal{L}$ ::= L∂, комплекс $\varkappa$ ::= $\mathcal{F}|\mathcal{L}$;
— индекс $\mathcal{J}$ ::= $.\partial|\upsilon|(\upsilon+\partial)|(\upsilon-\partial)$; элемент комплекса ::= $\varkappa\mathcal{J}$;
— Q∂ ::= мощность комплекса с номером ∂, S∂ ::= ёмкость комплекса с номером ∂;
— значение натуральной константы ::= неотрицательное целое|булев вектор;
— значение символьной константы ::= строка символов;
— значение переменной ::= неотрицательное целое|булев вектор;
— значение элемента логического комплекса ::= неотрицательное целое|булев вектор;
— значение элемента символьного комплекса ::= неотрицательное целое|булев вектор|символ.

В отличие от других языков программирования, типы значений в vЛЯПАСе не фиксированы. Тип значения (целое или вектор) для константы, переменной и элемента комплекса определяется типом операции (арифметической или логической соответственно), которая применяется к этому операнду.

1.2. Элементарные операции

Передача значения

Пусть, как обычно, τ есть собственная (внутренняя) переменная ЛЯПАСа, α — произвольные переменная или элемент комплекса, γ — переменная, элемент комплекса или константа, χ — значение на выходе генератора псевдослучайных чисел в компьютере (PRNG), ζ — начальное состояние PRNG, θ — значение на выходе таймера в компьютере. Тогда:

$\mathrm{O}\alpha$ — присвоение наименьшего значения (нулей): $\alpha := 00...0$;
$^{-}\alpha$ — присвоение наибольшего значения (единиц): $\alpha := 11...1$;
$\Rightarrow \alpha$ — присвоение τ: $\alpha := \tau$;
γ — присвоение γ: $\tau := \gamma$;
$\Leftrightarrow (\upsilon_1\upsilon_2)$ — обмен значениями переменных υ_1 и υ_2;
$\Leftrightarrow (\varkappa\upsilon_1\upsilon_2)$, или $\Leftrightarrow (\varkappa\upsilon\partial)$, $\Leftrightarrow (\varkappa.\partial\upsilon)$, или $\Leftrightarrow (\varkappa.\partial_1.\partial_2)$ — обмен значениями элементов комплекса $\varkappa\upsilon_1$ и $\varkappa\upsilon_2$, или $\varkappa\upsilon$ и $\varkappa.\partial$, или $\varkappa.\partial_1$ и $\varkappa.\partial_2$ соответственно;
X означает $\tau := \chi$; $\Rightarrow$X означает $\zeta := \tau$; T означает $\tau := \theta$.

Логические и арифметические операции

! — вычисление номера правой единицы: $\tau := !\tau$;
$\neg$ — отрицание $\tau := \neg\tau$;
% — вычисление веса булева вектора: $\tau := \%\tau$;
$\vee\gamma$ — дизъюнкция: $\tau := \tau \vee \gamma$;
$\&\gamma$ — конъюнкция: $\tau := \tau\&\gamma$;
$\oplus\gamma$ — сложение по модулю 2: $\tau := \tau \oplus \gamma$;
$< \gamma$ — левый сдвиг: $\tau := \tau < \gamma$;
$> \gamma$ — правый сдвиг $\tau := \tau > \gamma$;
$+\gamma$ — сложение по модулю 2^{32}: $\tau := \tau + \gamma$;
$-\gamma$ — вычитание по модулю 2^{32}: $\tau := \tau - \gamma$;
$*\gamma$ — умножение по модулю 2^{32}: $\tau := \tau * \gamma$, Z := переполнение;
$:\gamma$ — деление двойного числа: $\tau :=$ частное от деления Zτ на γ, Z := остаток;
$/\gamma$ — частное целочисленного деления: $\tau :=$ частное от деления τ на γ, Z := остаток;
$;\gamma$ — остаток целочисленного деления: $\tau :=$ остаток от деления τ на γ, Z := частное;
$\triangle\alpha$ — увеличение на 1: $\tau := \alpha := \alpha + 1$;
$\bigtriangledown\alpha$ — уменьшение на 1: $\tau := \alpha := \alpha - 1$.

1.3. Операции перехода

$\to \partial$ — безусловный переход к параграфу ∂;
$\hookrightarrow \partial$ — переход к параграфу ∂ по условию $\tau = 0$;
$\mapsto \partial$ — переход к параграфу ∂ по условию $\tau \neq 0$;
$\uparrow (\gamma_1 \diamond \gamma_2)\partial$ — переход к параграфу ∂ по отношению $\diamond \in \{=, \neq, <, >, \leqslant, \geqslant\}$;
$\rightarrowtail \partial$ — уход к параграфу ∂ с возвратом;
$\leftarrowtail$ — возврат к точке, следующей за точкой ухода $\rightarrowtail \partial$;
$\uparrow (\gamma)\partial$ — переход к параграфу ∂ по времени γ;
$\uparrow$X$\partial\alpha_1\alpha_2$ — перечисление единиц: если $\alpha_1 = 0$, то $\alpha_2 := 0$ и $\to \partial$, в противном случае правая 1 в α_1 заменяется на 0, её номер присваивается α_2 и выполняется следующая операция;
{assembly program} — ассемблерная вставка: выполняется программа на языке Ассемблера, указанная между { и }.

1.4. Операции над комплексами

Пусть ξ есть ∂ или υ; тогда:

@ + $\varkappa(\xi)$ — образование (создание) комплекса $\varkappa$ ёмкости ξ и мощности 0;

@ − $\varkappa$ — уничтожение комплекса $\varkappa$;

@%$\varkappa$ — сокращение ёмкости комплекса до его мощности;

O$\varkappa$ – очистка комплекса (в пределах его мощности): $\varkappa ::= 00\ldots0$;

@′$\sigma_1\sigma_2...\sigma'_m > \mathcal{F}$ — символы $\sigma_1, \sigma_2, ..., \sigma_m$ добавляются к символьному комплексу $\mathcal{F}$;

@ > $\varkappa\xi$ — вставка элемента: значение τ вставляется в комплекс $\varkappa$ перед ξ-м элементом (в отсутствие ξ — после последнего элемента);

@ < $\varkappa\xi$ — удаление элемента: ξ-й элемент (в отсутствие ξ — последний элемент) комплекса $\varkappa$ перемещается в τ, мощность комплекса уменьшается на 1;

@#$\varkappa_1\varkappa_2(\xi_1, \xi_2, \xi_3)$ — ξ_1 элементов комплекса $\varkappa_1$, начиная с ξ_2-го элемента (в отсутствие ξ_2 — первые ξ_1 элементов $\varkappa_1$), копируются в $\varkappa_2$, начиная с ξ_3-го элемента (в отсутствие ξ_3 — в конец $\varkappa_2$), мощности $\varkappa_1, \varkappa_2$ не изменяются.

Впредь комплекс, образованный с фиксированной ёмкостью (∂), называется статическим, а комплекс с переменной ёмкостью (υ) — динамическим.

Заметим также, что операция объявления части комплекса самостоятельным комплексом, существующая в ЛЯПАСе, не включена в vЛЯПАС по причине её потенциальной опасности.

1.5. Операции ввода-вывода

/$\mathcal{F}$ >C — вывод символьного комплекса $\mathcal{F}$ на консоль;

/′ς' >C — вывод символьной константы ς на консоль;

/$\mathcal{F}$ <C — ввод символьной константы с клавиатуры: вводимые символы добавляются к символьному комплексу $\mathcal{F}$, мощность комплекса увеличивается.

2. Расширение vЛЯПАСа

2.1. Длинная арифметика

Натуральные константы в расширении ЛЯПАС-Т являются целыми из множества $\{0, 1, \ldots, 2^n - 1\}$, где n кратно 32 и зависит от фактической реализации ЛЯПАСа-Т. В настоящее время значение $n = 2^{14}$ вполне приемлемо для криптографических приложений.

Обозначая число 2^{32} символом δ, любую натуральную константу c можно выразить в виде

$$c = c_0 + c_1\delta + c_2\delta^2 + \ldots + c_{r-1}\delta^{r-1} \tag{1}$$

для некоторых $r > 0$ и $c_i \in \Omega = \{0, 1, \ldots, 2^{32} - 1\}$, $i = 0, 1, \ldots, r - 1$. В стандартном двоичном представлении элементы множества Ω являются словами — булевыми векторами длины 32. Поэтому в ЛЯПАСе-Т последовательность $c_0, c_1, \ldots, c_{r-1}$ представляется логическим комплексом мощности r с c_i в качестве i-го элемента.

Все операции, определённые в vЛЯПАСе над переменными, в ЛЯПАСе-Т могут применяться к логическим комплексам. В случае арифметической операции последовательность элементов комплекса рассматривается как натуральная константа c, заданная формулой (1). Различные операнды для одной и той же арифметической операции могут иметь различные длины и типы (один из них — переменная, другой — комплекс). В случае логической операции значение комплекса рассматривается как булев вектор, являющийся конкатенацией элементов комплекса. Логические комплексы мощности $n/32$ со значениями единичных векторов являются единичными константами в ЛЯПАСе-Т.

2.2. Множественность собственной переменной

Таким образом, в отличие от ЛЯПАСа, в ЛЯПАСе-Т есть два типа операндов для элементарных операций: переменные длины в одно слово и логические комплексы различных длин — от одного до $n/32$ слов. Соответственно этому, в ЛЯПАСе-Т имеются и два типа собственной переменной — простая и комплексная. Первая — из ЛЯПАСа, имеет длину одного слова. Она может принимать значение любой переменной языка. В любой реализации ЛЯПАСа-Т, программной или аппаратной, она представляется в регистре процессора. Собственные переменные 2-го типа имеют длины логических комплексов и могут принимать значения последних. В аппаратной реализации ЛЯПАСа-Т все они могут представляться в одном и том же регистре максимальной возможной длины — n разрядов. В программной реализации ЛЯПАСа-Т, из-за отсутствия такого регистра, роль комплексной собственной переменной на время выполнения цепочки операций, начинающейся с обращения к логическому комплексу, возлагается непосредственно на этот комплекс, и хранится она по месту его расположения в памяти компьютера.

В дальнейшем изложении, там, где это не вызывает двусмысленности, собственная переменная любого типа, будь то простая или комплексная, обозначается (как в ЛЯПАСе) буквой τ и называется соответственно простой или комплексной τ.

2.3. Дополнительные операции

В дополнение к операциям в vЛЯПАСе расширение ЛЯПАС-Т содержит некоторые новые логические операции, используемые в записи криптографических алгоритмов, включая следующие, где $\lambda ::= \upsilon|\mathcal{L}|\varkappa\upsilon|\varkappa.\partial|\partial$:

1) $\Updownarrow \mathcal{L}$ — перестановка: компоненты булева вектора τ переставляются в соответствии с их порядковыми номерами, указанными в элементах логического комплекса $\mathcal{L}$;

2) $\_(\xi_1, \xi_2)$ — проекция: выбирается часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_1, ξ_2);

3) $\Uparrow \xi_1\lambda(\xi_2, \xi_3)$ — вставка: часть булева вектора λ с компонентами, имеющими номера в интервале (ξ_2, ξ_3), вставляется в τ перед ξ_1-й компонентой (в отсутствие ξ_3 вставляется правая часть булева вектора λ длины ξ_2, в отсутствие (ξ_2, ξ_3) — весь булев вектор λ);

4) $\Downarrow (\xi_1, \xi_2)$ — редукция: часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_1, ξ_2), вычёркивается из вектора;

5) $|\,\lambda$ — конкатенация: булев вектор λ присоединяется к τ;

6) $\ll \xi_1(\xi_2, \xi_3)$ или $\gg \xi_1(\xi_2, \xi_3)$ — левый или правый циклические сдвиги: часть булева вектора τ с компонентами, имеющими номера в интервале (ξ_2, ξ_3), циклически сдвигается на ξ_1 бит влево или вправо соответственно (в отсутствие ξ_3 сдвигается правая часть булева вектора τ длины ξ_2, в отсутствие (ξ_2, ξ_3) — весь булев вектор τ);

7) $\geqslant \varkappa$ — максимальный элемент комплекса $\varkappa$ помещается в τ, его номер — в Z;

8) $\leqslant \varkappa$ — минимальный элемент комплекса $\varkappa$ помещается в τ, его номер — в Z.

Что касается арифметических операций по модулю N (для некоторого натурального N), таких, как сложение и вычитание (mod N), умножение (mod N), возведение в степень (mod N) и другие, широко используемые в криптографии, фактически нет никакой реальной возможности для включения их в список элементарных операций ЛЯПАСа-Т из-за существования великого множества алгоритмов их выполнения, имеющих разные эффективности в различных случаях. Вместо этого решено эти алгоритмы реализовывать по мере надобности и возможности на ЛЯПАСе-Т и (или) на

языке Ассемблера и включать их в библиотеку для использования в программах на ЛЯПАСе-Т в качестве подпрограмм.

3. Компилятор ЛЯПАСа-Т

3.1. Что это такое?

Далее для краткости любая программа на ЛЯПАСе-Т и её подпрограммы называются L-программой и L-подпрограммами соответственно.

Для выполнения L-программы компьютером она должна быть подана в качестве параметра компилятору, который преобразует её в загрузочный модуль (исполняемый код) для ОС Linux. Компилятор запускается по команде последней
`>$ lc <prog>.l`,
где `<prog>.l` — это имя файла с L-программой, являющейся списком L-подпрограмм. (Рекомендуется файлу с L-программой давать имя с расширением l, но это не обязательно.) Первая в списке L-подпрограмма является головной. ОС Linux передаёт ей управление после загрузки файла в оперативную память компьютера. Порядок следования других L-подпрограмм в списке несущественный. Файл может содержать не все необходимые L-подпрограммы. В этом случае компилятор находит недостающие в файле `lib10.l`, являющемся библиотекой пользователя. В ней рекомендуется хранить наиболе часто используемые L-подпрограммы.

Результатом работы компилятора является загрузочный модуль, который хранится под именем `<prog>` (без расширения) в той же папке, где находится и L-программа. Запуск скомпилированной программы на выполнение осуществляется по команде `>$ ./<prog>`.

Компилятор написан на языке C++ с использованием библиотеки регулярных выражений, делающим его максимально простым и прозрачным.

3.2. Структура загрузочного модуля

Загрузочный модуль состоит из двух сегментов — сегмента программы (`.text`) и сегмента данных (`.data`). В свою очередь, первый сегмент состоит из подпрограмм, генерируемых компилятором для L-подпрограмм, вызываемых в процессе исполнения L-программы. Сегмент данных содержит: текущий адрес в памяти для размещения новых комплексов и границу памяти, отводимой ОС под комплексы; текущее состояние PRNG; единичные константы; веса всех булевых векторов в $\{0,1\}^8$ (нужны для реализации операции взвешивания %); все символьные константы, встречающиеся в L-программе.

3.3. Организация памяти

Для каждой L-подпрограммы все её локальные переменные, начала, мощности и ёмкости её локальных комплексов размещаются в стеке, образующем фрейм в 1420 байтов. Доступ к локальным данным осуществляется по фиксированным смещениям от начала фрейма (регистр `ebp`). Значение регистра `ebp` фрейма родительской подпрограммы также сохраняется во фрейме, образуя список фреймов вызванных L-подпрограмм.

Локальные комплексы L-подпрограммы размещаются в «куче». Адрес свободного участка кучи на момент вызова подпрограммы также сохраняется во фрейме.

Создание локального комплекса сопровождается проверкой свободного места путём сравнения величин ёмкости создаваемого комплекса, адреса свободного участка и границы памяти, отводимой под комплексы. Если места достаточно, то адрес начала комплекса получает значение адреса свободного участка, а адрес свободного участка

увеличивается на значение ёмкости комплекса. Если места недостаточно, то выполняется обращение к ОС для увеличения границы доступной памяти.

Такая организация памяти защищена от атак переполнения стека и «кучи», поскольку, во-первых, буферы (комплексы) убраны из стека вместе с возможностью переписать адрес возврата и, во-вторых, нет операций для освобожения «кучи» посредством ОС.

4. Процессор ЛЯПАСа-Т

4.1. Параметры

В ЛЯПАСе-Т, реализованном аппаратно, длина операнда, наибольший номер комплекса и наибольшее количество подпрограмм в иерархической структуре программы обозначаются натуральными n, m и k соответственно. Следовательно, количества различных комплексов (логических и символьных) и переменных в программе на ЛЯПАСе-Т, исполняемой процессором, равны соответственно mk и $27k$. В настоящее время максимальные значения $m = 64$, $k = 128$ вполне достаточны для большинства практических алгоритмов.

4.2. Исполняемый код

Для исполнения процессором программа на ЛЯПАСе-Т должна быть предварительно представлена последовательностью инструкций в исполняемом коде (называемом LE-код) для процессора. Каждая инструкция в нём содержит в себе поля под код операции, тип операнда (константа или нет, тип комплекса — логический, символьный, статический или динамический, если операнд — комплекс или его элемент) и под адреса комплекса и переменной в памяти данных. Поле адреса комплекса используется, если операция имеет дело с данными вида $\varkappa\upsilon$ или $\mathcal{L}$. В первом случае адрес статического комплекса $\varkappa$, являющийся, по определению, адресом его первого элемента, записывается в этом поле явно, и адрес переменной υ записывается в поле адреса переменной. Во втором случае поле адреса комплекса содержит адрес комплекса $\mathcal{L}$ и поле адреса переменной остаётся пустым (равно 0). Во всех других случаях поле адреса комплекса в инструкции пустое. То же самое справедливо и в отношении динамического комплекса с одним исключением: в поле адреса динамического комплекса указывается не сам адрес комплекса, а адрес, по которому он хранится.

4.3. Архитектура

Архитектура процессора, реализующего ЛЯПАС-Т аппаратно, содержит блоки: Память, Арифметико-логическое устройство (АЛУ), Устройство управления (CD — Control Device), Счётчик инструкций (IC — Instruction Counter), Регистр инструкций (IR — Instruction Register) и два Дешифратора — адреса (ADec) и операции (ODec).

Память

Память процессора подразделяется на два сегмента — IM (Instruction Memory) и DM (Data Memory), используемых для запоминания соответственно последовательности инструкций в LE-коде, представляющей некоторую программу P на ЛЯПАСе-Т, и данных для неё — единичных констант Ii, переменных и комплексов для каждой подпрограммы в иерархической структуре программы P, а также параметров (мощностей и ёмкостей) и адресов этих комплексов в DM. Соответственно, для P с k подпрограммами, DM подразделяется условно на четыре секции: секция I — для хранения векторов I$i, i = 0, 1, \ldots, n-1$; секции C и G, разбитые на k подсекций C_j и G_j — для хранения соответственно статических и динамических комплексов в j-й подпрограмме; и секция W, также разбитая на k подсекций W_j — для хранения принадлежащих

j-й подпрограмме локальных переменных a, b, ..., z, параметров и адресов всех комплексов j-й подпрограммы.

Секции I, C и W образуют так называемую статическую память, а секция G — динамическую память процессора. Размещение данных в первой выполняет препроцессор (до исполнения программы P), во второй — сам процессор (во время исполнения P).

Инструкции в IM и данные в DM располагаются плотно, без пропусков элементов памяти, в порядке следования секций I, C, W и G. Количество занятых элементов подсекции G_j в DM отображается значением одного из элементов в W_j. Адрес этого элемента обозначается далее a_j. Его значение есть наименьший из адресов свободных элементов в G_j. В частности, до запуска процессора все a_j совпадают с числом элементов в статической памяти DM.

Статические комплексы, создаваемые в j-й подпрограмме, размещаются в C_j препроцессором путём указания их параметров и адресов в W_j. Ёмкость и адрес в W_j динамического комплекса препроцессор оставляет неопределёнными. Динамический комплекс, создаваемый в j-й подпрограмме с ёмкостью, задаваемой значением некоторой переменной ξ, располагается в G_j в массиве требуемого размера с начальным элементом по адресу, записанному по адресу a_j. Это делается аналогичным образом процессором в момент исполнения им данной операции: параметры и адрес комплекса запоминаются в W_j, значение по адресу a_j увеличивается на ξ.

АЛУ

АЛУ состоит из трёх регистров τ, Z и O максимальной возможной длины n, называемых регистрами общего назначения (CURs – Common Use Registers) и предназначенных для представления соответственно переменных τ, Z и операнда, считываемого из DM, а также операционных устройств (OD — Operational Devices) и схемы CEA (Complex Element Address) определения адреса элемента комплекса в DM.

Операционные устройства используются для выполнения арифметических и логических операций в ЛЯПАСе-Т с участием операндов и результатов операций, представленных в регистрах τ, Z и O.

Для элемента комплекса $\varkappa\upsilon$ его адрес в DM вычисляется схемой CEA как $\theta = \varphi + 4\upsilon$, если $\varkappa = \mathcal{L}$, или как $\theta = \varphi + \upsilon$, если $\varkappa = \mathcal{F}$, где φ есть адрес в DM первого элемента в $\varkappa$ (называемый также адресом самого комплекса $\varkappa$).

Устройство управления

Основные функции блока CD следующие: получать и анализировать информационные сигналы от других блоков, выбирать следующую инструкцию из IM, идентифицировать код операции в ней, определять адрес операнда в DM и адрес следующей инструкции в IM, формировать и посылать управляющие сигналы к другим исполнительным блокам.

4.4. Алгоритм функционирования

1. Устройство управления CD выбирает из IM инструкцию по адресу, указанному в IC, и записывает её в IR.

2. ODec дешифрует содержимое поля кода операции, а ADec — содержимое полей типа и адреса операнда (комплекса и/или переменной) из инструкции в IR.

3. CD по информации от ODec и ADec генерирует сигналы либо для выбора из DM (возможно, посредством схемы CEA) операнда (константы, переменной, комплекса или элемента комплекса) по соответствующему адресу и для его записи в один из CURs, либо, если операнд является константой, заданной в инструкции явно, для записи его в регистр O или в IC.

4. Если операция в инструкции относится к функциональному типу, т. е. является арифметической или логической, то CD генерирует сигнал, инициализирующий соответствующее OD.

5. Это OD выполняет данную операцию, и результат записывается в CURs в соответствии с кодом операции.

6. Если код операции указывает на переход, то содержимое поля адреса переменной в инструкции в IR записывается в IC; в противном случае содержимое в IC увеличивается для выбора следующей инструкции из IM.

7. Если инструкция в IR указывает на создание некоторого динамического комплекса с переменной ёмкостью ξ в j-й подпрограмме, то мощность 0 и ёмкость ξ этого комплекса записываются по адресам его параметров в W_j, а значение по адресу a_j запоминается в W_j как адрес этого комплекса в G_j и затем увеличивается на величину ξ.

В этом алгоритме функционирования процессора поведение устройства управления CD описывается формально на языке конечных автоматов.

4.5. L - п р е п р о ц е с с о р

Для трансляции программы на ЛЯПАСе-Т в LE-код используется специальный компилятор, называемый L-препроцессор. Он может быть написан на любом языке программирования (на ЛЯПАСе-Т, например) для исполнения на любом компьютере, снабжённом соответствующим компилятором. L-препроцессор должен выполнять следующую последовательность действий над заданной L-программой P:

1) каждой подпрограмме в иерархической структуре программы P назначить уникальный номер из ряда 1, 2, ..., k;

2) для каждого $j = 1, 2, \ldots, k$ каждой локальной переменной и каждому статическому локальному комплексу j-й подпрограммы в структуре программы P поставить в соответствие уникальный адрес в W_j;

3) для каждого вызова j-й подпрограммы из i-й подпрограммы в P, $i, j \in \{1, 2, \ldots, k\}$, подставить реальные параметры, указанные в вызове, вместо соответствующих абстрактных (внешних) параметров в теле j-й подпрограммы и вместо самого вызова — инструкции a_i, $\Rightarrow a_j$ и текст j-й подпрограммы, в котором вместо имени всякого динамического комплекса указан адрес в W_j, где лежит адрес этого комплекса в G_j;

4) всякую другую операцию в программе заменить эквивалентной ей цепочкой инструкций в LE-коде — непосредственно либо с промежуточной заменой на цепочку унарных операций, т. е. таких, которые используют самое большее один операнд, отличный от τ.

4.6. А л ь т е р н а т и в н ы й в а р и а н т

В альтернативном варианте процессора в сегменте IM хранятся исполняемые коды как головной программы, так и всех подпрограмм в её иерархической структуре. В этом случае L-препроцессор вместо вызова подпрограммы подставляет в содержащую его (вызов) программу инструкцию A с кодом операции перехода по адресу исполняемого кода данной подпрограммы в IM и в конец последнего записывает инструкцию с кодом операции возврата, т. е. перехода по адресу инструкции, следующей в IM за A. Кроме того, в процессе исполнения программы процессор перед исполнением инструкции A пересылает значения входных операндов подпрограммы, указанных в её вызове, по соответствующим адресам DM в её исполняемом коде, а перед возвратом к инструкции, следующей за A, пересылает результаты исполнения кода подпрограммы по адресам соответствующих её выходных операндов, указанных в её вызове.

Пересылка значений операндов подпрограмм существенно снижает скорость работы процессора по сравнению с его базовым вариантом, но значительно сокращает объём требуемой памяти сегмента IM. Вместе с тем отсутствие пересылок между операндами в базовом варианте не гарантирует сохранения значений входных операндов подпрограммы, если это не предусмотрено программистом.

4.7. Применения

Есть, по меньшей мере, два возможных применения процессора ЛЯПАСа-Т: в качестве криптопроцессора и как управляющего процессора. В первом случае сегмент памяти IM заполняется LE-кодом некоторой программы на ЛЯПАСе-Т, реализующей некоторый криптографический алгоритм, а данные для него (открытый или шифрованный тексты, ключи и др.) записываются в сегмент DM. Во втором случае IM и DM используются для запоминания соответственно LE-кода ЛЯПАС-Т-программы, реализующей некоторый алгоритм, предназначенный для безопасного управления критически важным объектом (космическим, энергетическим, транспортным, технологическим и т. п.), и данных для этой программы.

5. Процессор для подмножества vЛЯПАСа

Пусть L_1 является подмножеством vЛЯПАСа, которое включает все операции первого уровня vЛЯПАСа и не содержит (вызовов) подпрограмм и операций над комплексами. Архитектура процессора для L_1 (называемого также L_1-процессором) была впервые разработана и описана на VHDL в 2012 г. С. Е. Солдатовым, студентом кафедры защиты информации и криптографии Томского государственного университета. С целью предварительной верификации все отдельные блоки в L_1-процессоре и его архитектура в целом промоделированы с помощью программного продукта ModelSim PE Student Edition 10.1d. Кроме того, с использованием автоматизированной системы проектирования ISE WebPACK 9.2i фирмы «Xilinx» синтезирована программируемая логическая интегральная схема L_1-процессора. Максимальная рабочая частота схемы равна 50 МГц, что эквивалентно схемной задержке в 20 нс. Схема занимает 1/3 часть отладочной платы Nexys2 FPGA, Digilent Inc.

Этот результат показывает, что аппаратная реализация процессора для ЛЯПАСа-Т является совершенно реальным проектом, обещающим создание доверенных средств для эффективного исполнения криптографических и безопасных управляющих алгоритмов.

Результаты работы доложены на 12 Всероссийской конференции «Сибирская научная школа-семинар с международным участием „Компьютерная безопасность и криптография“ — SIBECRYPT'13» [8, 9]. Там же был продемонстрирован алгоритм шифрования AES, представленный на языке vЛЯПАС [10].

ЛИТЕРАТУРА

1. LYaPAS, a Programming Language for Logic and Coding Algorithms / eds. M. A. Gavrilov and A. D. Zakrevskii. New York, London: Academic Press, 1969. 475 p.
2. *Торопов Н. Р.* Язык программирования ЛЯПАС // Прикладная дискретная математика. 2009. № 2(4). С. 9–25.
3. *Nadler N.* User group for Russian programming language // IEEE, Newsletter for Computer-Aided Design. 1971. Iss. 3.
4. *Charles J. and Albright Jr.* An Interpreter for the Language LYaPAS. University of North Carolina at Chapel Hill: Department of Computer Science. 1974. 125 p.

5. *Агибалов Г. П.* К возрождению русского языка программирования // Прикладная дискретная математика. 2012. № 3(17). С. 77–84.
6. *Закревский А. Д., Торопов Н. Р.* Система программирования ЛЯПАС-М. Минск: Наука и техника, 1978. 240 с.
7. *Торопов Н. Р.* Диалоговая система программирования ЛЕС. Минск: Наука и техника, 1985. 263 с.
8. *Agibalov G. P., Lipsky V. B., and Pankratova I. A.* Cryptographic extension of Russian programming language // Прикладная дискретная математика. Приложение. 2013. № 6. С. 93–98.
9. *Agibalov G. P., Lipsky V. B., and Pankratova I. A.* Project of hardware implementation of Russian programming language // Прикладная дискретная математика. Приложение. 2013. № 6. P. 98–102.
10. *Broslavskiy O. V.* AES in LYaPAS // Прикладная дискретная математика. Приложение. 2013. № 6. С. 102–104.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.6

О ЗАДАЧЕ ОПРЕДЕЛЕНИЯ ЛИНЕЙНОЙ И АФФИННОЙ ЭКВИВАЛЕНТНОСТИ ПОДСТАНОВОК

И. В. Панкратов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: ivan.pankratov2010@yandex.ru

Представлены результаты экспериментов над программной реализацией алгоритмов определения линейной и аффинной эквивалентности подстановок. Подробно описаны алгоритмы определения линейной эквивалентности между двумя блоками замены, построения класса линейной эквивалентности и определения аффинной эквивалентности. Произведено сравнение оценки трудоёмкости алгоритмов с реальным временем работы.

Ключевые слова: *линейная эквивалентность, аффинная эквивалентность, блок замены, подстановка.*

В работе рассматривается задача определения линейной и аффинной эквивалентности подстановок, описанная в [1]. Подробно описаны и программно реализованы алгоритм LE определения линейной эквивалентности между двумя блоками замены, алгоритм LEC построения класса линейной эквивалентности и алгоритм AE определения аффинной эквивалентности.

1. Постановка задачи

Введём некоторые обозначения:

- $V_n = \{0,1\}^n$ — пространство булевых векторов размерности n;
- $A \cdot B(x) = A(B(x))$ — композиция отображений A и B;
- пару векторов $x, y \in V_n$, такую, что $A(x) = y$, назовём *точкой преобразования* A и обозначим $(x \to y)$.

Отображение $L : V_n \to V_n$ назовём *линейным преобразованием пространства* V_n, если оно удовлетворяет свойству

$$\forall x, y \in V_n \; L(x+y) = L(x) + L(y).$$

Такие преобразования удобно задавать матрицами размера $n \times n$. Квадратная матрица L размера $n \times n$ задаёт линейное преобразование вектор-строк $L(x) = x \cdot L$. Далее будем отождествлять матрицу и задаваемое ею отображение.

Преобразование называется *аффинным*, если его можно представить в виде $A(x) = L(x) + c$, где $L(x)$ — линейное преобразование с матрицей L, а c — константа из V_n. Матрицу L будем называть матрицей аффинного преобразования. Известно, что аффинное преобразование обратимо, если его матрица невырождена.

Две обратимые подстановки S_1 и S_2 называются *линейно* или *аффинно эквивалентными*, если существуют такие линейные или аффинные соответственно отображения A и B^{-1}, что

$$B^{-1} \cdot S_1 \cdot A = S_2.$$

Очевидно, что отображения A и B^{-1} обратимы, то есть их матрицы невырождены. Можно записать эквивалентное условие: существуют такие обратимые линейные или аффинные соответственно отображения A и B, что

$$S_1 \cdot A = B \cdot S_2. \tag{1}$$

Необходимо для пары заданных подстановок S_1, S_2 найти отображения A и B, удовлетворяющие (1), либо убедиться, что таких отображений нет.

2. Алгоритм определения линейной эквивалентности

Рассмотрим задачу определения линейной эквивалентности подстановок S_1 и S_2 на множестве V_n. Для решения этой задачи разработан алгоритм LE [1]. Данный алгоритм пытается построить невырожденные матрицы A и B, удовлетворяющие условию (1).

В алгоритме используются две основные идеи:

1) «эффект иголки» (needlework effect), суть которого в том, что подбор части точек преобразования A позволяет найти точки преобразования B. В свою очередь, новые точки B позволяют получить новую информацию про точки A;

2) «экспоненциальный рост количества догадок» (exponential amplification of guesses), происходящий благодаря линейности преобразований. Этот эффект состоит в том, что, зная k линейно-независимых точек A, можно легко получить 2^k линейных комбинаций этих точек.

Будем использовать следующие множества:

— C_A и C_B — множества точек, для которых соответствующее преобразование (A или B) известно. По построению, эти множества линейно замкнуты, то есть содержат все линейные комбинации известных точек;
— множества N_A и N_B содержат вновь полученные точки, для которых стали известны преобразования (A и B соответственно), но которые не являются линейными комбинациями точек из C_A и C_B;
— U_A и U_B — множества неизвестных точек.

Множества прообразов точек в C_A, N_A и U_A попарно не пересекаются и их объединение есть все векторное пространство V_n, то есть они представляют разбиение векторного пространства. То же верно и для множеств C_B, N_B и U_B. С учётом этого свойства в программной реализации вместо множеств U_A и U_B использованы объединения $C_A \cup N_A$ и $C_B \cup N_B$ соответственно.

Суть алгоритма такова: пока имеются точки в N_A, можно построить соответствующие им точки преобразования B. Если $(x \to y)$ — точка преобразования A, то $(S_2(x) \to S_1(y))$ должна быть точкой преобразования B, поскольку предполагается, что $S_1 \cdot A = B \cdot S_2$. Помимо этого, поскольку преобразования A и B линейны, можно получить точки преобразования B для всех линейных комбинаций известных точек преобразования A с участием точки из N_A. Часть вновь полученных точек преобразования B может оказаться линейно независимой от его известных точек и попасть в множество N_B. Это происходит благодаря нелинейности преобразований S_1 и S_2. Аналогично по точкам из N_B можно получать точки преобразования A.

Когда множества N_A и N_B пусты, необходимо заниматься подбором, то есть строить предположение о точке преобразования A и добавлять её в N_A. Затем можно пытаться достроить преобразования A и B с помощью описанной выше процедуры. Когда найдено достаточное количество точек, строятся матрицы A и B и проверяется, удовлетворяют ли они свойству (1). Если да, то подстановки линейно эквивалентны и найдены соответствующие матрицы, в противном случае последнее предположение отвергается и строится следующее. Если все предположения исчерпаны, делается вывод, что подстановки не линейно эквивалентны.

Для описания алгоритма введём обозначения: $F(W) = \{F(x) : x \in W\}$, $W + c = \{x + c : x \in W\}$, где $W \subseteq V_n$; $c \in V_n$; $F : V_n \to V_n$. Для простоты изложения операции над точками будем обозначать как операции над прообразами, предполагая соответствующие преобразования над образами. Например, в шаге 3.3.2 под $S_2(x+C_A)$ предполагается $\{(S_2(x + c) \to S_1(y + d)) : (c \to d) \in C_A\}$, а в шаге 3.3.3 под $x + C_A$ понимается $\{((x + c) \to (A(x) + A(c))) : (c \to A(c)) \in C_A\}$.

Алгоритм LE

1. $N_A := N_B := \varnothing$; $C_A := C_B := \{(0 \to 0)\}$.
2. **Если** $S_1(0) \neq 0$ и $S_2(0) \neq 0$, добавляем $(S_2^{-1}(0) \to S_1^{-1}(0))$ в C_A и $(S_2(0) \to S_1(0))$ в C_B;
   иначе проверяем, что $S_1(0) = S_2(0) = 0$, в противном случае подстановки не эквивалентны.
3. **Повторяем** в безусловном цикле:
   3.1. **Если** последнее предположение отвергнуто, восстанавливаем состояния C_A и C_B, какие были до предположения; полагаем $N_A := N_B := \varnothing$.
   3.2. **Если** $N_A = N_B = \varnothing$:
      3.2.1. Делаем предположение о значении $A(x)$ для некоторого $x \in U_A$.
         Если все возможные предположения были отвергнуты, то подстановки не эквивалентны.
      3.2.2. Добавляем в N_A точку $(x \to A(x))$.
   3.3. **Пока** $N_A \neq \varnothing$:
      3.3.1. Выбираем некоторую точку $(x \to y) \in N_A$; полагаем $N_A := N_A \setminus \{(x \to y)\}$.
      3.3.2. Полагаем $N_B := S_2(x + C_A) \setminus C_B$.
      3.3.3. Полагаем $C_A := C_A \cup (x + C_A)$.
      3.3.4. **Если** $|N_B| + \log|C_B| > \text{const} \cdot n$:
         Если B — линейное обратимое отображение, порождаем A и проверяем условие (1) на всех точках, оставшихся в U_A и U_B;
         иначе отвергаем последнее предположение.
   3.4. **Пока** $N_B \neq \varnothing$:
      3.4.1. Выбираем некоторую точку $(y \to z) \in N_B$; полагаем $N_B := N_B \setminus \{(y \to z)\}$.
      3.4.2. $N_A := S_2^{-1}(y + C_B) \setminus C_A$.
      3.4.3. $C_B := C_B \cup (y + C_B)$.
      3.4.4. **Если** $|N_A| + \log|C_A| > \text{const} \cdot n$:
         Если A — линейное обратимое отображение, порождаем B и проверяем условие (1) на всех точках, оставшихся в U_A и U_B;
         иначе отвергаем последнее предположение.

Константа const влияет на быстродействие алгоритма. Очевидно, её значение должно быть меньше единицы, иначе условие может стать невыполнимым.

В случае, когда $S_1(0) \neq 0$ и $S_2(0) \neq 0$, как правило, достаточно подобрать одну точку преобразования A, после чего остальные точки находятся с помощью описанной процедуры. Если же $S_1(0) = S_2(0) = 0$, то необходимо перебирать, как минимум, две точки. Объём перебора составляет соответственно 2^n и 2^{2n}. Самый трудоёмкий шаг проверки подобранных точек — построение матриц — имеет трудоёмкость порядка n^3. Итоговая трудоёмкость получается соответственно $\mathrm{O}(n^3 2^n)$ и $\mathrm{O}(n^3 2^{2n})$.

3. Построение класса линейной эквивалентности

Класс линейной эквивалентности определяется каноническим представителем. От выбора канонического представителя класса зависит трудоёмкость его нахождения. В [1] в качестве представителя класса выбрана минимальная подстановка в смысле лексикографического упорядочивания таблицы значений. Там же даётся алгоритм нахождения этого представителя LEC.

Алгоритм LEC последовательно строит минимальную подстановку, эквивалентную заданной, то есть для подстановки S строится минимальная подстановка $R_S = B^{-1} \cdot S \cdot A$. Схема алгоритма аналогична алгоритму LE, используются похожие множества точек:

- C_A и C_B — множества точек одного преобразования, для которых известны соответствующие им точки другого преобразования. Например, если $(x \to y)$ — точка из C_A, то в C_B должна быть точка $(x' \to S(y))$ для некоторого x', и наоборот. Таким образом, для точек из этих множеств известны точки самого отображения R_S;
- множества N_A и N_B содержат точки, для которых ещё не известны соответствующие точки другого преобразования;
- U_A и U_B — множества неизвестных прообразов.

Основной шаг алгоритма — достраивание преобразований A и B оптимальным образом с точки зрения минимизации итогового отображения R_S. Для этого, пока есть точки в N_A, то есть прообразы R_S, для которых ещё не определены образы, выбирается наименьший прообраз $x \in N_A$. Затем выбирается наименьший прообраз $y \in U_B$ и определяются точки A и B так, чтобы добиться выполнения условия $R_S(x) = y$, после чего все множества обновляются с использованием свойства линейности отображений A и B. Если же множество N_A пусто, поступаем наоборот, выбирая наименьший образ $y \in N_B$ и наименьший прообраз $x \in U_A$ и вновь определяя точки A и B, добиваясь $R_S(x) = y$. Это продолжается до тех пор, пока не окажутся пустыми оба множества N_A и N_B. При этом либо станут полностью известны преобразования A и B, либо производится подбор какой-либо точки преобразования A. При подборе из всех полученных вариантов выбирается тот, что доставляет минимум отображению R_S.

Как и в случае алгоритма LE, теоретическая трудоёмкость алгоритма LEC составляет $\mathrm{O}(n^3 2^n)$ и $\mathrm{O}(n^3 2^{2n})$ в зависимости от того, отображает ли подстановка S ноль на ноль.

4. Определение аффинной эквивалентности

Простой алгоритм определения аффинной эквивалентности, основанный на алгоритме LE, подразумевает подбор констант аффинных преобразований A и B из условия (1). Перебор двух констант приводит к необходимости 2^{2n} раз выполнить процедуру LE, что даёт итоговую трудоёмкость $\mathrm{O}(n^3 2^{3n})$.

Другой подход к этой задаче основан на алгоритме LEC. Он позволяет решить задачу, выполнив алгоритм LEC $2 \cdot 2^n$ раз: идея состоит в построении классов эквивалентности для подстановки $S_1(x + a)$ для всех $a \in V_n$ и для подстановки $S_2(x + b)$ для

всех $b \in V_n$. После этого достаточно проверить наличие пересечения между классами первой и второй подстановок.

Алгоритм AE

1. Для всех $a \in V_n$ находим $\mathrm{LEC}(S_1(x+a))$ и помещаем в таблицу T_1.
2. Для всех $b \in V_n$ находим $\mathrm{LEC}(S_2(x)+b)$ и помещаем в таблицу T_2.
3. Если $T_1 \cap T_2 \neq \varnothing$, то подстановки аффинно эквивалентны, иначе нет.

Сложность данного алгоритма равна $\mathrm{O}(n^3 2^{2n})$, поскольку среди подстановок с константами найдётся ровно одна такая, которая отображает ноль на ноль.

5. Экспериментальные данные

Для получения статистических данных для каждого n от двух до десяти построено по 10000 подстановок на векторах размерности n. Для каждой размерности проведено по 1000 экспериментов каждого типа.

Для измерения времени работы алгоритмов использовалась функция Windows API `GetThreadTimes`, которая считает время работы только одного потока, для большей объективности измерений и независимости от загруженности компьютера в целом.

В столбцах табл. 1, 2, 4 перечислены размерность векторного пространства V_n, над которым строились подстановки, среднее $T_{\mathrm{ср}}$ и максимальное $T_{\max}$ время работы алгоритма (в секундах), дисперсия времени работы и разброс, то есть отношение максимального времени к среднему. В табл. 1 приведены результаты измерения времени работы алгоритма LE над двумя случайно выбранными подстановками из общего множества. В табл. 2 приведены аналогичные результаты для алгоритма LEC. Полужирным шрифтом выделены ячейки с неожиданно большими значениями.

Таблица 1

Время работы алгоритма LE

n	$T_{\mathrm{ср}}$	$T_{\max}$	Дисперсия	Разброс
2	0,000234	0,0156	3,6E−06	66,66667
3	0,001841	0,0624	2,88E−05	33,89831
4	0,005242	0,093601	6,99E−05	17,85714
5	0,011513	0,343202	0,000171	29,8103
6	0,023915	0,0624	8,35E−05	2,609263
7	0,094552	0,280802	0,000633	2,969807
8	0,198948	0,561604	0,001218	2,822865
9	0,411624	0,483603	0,002031	1,174865
10	0,840861	0,967206	0,004319	1,150257

По табл. 1 и 2 можно заметить, что среднее время работы алгоритма LEC меньше такового для алгоритма LE для всех размерностей, за исключением размерности 10, где среднее время работы алгоритма LEC превышает таковое для алгоритма LE более чем вдвое. На векторах этой размерности самый долгий эксперимент длился 759,4753 с, хотя самый долгий из остальных — только 3,775224 с. Более детальный анализ показал, что в этом случае алгоритму понадобилось находить подбором три точки преобразования A. Дополнительные опыты показали, что при поиске канонического представителя класса эквивалентности подобное случается достаточно редко. В табл. 3 представлены результаты исследования длины подбора для задач различных размерностей; для каждого k от 1 до 4 указано количество экспериментов, в которых пришлось находить подбором k точек. Видно, что для задач размерности 10 бит подбор трёх точек

Таблица 2

Время работы алгоритма LEC

n	$T_{\text{ср}}$	$T_{\max}$	Дисперсия	Разброс
2	0,000156	0,0156	2,41E−06	100
3	0,001123	0,0156	1,63E−05	13,88889
4	0,00404	0,156001	9,06E−05	38,61004
5	0,009641	0,234002	0,000507	24,27184
6	0,022995	0,951606	0,004902	41,38399
7	0,059046	3,650423	0,067703	61,82299
8	0,177467	**19,39092**	1,276042	**109,2651**
9	0,317727	1,107607	0,032235	3,486031
10	1,769223	**759,4753**	**575,5924**	**429,2705**

случается приблизительно в 1,2 % случаев. Правый столбец таблицы содержит математическое ожидание M количества подстановок, сохраняющих ноль, среди десяти тысяч случайных подстановок указанной размерности. Это значение достаточно близко к количеству подстановок, вызвавших максимально длинный перебор.

Таблица 3

Количество точек, получаемых подбором

	k				
n	1	2	3	4	M
2	7481	2519	0	0	2500
3	4753	4287	960	0	1250
4	5380	3953	650	17	625
5	5721	3942	337	0	312,5
6	5968	3905	127	0	156,25
7	5990	3923	87	0	78,125
8	5956	4006	38	0	39,0625
9	6041	3941	18	0	19,53125
10	6020	3968	12	0	9,765625
11	6117	3876	7	0	4,882813
12	6060	3939	1	0	2,441406

Собрана статистика по алгоритму LEC без учёта одного — самого длительного — эксперимента для каждой размерности. Результаты можно видеть в табл. 4. Далее такую статистику будем обозначать LEC*. Можно заметить, что величина разброса в ней значительно меньше.

Сумма величин среднего времени работы алгоритма LE составила 1,58873 с, LEC — 2,36141 с, а LEC* — 1,57799 с. Можно видеть, что значения для LE и LEC* близки. Они были использованы для нормирования значений времени с тем, чтобы проверить соответствие значений реальных измерений времени работы алгоритма их оценке $\mathrm{O}(n^3 2^n)$. В табл. 5 приведены отношения среднего времени работы алгоритма к теоретической оценке, нормированной по сумме значений.

Таблица 4

Время работы алгоритма LEC без учёта самых длительных экспериментов

n	$T_{\text{ср}}$	$T_{\max}$	Дисперсия	Разброс
2	0,000141	0,0156001	2,17E−06	111
3	0,001109	0,0156001	1,61E−05	14,07042254
4	0,003888	0,0468003	6,75E−05	12,03614458
5	0,009416	0,1716011	0,000457	18,2238806
6	0,022065	0,7020045	0,004042	31,81528662
7	0,055451	3,6192232	0,054834	65,26837511
8	0,158234	18,798121	0,907054	118,7994698
9	0,316937	0,936006	0,031641	2,953291289
10	1,010758	3,7752242	0,324316	3,735041018

Таблица 5

Сравнение экспериментальных значений времени работы с теоретической оценкой

n	LE	LEC
2	7,324469	3,285201
3	8,536171	3,504215
4	5,127128	2,65896
5	2,882911	1,624204
6	1,732779	1,120923
7	2,157124	0,906304
8	1,520328	0,912413
9	1,104615	0,573643
10	0,822492	1,164306

Заключение

Практические измерения показали, что для алгоритма LE разброс значений времени работы, то есть максимальное превышение среднего времени работы для одиночного опыта, очень невелик — около 15–17 % для подстановок размерности 9 и 10 битов. Для алгоритма LEC разброс может превышать 40000 %.

ЛИТЕРАТУРА

1. *Biryukov A., De Canniere C., Braeken A., and Preneel B.* A toolbox for cryptanalysis: linear and affine equivalence algorithms // EUROCRYPT 2003. LNCS. 2003. V. 2656. P. 33–50.

УДК 519.7

О РЕАЛИЗАЦИИ АЛГОРИТМА КОППЕРСМИТА ДЛЯ ДВОИЧНЫХ МАТРИЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА ВЫЧИСЛИТЕЛЯХ КЛАСТЕРНОГО ТИПА

А. С. Рыжов

Лаборатория ТВП, г. Москва, Россия

E-mail: ryzhovalexander@mail.ru

Рассматривается задача реализации алгоритма Копперсмита, вычисляющего векторные аннулирующие многочлены для матричных последовательностей, на современных 64-разрядных ЭВМ. Рассмотрены вопросы представления данных для случая последовательностей бинарных матриц с точки зрения снижения трудоёмкости алгоритма. Предложены способы эффективного распараллеливания алгоритма для реализации на ЭВМ с многоядерными процессорами, а также для выполнения алгоритма на вычислителях кластерного типа.

Ключевые слова: *матричные последовательности, алгоритм Копперсмита.*

Введение

Во многих алгоритмах вычислительной алгебры возникает задача нахождения решений сильно разреженной системы однородных линейных уравнений (СОЛУ). Например, при нахождении дискретного логарифма в поле $\mathrm{GF}(2^n)$ (при $p = 2^n - 1$ — простом) методом Копперсмита [1] необходимо найти решения такой системы над простым полем $\mathbb{Z}_p$.

В настоящий момент для решения данной задачи обычно используется один из двух алгоритмов: алгоритм Ланцоша — Монтгомери [2] или алгоритм Видемана — Копперсмита [3]. Оба алгоритма имеют свои преимущества и недостатки, а также ограничения на используемые вычислители. Алгоритм Ланцоша — Монтгомери реализован, например, в пакете GGNFS. Он ориентирован на выполнение на одном вычислителе кластерного типа с высокоскоростным внутренним соединением узлов. Алгоритм Видемана — Копперсмита, напротив, позволяет наиболее трудоёмкие этапы выполнять на нескольких не связанных друг с другом кластерах.

Блочный алгоритм Видемана — Копперсмита является обобщением алгоритма Видемана поиска решения разреженной СОЛУ [4], ориентированным на вычислительную технику, работающую со словами в несколько байтов. Один из этапов алгоритма Видемана — нахождение аннулирующего многочлена двоичной последовательности с помощью алгоритма Берлекэмпа — Мэсси, и его непосредственное обобщение на многомерный случай является, по-видимому, трудной задачей. Копперсмит решил эту проблему, сведя задачу к поиску векторных приближений Паде и разработав новый алгоритм для построения таких приближений.

В данной работе рассматриваются вопросы реализации алгоритма Копперсмита построения векторных приближений Паде при решении СОЛУ над полем $\mathrm{GF}(2)$. Приводится описание алгоритма Видемана — Копперсмита нахождения решений разреженных СОЛУ, подробное описание метода Копперсмита построения векторных приближений Паде, особенности реализации этого алгоритма на ЭВМ. Отдельно рассмотрены особенности реализации алгоритма Копперсмита на вычислителях кластерного типа.

1. Блочный алгоритм Видемана — Копперсмита решения систем однородных линейных уравнений

Описание алгоритма Видемана — Копперсмита нахождения решений разреженных систем однородных линейных уравнений можно найти, например, в [3]. Приведём основные шаги алгоритма, чтобы показать, где используется алгоритм Копперсмита построения векторных приближений Паде. Пусть $P = \mathrm{GF}(2)$ — поле из двух элементов, A — матрица размеров $M \times N$ над полем P, и пусть необходимо найти нетривиальное (т. е. ненулевое) решение СОЛУ

$$Ax = 0. \tag{1}$$

Заметим, что не ставится задача нахождения всех решений системы (1) или приведения матрицы A к специальному виду. Чтобы гарантировать наличие нетривиальных решений, будем считать, что $M < N$.

Пусть $m, n \in \mathbb{N}$ — параметры алгоритма, $m \geqslant n$. Пусть, далее, $Z \in P_{m,M}$ — случайная матрица размера $m \times M$ над полем P, $B \in P_{M,M}$ — матрица из первых M столбцов матрицы A решаемой СОЛУ (1), $Y \in P_{M,n}$ — матрица, составленная из столбцов матрицы A с номерами $M+1, M+2, \ldots, M+n$ (пусть $M + n \leqslant N$). Последовательно выполняем следующие три этапа:

1) Вычислим первые $L+1$ элементов последовательности матриц $a = a_0 a_1 \ldots$ размера $m \times n$ над полем P, где $a_i = ZB^iY$ (здесь L — параметр метода; см. ниже).
2) Найдём *векторный аннулирующий многочлен* для последовательности a, т. е. такие векторы $f_0, \ldots, f_d \in P^n$, что $\sum\limits_{j=0}^{d} a_{i+j} f_j = 0 \in P^{(m)}$ для всех $i \geqslant 0$.
3) Из последнего равенства получаем, что вектор $\sum\limits_{j=0}^{d} B^j Y f_j$ лежит в ядре любой матрицы ZB^i, $i = 0, 1, \ldots$ Отсюда с большой вероятностью $\sum\limits_{j=0}^{d} B^j Y f_j = 0$, т. е. $B \sum\limits_{j=0}^{d-1} B^j Y f_{j+1} + Y f_0 = 0$. Следовательно, вектор-столбец $w = \begin{pmatrix} \sum\limits_{j=0}^{d-1} B^j Y f_{j+1} \\ f_0 \end{pmatrix}$, дополненный нулями до длины N, является решением системы (1) (напомним, Y состоит из столбцов матрицы A, идущими после подматрицы B). Если вектор w равен нулю, то $\sum\limits_{j=0}^{d-1} B^j Y f_{j+1} = 0$, и выполняются аналогичные рассуждения.

Замечание 1. Необходимо отметить, что для последовательности a указанного вида векторный аннулирующий многочлен, вычисляемый на втором этапе алгоритма, всегда существует. При $m = n = 1$ таким многочленом является любой аннулирующий многочлен матрицы B — например характеристический или минимальный. В общем случае равенство $\sum\limits_{j=0}^{d} B^j Y f_j = 0$ можно рассматривать как систему из M однородных линейных уравнений от $(d+1)n$ неизвестных элементов векторов $\{f_j\}$ над полем P, и при $d \geqslant M/n$ эта система имеет ненулевое решение — искомый векторный аннулирующий многочлен.

Исходный алгоритм Видемана, опубликованный в работе [4], является частным случаем приведённого выше блочного алгоритма Видемана — Копперсмита при $m = n = 1$. В алгоритме Видемана на втором этапе необходимо найти аннулирующий многочлен

последовательности $a = a_0 a_1 \ldots$ элементов поля P, при этом достаточно вычислить первые $2M + 1$ элементов, так как ранг последовательности не превосходит ранг матрицы B. Для поиска аннулирующего многочлена можно использовать, например, алгоритм Берлекэмпа — Мэсси.

Для нахождения векторного аннулирующего многочлена последовательности матриц Копперсмит предложил новый алгоритм, основанный на последовательном вычислении приближений Паде. При этом в результате работы алгоритма может получиться до n различных независимых аннулирующих многочленов, что в результате дает до n линейно независимых решений системы (1).

Для нахождения векторного аннулирующего многочлена достаточно иметь первые $L + 1$ элементов последовательности a, где $L = \lfloor M/m \rfloor + \lfloor M/n \rfloor + \Delta$, $\Delta \approx 100$. Это следует из набора линейных соотношений, а также из теоретико-вероятностных соображений [3].

По сравнению с методом Видемана, в алгоритме Видемана — Копперсмита на первом и третьем этапах требуется выполнить существенно меньшее число итераций. Для удобства реализации на современных ЭВМ числа m и n выбирают, как правило, кратными длине машинного слова: например, $m = 128$, $n = 64$. Расчет трудоёмкости приведён автором метода в [3]. Алгоритм дает выигрыш только в случае сильной разреженности матрицы системы: для первого и третьего этапов трудоёмкость пропорциональна $M^2 w$, где w — средний вес строки матрицы B, т. е. среднее число ненулевых элементов в строке. Трудоёмкость второго этапа при использовании алгоритма Копперсмита равна $\mathrm{O}(M^2 m)$.

Далее приводится подробное изложение алгоритма Копперсмита и рассматриваются вопросы, связанные с его эффективной реализацией на вычислителях кластерного типа.

2. Алгоритм Копперсмита вычисления векторных аннулирующих многочленов

Описание алгоритма Копперсмита вычисления векторных аннулирующих многочленов приведено автором алгоритма в [3]. Алгоритм сформулирован достаточно громоздко в терминах отдельных элементов коэффициентов степенных рядов над кольцом матриц. Здесь приводится описание алгоритма, в основном схожее с описанием в работе [5] и более удобное для дальнейшего изложения.

Итак, для последовательности матриц над конечным полем (не обязательно $\mathrm{GF}(2)$) необходимо найти векторный аннулирующий многочлен. Поскольку алгоритм Копперсмита, как было сказано ранее, выдаёт сразу несколько решений, будем рассматривать задачу нахождения *матричного аннулирующего многочлена*: для последовательности $a = a_0 a_1 \ldots$ матриц размера $m \times n$ над конечным полем P необходимо найти матрицы $f_0, \ldots, f_d \in P_{n,r}$ размера $n \times r$, одновременно не равные нулю, такие, что

$$\forall i \geqslant 0 \ \sum_{j=0}^{d} a_{i+j} f_j = 0_{m \times r}. \tag{2}$$

Будем изначально предполагать существование решения.

Замечание 2. Последовательность $a = a_0 a_1 \ldots$, обладающую свойством (2), в общем случае нельзя называть линейной рекуррентной последовательностью, поскольку при $m \neq n$ множество возможных значений коэффициентов этой последовательности не образует структуру кольца. Нельзя также говорить о многочлене вида $f_0 x^d + f_1 x^{d-1} + \cdots + f_d$. Тем не менее далее соответствующие объекты будем

называть многочленами и степенными рядами для удобства и наглядности изложения. Например, будем говорить «степенной ряд $H(x) = \sum a_i x^i$» или «многочлен $F(x) = f_0 x^d + f_1 x^{d-1} + \cdots + f_d$», а также рассматривать их «произведение», если размеры коэффициентов допускают перемножение. При этом какие-либо алгебраические свойства этих объектов использоваться не будут.

Замечание 3. Как было отмечено выше, при решении исходной задачи — нахождения решений СОЛУ (1) — построенная последовательность $a = (ZB^iY : i = 0, 1, \ldots)$ заведомо обладает свойством (2), поскольку система $\sum\limits_{j=0}^{d} B^jY f_j = 0$, состоящая из rM уравнений от $r(d+1)n$ неизвестных (коэффициентов матриц $f_0, \ldots, f_d$) при $d \geqslant M/n$ имеет ненулевое решение.

Рассмотрим произведение ряда $H(x) = \sum a_i x^i$ и многочлена $F(x) = f_0 x^d + f_1 x^{d-1} + + \cdots + f_d$:

$$H(x)F(x) = \sum_{i=0}^{\infty} \sum_{j=0}^{d} a_i f_j x^{i+d-j} = \sum_{k=0}^{\infty} \left(\sum_{j=0}^{d} a_{k-d+j} f_j \right) x^k$$

(здесь $a_i = 0$ при $i < 0$). Ясно, что если выполняется условие (2), то ряд $H(x)F(x)$ является многочленом степени не выше $d-1$, то есть его коэффициенты при $x^d, x^{d+1}, \ldots$ равны нулю. При этом $F(x)$ называется *правым порождающим многочленом* последовательности a. Обратно, если для некоторых многочленов $F(x) \in P_{n,r}[x], Q(x) \in P_{m,r}[x]$ выполняется $H(x)F(x) = Q(x)$, то коэффициенты многочлена $F(x)$ являются решением поставленной задачи.

Если для некоторых многочленов $F(x)$, $Q(x)$ и рядов $H(x)$, $C(x)$ выполняется равенство

$$H(x)F(x) = Q(x) + x^t C(x), \tag{3}$$

то многочлены F и Q называют *приближениями Паде* порядка t для степенного ряда $H(x)$. Алгоритм Копперсмита итеративно строит приближения Паде порядка $t = t_0, t_0 + 1, \ldots$ Как было оговорено выше, последовательность a обладает правым порождающим многочленом, поэтому при достаточно большом t получим правый порождающий многочлен этой последовательности.

Приведём описание алгоритма Копперсмита. Через E_m будем обозначать единичную матрицу размера $m \times m$. Положим $A(x) = (H(x)| - E_m)_{m\times(m+n)}$, $G_t(x) = \begin{pmatrix} F_t(x) \\ Q_t(x) \end{pmatrix}_{(m+n)\times(m+n)}$. Тогда $A(x)G_t(x) = H(x)F_t(x) - Q_t(x)$ и равенство (3) равносильно равенству $A(x)G_t(x) = x^t C_t(x)$ (здесь $C(x)$ из равенства (3) заменено на $C_t(x)$, чтобы показать, что этот матричный многочлен определяется номером шага алгоритма $t = 0, 1, 2, \ldots$). Для столбцов матрицы $G_t(x) = \big(G_{t,1}(x), \ldots, G_{t,(m+n)}(x)\big)$ введём целочисленные векторы $\delta_t = (\delta_{t,1}, \ldots, \delta_{t,(m+n)})$, элементы которых вычисляются в процессе работы алгоритма и в некотором роде соответствуют степеням этих векторных многочленов. (Под степенью столбца матрицы над кольцом многочленов будем понимать максимум степеней многочленов, являющихся элементами этого столбца: $\deg\,(g_1(x), \ldots, g_k(x))^T = \max\limits_{i\in\{1,\ldots,k\}} \{\deg g_i(x)\}$.) А именно:

— степень столбца $G_{t,j}(x)$ не превосходит соответствующее значение $\delta_{t,j}$;
— при сложении двух столбцов $G_{t,j_1}(x)$ и $G_{t,j_2}(x)$ результату будет соответствовать максимум их значений $\max\{\delta_{t,j_1}, \delta_{t,j_2}\}$;
— при умножении столбца на x соответствующее ему значение увеличивается на 1.

На каждом шаге алгоритма Копперсмита последовательно вычисляются пары $(G_t(x), C_t(x))$, для которых выполняются следующие три условия:

$$A(x)G_t(x) = x^t C_t(x); \tag{4}$$

$$\operatorname{rank} C_t(0) = m; \tag{5}$$

$$\deg G_{t,j}(x) \leqslant \delta_{t,j},\ j = 1, 2, \ldots, m+n,\ \sum_{j=1}^{m+n} \delta_{t,j} = tm. \tag{6}$$

Инициализация значений $(G_0(x), C_0(x))$ выполняется следующим образом: $G_0(x) = E_{(m+n)\times(m+n)}$, $C_0(x) = (H(x)|E_m)$. Значения $\delta_0 = (\delta_{0,1}, \ldots, \delta_{0,m+n})$ задаются равными нулю. Выполнение условий (4) – (6) для $t = 0$ очевидно.

При выполнении очередного шага t необходимо так изменить матрицу $C_t(x)$, чтобы можно было выделить одну степень x. Для этого достаточно n столбцов младшего коэффициента (т. е. $C_t(0)$) сделать нулевыми, а остальные m столбцов домножить на x.

Формально, на шаге t по паре $(G_t(x), C_t(x))$ вычисляется пара $(G_{t+1}(x), C_{t+1}(x))$ следующим образом:

1) Вычислим матрицу перехода τ_t размера $(m+n)\times(m+n)$, которая представляет собой невырожденные линейные преобразования столбцов, приводящие матрицу $C_t(0)$ к ступенчатому виду: $C_t(0)\tau_t = (0|E)$. При этом к столбцу $G_{t,j}(x)$ можно прибавлять только те столбцы $G_{t,j_2}(x)$, соответствующие значения δ_{t,j_2} которых не больше $\delta_{t,j}$ (этого можно добиться за счёт перестановки столбцов; при этом значения $\delta_{t,j}$ также переставляются).
2) Вычислим $G_{t+1}(x) = G_t(x)\tau_t D$, где $D = \operatorname{diag}(1, \ldots, 1, x, \ldots, x)$ (единицы стоят на первых n элементах диагонали).
3) Вычислим $C_{t+1}(x) = C_t(x)\tau_t D/x$.
4) Положим $\delta_{t+1,j} = \begin{cases} \delta_{t,j}, & \text{если } j \in \{1, \ldots, n\} \\ \delta_{t,j} + 1, & \text{если } j \in \{n+1, \ldots, m+n\}. \end{cases}$

Заметим, что вычисление матрицы τ_t возможно в силу выполнения условия (5). Деление на x при вычислении $C_{t+1}(x)$ возможно в силу того, что первые n столбцов матрицы $C_t(0)\tau_t$ равны нулю, т. е. делятся на x, а оставшиеся m столбцов домножаются на x при умножении на матрицу D.

Проверим выполнение условий (4) – (6). Выполнение условия (4) для $t+1$ очевидно (если оно выполняется для t): левая и правая части равенства (4) для t домножены справа на одну и ту же полиномиальную матрицу $\tau_t D$. Условие (5) для $t+1$ следует из того, что последние m столбцов матрицы $C_t(0)\tau_t$ являются единичными. Условие (6) выполняется в силу особенностей построения матрицы перехода τ_t, а также умножения $G_t(x)\tau_t$ на матрицу D, содержащую ровно m элементов x на диагонали.

Признаком останова алгоритма является наличие в матрице $G_t(x)$ достаточного (требуемого) числа столбцов $G_{t,j}(x)$, для которых выполняется неравенство

$$t - \delta_{t,j} > \frac{N}{m} + \Delta_2, \tag{7}$$

где Δ_2 — наперёд заданный постоянный параметр; $\Delta_2 \approx 10 - 100$. Если необходимо найти n решений, то в силу условия (6) останов выполнится по крайней мере через $t \sim N/m + N/n$ шагов.

Столбцы, для которых выполняется условие (7), представляют собой решение задачи. Поскольку нас интересуют только сами правые порождающие многочлены $F_{t,j}(x)$, нет необходимости вычислять на каждом шаге всю матрицу $G_t(x) =$

$= \begin{pmatrix} F_t(x) \\ Q_t(x) \end{pmatrix}_{(m+n)\times(m+n)}$, а достаточно выполнять действия только с первыми n строками.

Таким образом, на каждом шаге необходимо выполнить следующие действия:

— с помощью невырожденных линейных преобразований столбцов привести матрицу $C_t(0)$ к ступенчатому виду $C_t(0)\tau_t = (0|E)$;
— умножить полиномиальную матрицу $G_t(x)$ на матрицу $\tau_t D$;
— умножить полиномиальную матрицу $C_t(x)$ на $\tau_t D/x$.

Замечание 4. За счёт выбора матрицы $\tau_t D$ на каждом шаге ровно m столбцов матрицы $G(x)$ увеличивают свою степень. Именно это позволяет к определённому моменту набрать достаточное число столбцов, удовлетворяющих условию (7).

Оценим трудоёмкость алгоритма Копперсмита. На каждом шаге наиболее трудоёмким действием является умножение полиномиальных матриц $G_t(x)$ и $C_t(x)$ на скалярную матрицу перехода τ_t. Степень матрицы $G_t(x)$, очевидно, не превосходит t. Степень матрицы $C_t(x)$ не превосходит $L - t$, где $L = \deg C_0(x) = \deg H(x)$ — длина отрезка последовательности, для которой ищется правый порождающий многочлен. Таким образом, при $m \geqslant n$ трудоёмкость алгоритма имеет порядок $\mathrm{O}(L^2 m^3)$. При $L = \lfloor M/m \rfloor + \lfloor M/n \rfloor + \Delta$, как в алгоритме Видемана — Копперсмита нахождения решений СОЛУ, получаем трудоёмкость порядка $\mathrm{O}(M^2 m)$.

3. Вопросы реализации алгоритма Копперсмита на ЭВМ

Рассмотрим некоторые подходы, позволяющие эффективно реализовать алгоритм Копперсмита нахождения векторных аннулирующих многочленов матричной последовательности над полем $\mathrm{GF}(2)$ на современной 64-разрядной вычислительной технике. Будем считать, что размеры последовательности m и n кратны 64: $m = 64m'$, $n = 64n'$.

3.1. Представление данных

С точки зрения оптимизации реализации алгоритма на ЭВМ большую роль играет способ представления данных. Естественным выбором является представление, позволяющее упростить наиболее трудоёмкие операции, выполняемые на каждом шаге алгоритма. Кроме того, при использовании вычислителей кластерного типа необходимо минимизировать объём данных, которыми обмениваются узлы кластера на каждом шаге.

Наиболее трудоёмкими операциями, выполняемыми на каждом шаге алгоритма Копперсмита, являются:

— умножение (справа) матрицы $C_t(x)$ на матрицу $\tau_t D/x$;
— умножение (справа) матрицы $G_t(x)$ на матрицу $\tau_t D$.

Выделение нулевого коэффициента $C_t(0)$ полиномиальной матрицы $C_t(x)$ при любом её естественном представлении не составляет труда. Операция приведения $C_t(0)$ при небольших значениях m и n также не является трудоёмкой по сравнению с умножением матриц $C_t(x)$ и $G_t(x)$ на матрицу перехода τ_t. Вектор степеней $\delta_{t+1} = (\delta_{t+1,1}, \ldots, \delta_{t+1,m+n})$ вычисляется за $\mathrm{O}(m+n)$ операций.

Матрица перехода τ_t является скалярной матрицей, поэтому умножение полиномиальных матриц $C_t(x)$ и $G_t(x)$ на τ_t справа сводится к отдельному умножению каждой строки каждого коэффициента этих матриц на τ_t. Поэтому с точки зрения снижения трудоёмкости этой операции, а также с учётом использования 64-разрядной ЭВМ естественно представлять полиномиальные матрицы $C_t(x)$ и $G_t(x)$ в виде массивов из m и n (соответственно) полиномиальных строк (длины этих массивов соответству-

ют степеням матриц $C_t(x)$ и $G_t(x)$), каждую полиномиальную строку — в виде массива строк-коэффициентов, а каждую строку-коэффициент (длины $m+n=64(m'+n')$) — в виде массива из $m'+n'$ 64-разрядных машинных слов.

Умножение на x выполняется над последними $m=64m'$ столбцами матрицы $G_t(x)$ (соответственно деление на x — над первыми $n=64n'$ столбцами матрицы $C_t(x)$), поэтому эту операцию можно выполнить автоматически, записывая результат умножения строки-коэффициента на матрицу перехода τ_t со сдвигом: последние m' машинных слов результата умножения строки-коэффициента при x^k записываются в массив, соответствующий коэффициенту при x^{k+1} (для $C_t(x)$ — первые n' машинных слов результата умножения строки-коэффициента при x^k записываются в массив, соответствующий коэффициенту при x^{k-1}).

Предложенный способ хранения данных позволяет записывать результаты очередного шага алгоритма на место предыдущих результатов: $G_{t+1}(x)$ вместо $G_t(x)$, а $C_{t+1}(x)$ вместо $C_t(x)$, что снимает необходимость постоянного выделения дополнительных объёмов оперативной памяти и, в свою очередь, снижает нагрузку на системный диспетчер памяти.

3.2. Быстрое умножение полиномиальных матриц на матрицу перехода

Поскольку основной элементарной операцией на каждом шаге алгоритма Копперсмита является умножение большого числа строк длины $m+n$ на одну и ту же матрицу перехода τ_t размера $(m+n)\times(m+n)$, для снижения трудоёмкости всего алгоритма необходимо прежде всего оптимизировать эту операцию.

Один из эффективных методов умножения строки на матрицу небольших размеров связан с использованием так называемых lookup-таблиц. Идея эта используется давно [6, гл. 6]; она достаточно проста и заключается в следующем. Пусть необходимо реализовать умножение строки a длиной $8k$ битов на двоичную матрицу C размеров $8k\times 8k$. Будем считать, что строка a хранится в массиве байтов длины k. Представим массив a в виде суммы k байтовых массивов длины k, в которых только один байт может быть ненулевым: $a=\sum\limits_{i=1}^{k} a_i$, где $a_i=(0,\ldots,0,*,0,\ldots,0)\in\mathbb{Z}_{256}^k$ (ненулевой элемент — на i-й позиции). Тогда произведение строки на матрицу раскладывается в сумму k произведений: $aC=\sum\limits_{i=1}^{k} a_iC$. Если для матрицы C заранее вычислить все возможные произведения вида a_iC — а их число равно $256k$, — то умножение строки длины $8k$ битов на матрицу $8k\times 8k$ сводится к сложению k строк длины $8k$ битов.

В нашем случае роль матрицы C играет матрица перехода τ_t размером $(m+n)\times$ $\times(m+n)$ битов и $k=8(m'+n')$. Для реализации приведённого алгоритма необходимо составить таблицу из $2^{11}(m'+n')$ строк длины $m+n$ битов. Составление этой таблицы является предварительным этапом и выполняется один раз на каждом шаге алгоритма Копперсмита после вычисления матрицы перехода τ_t, поэтому вклад этой процедуры в общую трудоёмкость алгоритма ничтожно мал по сравнению с последующим использованием таблицы при умножении строк $C_t(x)$ и $G_t(x)$ на τ_t. Умножение одной строки на матрицу τ_t в этом случае выполняется за $k=8(m'+n')$ побитовых сложений массивов из $m'+n'$ 64-разрядных слов.

3.3. Многопоточная реализация алгоритма

Рассмотрим вопросы, связанные с параллельным выполнением алгоритма Копперсмита несколькими потоками на многоядерной/многопроцессорной ЭВМ с общей памятью.

Наиболее трудоёмкими операциями алгоритма являются операции умножения полиномиальных матриц $C_t(x)$ и $G_t(x)$ на скалярную матрицу перехода τ_t, причём эти операции выполняются независимо для различных строк матриц $C_t(x)$ и $G_t(x)$, а также для матриц-коэффициентов при различных степенях x. Это предоставляет практически неограниченные возможности для распараллеливания данных операций.

При организации многопоточных вычислений применительно к рассматриваемому случаю необходимо учитывать следующие особенности:

— выбранный способ представления данных (строки матриц $C_t(x)$ и $G_t(x)$ представляются в виде отдельных массивов скалярных строк-коэффициентов при $1, x, x^2, \dots$);
— число строк матриц $C_t(x)$ и $G_t(x)$ равно, как правило, $m = 64m'$ и $n = 64n'$ соответственно, где m' и n' — натуральные числа;
— число ядер в современных ЭВМ с серийными процессорами составляет порядка 12–16 ядер на 1 процессор.

С учётом сказанного, достаточно реализовать параллельное умножение различных строк матриц $C_t(x)$ и $G_t(x)$, так как их общее количество даже при $m' = n' = 1$ в разы превышает возможное число вычислительных ядер. В этом случае многопоточная реализация одного шага алгоритма Копперсмита выглядит следующим образом:

1) управляющий поток вычисляет матрицу $C_t(0)$, а также матрицу перехода τ_t;
2) каждый рабочий поток выполняет умножение выделенных ему строк матрицы $C_t(x)$ ($G_t(x)$) на $\tau_t D/x$ ($\tau_t D$);
3) управляющий поток вычисляет вектор степеней δ_{t+1} и дожидается завершения умножения строк матриц $C_t(x)$ и $D_t(x)$ всеми рабочими потоками.

Важно отметить, что при большой длине L входной матричной последовательности трудоёмкость вычисления матрицы перехода τ_t и вектора степеней δ_{t+1} ничтожно мала по сравнению с умножением матриц $C_t(x)$ и $G_t(x)$ на τ_t, поэтому «простой» рабочих потоков является незначительным.

При использовании большего числа вычислительных ядер — скажем, порядка 1000 — вычисления отдельных строк матриц $C_{t+1}(x)$ и $G_{t+1}(x)$ можно также распределять по нескольким ядрам, разделяя между ними вычисления по диапазону степеней x. Например, при умножении $G_t(x)$ на матрицу перехода τ_t каждый коэффициент каждой строки $G_t(x)$ умножается на τ_t независимо от остальных: если разделить $G_t(x)$ по строкам $G_t(x) = (G_{t,1}(x), \dots, G_{t,n}(x))^T$, а каждую строку $G_{t,i}(x)$ по степеням x: $G_{t,i}(x) = \sum\limits_{j=0}^{t} G_{t,i,j}x^j$, то умножение выполняется отдельно для каждого коэффициента при различных степенях x: $G_{t,i}(x)\tau_t = \sum\limits_{j=0}^{t}(G_{t,i,j}\tau_t)x^j$. В этом случае необходимо лишь учитывать перекрытия на границах диапазонов степеней x, выделенных различным вычислительным ядрам, поскольку одновременно с умножением на τ_t выполняется умножение последних m столбцов $G_t(x)\tau_t$ на x, что достигается за счёт записи результата умножения на τ_t со сдвигом на одну степень x; аналогичным образом выполняется деление первых n столбцов $C_t(x)\tau_t$ на x (см. п. 3.1).

Практически единственным препятствием для эффективного использования большого количества вычислительных ядер при реализации алгоритма Копперсмита яв-

ляется большое число обращений к оперативной памяти. Действительно, с вычислительной точки зрения на каждом шаге выполняются простейшие операции, связанные с умножением строки на матрицу, а при использовании lookup-таблиц это умножение сводится к побитному сложению предварительно вычисленных массивов машинных слов. В то же время выполняется полное чтение и перезапись больших по объёму матриц $G_t(x)$ и $C_t(x)$. Другими словами, основными операциями являются чтение из памяти и запись в память. Поскольку современные модули памяти не могут обрабатывать запросы от нескольких ядер одновременно, распараллеливание алгоритма на большое число ядер становится неэффективным.

Одним из способов решения этой проблемы является использование вычислителей, в которых отдельные модули памяти соответствуют различным процессорам. В этом случае ядра одного процессора обращаются к своей доли оперативной памяти и выполнение алгоритма происходит более эффективно. Частным случаем является использование вычислителей кластерного типа.

3.4. Реализация алгоритма на вычислителях кластерного типа

При организации высокопроизводительных вычислений обычно используются вычислители кластерного типа, или кластеры. Кластер — это разновидность распределённой вычислительной системы, состоящей из нескольких связанных между собой компьютеров (узлов) и используемой как единый ресурс. В качестве связующего элемента выступают Ethernet, InfiniBand или любые другие относительно недорогие сети. Распределение вычислений по отдельным узлам выполняет один или несколько выделенных управляющих узлов.

В отличие от обычных многопоточных реализаций, при реализации алгоритмов на кластерах необходимо учитывать отсутствие общей для всех узлов оперативной памяти. Тем не менее в случае распараллеливания алгоритма Копперсмита всё достаточно просто: каждый узел хранит свою порцию строк матриц $G_t(x)$ и $C_t(x)$, а один шаг алгоритма выглядит следующим образом:

1) управляющий узел получает от каждого рабочего узла часть строк матрицы $C_t(0)$, вычисляет матрицу перехода τ_t и рассылает её копии каждому рабочему узлу;
2) каждый рабочий узел выполняют умножение выделенных ему строк матрицы $C_t(x)$ ($G_t(x)$) на $\tau_t D/x$ ($\tau_t D$);
3) управляющий узел вычисляет вектор степеней δ_{t+1} и дожидается завершения вычислений на всех рабочих узлах.

Объём данных, передаваемых по сети на каждом шаге, минимален и составляет несколько килобайтов: $512m'(m'+n')$ байтов при пересылке матрицы $C_t(0)_{m\times(m+n)}$ и $512(m'+n')^2$ байтов при пересылке матрицы $(\tau_t)_{(m+n)\times(m+n)}$. Трудоёмкость приведения $C_t(0)$ к ступенчатому виду мала по сравнению с остальными вычислениями, поэтому простой узлов кластера при этой операции является несущественным.

На отдельных узлах кластера возможно дальнейшее распараллеливание вычислений по отдельным ядрам, как это описано в предыдущем пункте. Тем не менее из-за особенностей работы оперативной памяти распределение вычислений по большому количеству ядер внутри одного узла не является эффективным.

Необходимо также отметить, что на каждом шаге алгоритма используется весь объём данных, полученных на предыдущем шаге, поэтому выполнение алгоритма на нескольких не связанных между собой вычислителях не представляется возможным.

4. Результаты экспериментов

Приведём результаты экспериментальных исследований времени работы алгоритма Копперсмита. Реализация алгоритма выполнена на языке C++, для компиляции программного кода использован 64-разрядный компилятор среды разработки Microsoft Visual Studio 2008. В качестве вычислителя использована ЭВМ с процессором Intel Xeon (2 процессора 2,53 ГГц по 4 ядра каждый) под управлением 64-разрядной ОС Windows 7. В таблице представлено время работы алгоритма при различных размерах задачи. Рассматривалась последовательность, получаемая в качестве результата работы первого этапа блочного алгоритма Копперсмита при решении СОЛУ из M уравнений; m и n — параметры алгоритма; L — длина последовательности; threads — число потоков.

Время работы алгоритма

M	m	n	L	threads	Время, с
100 000	2	1	2443	1	71
100 000	2	1	2443	2	91
100 000	2	1	2443	4	90
100 000	2	1	2443	8	93
1 000 000	2	1	23537	1	6578
1 000 000	4	2	11818	1	12969
1 000 000	4	2	11818	2	8849
1 000 000	4	2	11818	4	5726
1 000 000	4	2	11818	8	4700

Из таблицы видно, что при небольших размерах задачи распараллеливание не приводит к ускорению работы алгоритма, а наоборот, замедляет его. Это связано с тем, что на каждом шаге алгоритма необходимо выполнить небольшой объём вычислений, и синхронизация потоков на каждом шаге отнимает время, сравнимое с временем вычислений. При увеличении размера входа видно, что параллельная реализация работает быстрее. Однако увеличение числа потоков в 2 раза не приводит к соответствующему снижению времени работы алгоритма из-за особенностей работы оперативной памяти.

5. Субквадратичные алгоритмы

Трудоёмкость алгоритма Копперсмита составляет $\mathrm{O}(m^3L^2)$ элементарных операций, где L — длина матричной последовательности, а m и n — размеры её элементов ($m \geqslant n$). В то же время предложено большое число алгоритмов построения векторных аннулирующих многочленов с асимптотически меньшей трудоёмкостью. Например, если поле P допускает быстрое преобразование Фурье (БПФ), то изложенный в работе [7] алгоритм решает эту задачу за $\mathrm{O}\big(m^3L\log^2 L\big)$ операций. В работе [5] предложена субквадратичная версия алгоритма Копперсмита — рекурсивный алгоритм Копперсмита — Томэ, который также использует БПФ и имеет трудоёмкость $\mathrm{O}\big(m^3L\log^2 L\big)$ операций.

Несмотря на то, что эти алгоритмы имеют в асимптотике меньшую трудоёмкость, при относительно небольших размерах входа алгоритм Копперсмита работает значительно быстрее, поскольку не имеет большой константы в формуле трудоёмкости. Кроме того, у алгоритма Копперсмита на порядок ниже требуемый объём оперативной памяти, что также является немаловажным.

Заключение

Рассмотренный в данной работе алгоритм Копперсмита используется для построения векторных аннулирующих многочленов для матричных последовательностей над

полем. В частности, этот алгоритм является составляющей частью блочного алгоритма Видемана — Копперсмита нахождения решений больших разреженных систем линейных уравнений над полем.

Проведены исследования, связанные с эффективной реализацией алгоритма Копперсмита на современной вычислительной технике. Предложен способ представления входных и промежуточных данных, позволяющий эффективно распараллеливать наиболее трудоёмкие операции каждого шага алгоритма. Рассмотрен подход к оптимизации основной операции — умножения полиномиальных матриц большой степени на скалярную матрицу — за счёт выполнения предварительных вычислений.

Отдельное внимание уделено многопоточной реализации алгоритма, а также выполнению алгоритма на вычислителях кластерного типа. Приведены экспериментальные результаты.

ЛИТЕРАТУРА

1. *Coppersmith D.* Fast evaluation of logarithms in fields of characteristic two // IEEE Trans. Inform. Theory. 1984. V. IT-30(4). P. 587–594.
2. *Montgomery P. L.* A block Lanczos algorithm for finding dependencies over GF(2) // EUROCRYPT'95. LNCS. 1995. V. 921. P. 106–120.
3. *Coppersmith D.* Solving linear equations over GF(2) via block Wiedemann algorithm // Math. Comp. 1994. No. 62(205). P. 333–350.
4. *Wiedemann D. H.* Solving sparse linear equations over finite fields // IEEE Trans. Inform. Theory. 1986. V. IT-32(1). P. 54–62.
5. *Thomé E.* Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm // J. Symbolic Comput. 2002. No. 33. P. 757–775.
6. *Ахо А., Хопкрофт Д., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
7. *Beckerman B. and Labahn G.* A uniform approach for the fast computation of matrix-type Padé approximants // SIAM J. Matrix Anal. Appl. 1994. No. 15(3). P. 804–823.

СВЕДЕНИЯ ОБ АВТОРАХ

АБРОСИМОВ Михаил Борисович — доцент, кандидат физико-математических наук, доцент Саратовского государственного университета им. Н. Г. Чернышевского, г. Саратов. E-mail: **mic@rambler.ru**

АГИБАЛОВ Геннадий Петрович — заведующий кафедрой защиты информации и криптографии Национального исследовательского Томского государственного университета, доктор технических наук, профессор, г. Томск. E-mail: **agibalov@isc.tsu.ru**

ГАНОПОЛЬСКИЙ Родион Михайлович — кандидат физико-математических наук, директор Центра коллективного пользования высокопроизводительных вычислений Главного вычислительного центра Тюменского государственного университета, г. Тюмень. E-mail: **rodion@utmn.ru**

КАМЛОВСКИЙ Олег Витальевич — доцент, кандидат физико-математических наук, ведущий научный сотрудник ООО «Центр сертификационных исследований», г. Москва. E-mail: **ov-kam@yandex.ru**

ЛАРИОНОВ Виталий Борисович — кандидат физико-математических наук, ведущий разработчик ООО «Атес Медика Софт», г. Москва.
E-mail: **VitalyBLarionov@yandex.ru**

ЛИПСКИЙ Валерий Борисович — доцент кафедры защиты информации и криптографии Национального исследовательского Томского государственного университета, кандидат технических наук, г. Томск. E-mail: **lipsky@mail.tsu.ru**

МОДЕНОВА Ольга Владимировна — аспирантка Саратовского государственного университета им. Н. Г. Чернышевского, г. Саратов. E-mail: **oginiel@rambler.ru**

МОНАХОВА Эмилия Анатольевна — доцент, кандидат технических наук, старший научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **emilia@rav.sscc.ru**

ПАНКРАТОВ Иван Владимирович — научный сотрудник Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **ivan.pankratov2010@yandex.ru**

ПАНКРАТОВА Ирина Анатольевна — доцент кафедры защиты информации и криптографии Национального исследовательского Томского государственного университета, кандидат физико-математических наук, доцент, г. Томск.
E-mail: **pank@isc.tsu.ru**

РАЦЕЕВ Сергей Михайлович — доцент кафедры информационной безопасности и теории управления Ульяновского государственного университета, г. Ульяновск.
E-mail: **RatseevSM@mail.ru**

РОМАНЬКОВ Виталий Анатольевич — доктор физико-математических наук, профессор, заведующий кафедрой информационных систем Омского государственного университета им. Ф. М. Достоевского, главный научный сотрудник Омского государственного технического университета, г. Омск. E-mail: **romankov48@mail.ru**

РЫЖОВ Александр Сергеевич — сотрудник лаборатории ТВП, г. Москва. E-mail: **ryzhovalexander@mail.ru**

ТИМЕРЯЕВ Тимофей Валерьевич — аспирант Уфимского государственного авиационного технического университета, г. Уфа. E-mail: **timeryaev@yandex.ru**

УРАКОВ Айрат Ренатович — кандидат физико-математических наук, доцент Уфимского государственного авиационного технического университета, г. Уфа. E-mail: **urakov@ufanet.ru**

ШУМИЛИН Андрей Викторович — ведущий научный сотрудник ОАО «НПО РусБИТех», г. Москва. E-mail: **a.shumilin@rusbitech.ru**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Ganopolsky R. M. **GENERATING FUNCTIONS FOR SEQUENCES OF CONNECTED COVERS NUMBERS.** Analytical expressions are obtained for generating functions of the sequences of numbers being the amounts of connected covers of a finite set by subsets having the fixed cardinalities and properties. Recurrence relations are found for this numbers.

Keywords: *cover, connected cover, finite set, subsets, combinatoric numbers, generating functions, connected graphs.*

Kamlovskii O. V. **DISTRIBUTION PROPERTIES OF SEQUENCES PRODUCED BY FILTERING GENERATORS.** The distributions of r-tuples in output sequences of filtering generators over finite fields are considered. Bounds for the number of a given r-tuple occurrences are proved. Also, bounds for cross-correlation coefficients are established, and conditions for stated sequences to be different are got.

Keywords: *filter generators, finite fields, linear recurring sequences, additive character sums.*

Larionov V. B. **ON SUPERSTRUCTURE OF CLASS OF k-VALUED QUASIUNIFORM FUNCTIONS.** Closed classes of multivalued functions are considered. The superstructure of the class of so called quasiuniform functions is studied. It is proved that, except the classes of the quasiselfdual functions and their interconnections, there are no other classes, which contain this class.

Keywords: *multivalued logic, lattice of closed classes, selfdual functions.*

Ratseev S. M. **ON EXPONENTS OF SOME VARIETIES OF LINEAR ALGEBRAS.** The algebra UT_s of upper triangular matrices of a size s is considered. The equivalent conditions for the growth estimation are obtained for subvarieties in $var(UT_s)$, for varieties of Leibnitz algebras with nilpotent commutant, and for varieties of Leibniz — Poisson algebras with the identities $\{\{x_1, y_1\}, \ldots, \{x_n, y_n\}\} = 0$, $\{x_1, y_1\} \cdot \ldots \cdot \{x_n, y_n\} = 0$.

Keywords: *variety of linear algebras, growth of a variety, exponent of a variety.*

Romankov V. A. **CRYPTANALYSIS OF SOME SCHEMES APPLYING AUTOMORPHISMS.** Some methods are given for cryptanalysis of encryption schemes and key establishment protocols based on a group (loop) algebra or on a graded algebra with multiplicative base and proposed by Rososhek; Mihalev et. al.; Mahalanobis, etc. These cryptosystems have a common feature that all of them (except the scheme of Mihalev) use automorphisms. Also, a cryptanalysis of the key exchange protocol proposed by Megreleshvili and Djindjihadze is given. An original approach is described to find a secret message or a shared key based on usual tools of linear algebra assuming that platform can be chosen as a finite dimensional algebra, e. g., a matrix algebra over a field. The approach does not suppose to find the secret automorphism used in protocol. A theoretical foundation of this approach and a series of attacks on some cryptosystems based on different generalizations of discrete logarithm and Diffie — Hellman's ideas to noncommutative groups are described by the author earlier. Here the approach is developed by presenting its new applications in cryptanalysis of different schemes and protocols.

Keywords: *cryptographic scheme, group algebra, loop algebra, matrix algebra, graded algebra, discrete logarithm, generalized discrete logarithm, Diffie — Hellman scheme, El Gamal protocol, automorphism.*

Shumilin A. V. **THE MAIN ELEMENTS OF THE MANDATORY ENTITY-ROLE DP MODEL OF ACCESS AND INFORMATION FLOWS CONTROL FOR POSTGRESQL DBMS USED IN THE SPECIAL-PURPOSE OPERATING SYSTEM ASTRA LINUX SPECIAL EDITION.** A mandatory entity-role DP model is suggested for access and information flows control in the PostgreSQL DBMS used in the special-purpose operating system Astra Linux Special Edition. Some new features differing it from the other models of like destiny are discussed.
Keywords: *computer security, DBMS, role DP model, Astra Linux.*

Abrosimov M. B., Modenova O. V. **CHARACTERIZATION OF GRAPHS WITH THREE ADDITIONAL EDGES IN A MINIMAL 1-VERTEX EXTENSION.** Oriented graphs whose minimal vertex 1-extensions have three additional arcs are described.
Keywords: *graph, minimal vertex extension, exact vertex extension, fault tolerance.*

Monakhova E. A. **ON A CONSTRUCTION OF QUADRUPLE CIRCULANT NETWORKS WITH THE MAXIMAL NUMBER OF NODES FOR ANY DIAMETER.** For undirected circulant networks, the maximization problem for the number of nodes under given degree and diameter of a graph is considered. A new lower estimate is obtained for the attainable number of nodes in the circulant graphs of dimension 4 and any diameter. Some new infinite families of circulants reaching this estimate are constructed. For graphs of these families, some analytical descriptions are given.
Keywords: *undirected circulant graphs, diameter, maximum order of a graph.*

Urakov A. R., Timeryaev T. V. **TWO PROBLEMS OF WEIGHTED GRAPHS APPROXIMATION AND THEIR SOLUTION ALGORITHMS.** Two approximation problems for weighted sparse graphs are considered. By the approximation error is meant the absolute value of the difference of the distances between the vertices in graph and the corresponding vertices in an approximation graph. The first problem is to minimize the approximation error under a given dimension of the approximation graph. The second problem is to minimize the dimension of the approximation graph under a given approximation error threshold. For both problems, their solution algorithms are proposed and their presentation in the form of a graph partitioning problem are presented.
Keywords: *graph approximation, graph partitioning, sparse graphs, problem complexity reduction.*

Agibalov G. P., Lipsky V. B., Pankratova I. A. **CRYPTOGRAPHIC EXTENSION AND ITS IMPLEMENTATION FOR RUSSIAN PROGRAMMING LANGUAGE.** Cryptographic extension of the Russian programming language LYaPAS called LYaPAS-T is presented. The extension concerns the size of operands and the set of elementary operations over them. It is caused by the need of trustworthy and effective soft and hard implementations of contemporary cryptographic algorithms in secure computer systems applied for the logical control of critically important objects such as cosmic systems, nuclear weapons, energetic plants, submarines, etc. A LYaPAS-T compiler generating a load module for operating system Linux, and the projects of a LYaPAS-T processor implementing LYaPAS-T in hardware and of a preprocessor translating LYaPAS-T programs to the executive code of the processor are presented too. It is also told that for a LYaPAS-T subset containing neither subprograms nor operations over complexes and long operands,

the architecture of the processor has been described in VHDL, tested by means of a computer simulation, and implemented in a programmable logical integrated circuit obtained with the help of a computer-aided design.
Keywords: *Russian programming language, LYaPAS-T, compiler, preprocessor, processor, hard implementation.*

Pankratov I. V. **ABOUT LINEAR AND AFFINE EQUIVALENCE OF SUBSTITUTIONS.** The results of computer experiments with a software implementation of algorithms for determining the linear and affine equivalence of substitutions are presented. The detailed description of the algorithms is given. Experimental operation time of the algorithms is compared with the theoretical complexity of the problem.
Keywords: *linear equivalence, affine equivalence, substitution.*

Ryzhov A. S. **IMPLEMENTING COPPERSMITH ALGORITHM FOR BINARY MATRIX SEQUENCES ON CLUSTERS.** This paper concerns implementation of Coppersmith algorithm, which allows to calculate vector generating polynomials. Data representation for binary matrix sequences is considered. Effective parallelization for multicore CPUs and clusters provided.
Keywords: *matrix sequences, Coppersmith algorithm.*

Журнал «Прикладная дискретная математика» включен в перечень ВАК рецензируемых российских журналов, в которых должны быть опубликованы основные результаты диссертаций, представляемых на соискание учёной степени кандидата и доктора наук, а также в перечень журналов, рекомендованных УМО в области информационной безопасности РФ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте vestnik.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также и правила подготовки рукописей статей в журнал.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*