ГЕНЕРАЦИЯ ПСЕВДОСЛУЧАЙНЫХ PIN-КОДОВ

Гибкий [1] *алгоритм построения десятичных* PIN-*кодов на основе псевдослучайных подстановок и инволютивных матриц*

1. Введение

PIN-код (***Personal Identification Number*** – личный идентификационный номер) обычно используется как пароль доступа к терминалу, с помощью него производится авторизация держателя кредитной карты. Алгоритм построения PIN-кодов реализует биективное (т.е. взаимно однозначное) отображение

$$\mathcal{E}_{mk} : A^n \longrightarrow A^n \qquad (1)$$

множества A^n n-разрядных слов в алфавите A на себя. Секретность отображения обеспечивается тем, что алгоритм выполняется под управлением секретного ключа mk. Непосредственно ключ mk в алгоритме не используется (т.е. остается в тени), но на его основе на этапе предвычислений формируются ключевые материалы (такие, как псевдослучайные подстановки и матрицы), которые затем используются в алгоритме. Задача вычисления PIN-кодов аналогична задаче построения криптографических симметричных блочных шифров, в которых конструируется отображение

$$\mathcal{E}_k : \mathbb{E}_2^n \longrightarrow \mathbb{E}_2^n, \qquad (2)$$

где $\mathbb{E}_2 = \{0, 1\}$, $\mathbb{E}_2^n = \mathbb{E}_2 \times \ldots \times \mathbb{E}_2$ (n раз) – множество двоичных блоков (слов) длины n, $n = 64, 128,\ 96, 192, 256$. В частности, если требуется построение 16-разрядных PIN-кодов в алфавите шестнадцатеричных цифр

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\},$$

то можно использовать любой 64-битовый шифр (ГОСТ 28147, DES и т.п.). В этом случае любой зашифрованный блок B состоит из 8 байтов, или из 16 полубайтов, каждый из которых имеет значение цифры из множества A. Таким образом, блок B будет искомым PIN-кодом.

В данной работе приводится алгоритм построения n-разрядных десятичных PIN-кодов, т.е. в алфавите $A \equiv \mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Для определенности рассматривается случай $n = 16$, однако алгоритм может быть обобщен на случай произвольного n. Отличие от алгоритмов блочных шифров состоит в том, что для блочных шифров характерно использование вычислений в конечных полях характеристики 2 (что, конечно, не исключает использование и суррогатных вычислений над другими алгебраическими системами). В нашем случае $\mathbb{Z}_{10}$ не может быть наделено структурой поля, а при использовании операций сложения и умножения по модулю 10 множество $\mathbb{Z}_{10}$ становится коммутативным кольцом с делителями нуля, в котором только элементы 1, 3, 7, и 9 имеют мультипликативные обратные. Тем не менее. при конструировании отображения (1) для $A \equiv \mathbb{Z}_{10}$ уместно использовать идеи, которые применяются в блочных шифрах [1].

2. Этап предвычислений

На этапе предвычислений формируются вспомогательные ключевые материалы – раундовые ключи, таблицы подстановок, инволютивные матрицы и т.п. Формирование этих материалов осуществляется с использованием датчика псевдослучайных байтов, который предварительно инициализируется с помощью главного ключа mk. В качестве датчика псевдослучайных байтов можно использовать, например, любой поточный шифр. Для определенности рассмотрим шифр $RC4$, разработанный Р. Ривестом [2]. RC4 популярен благодаря простоте его реализации и высокой скорости работы алгоритма, RC4 устойчив к дифференциальному и линейному методам криптоанализа. Шифр использует два массива байтов – память $S[0..255] = (S_0, S_1, \ldots, S_{255})$, из которой будут извлекаться псевдослучайные байты, и память $K[0..255] = (K_0, K_1, \ldots, K_{255})$ для записи ключа mk. Ключ mk может иметь длину l от 5 до 255 байтов.

---

[1] Гибкий криптоалгоритм конструируется в зависимости от секретного ключа и данных.

Инициализация RC4 (выполняется один раз):

1. (Инициализация начального состояния памяти S.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ S[i] := i;$

2. (Ключ $mk = (mk_0, \ldots, mk_{l-1})$ переписывается в массив K с повторениями, если $l <$ 255.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ K[i] := mk[i\ mod\ l];$

3. (Настройка датчика, инициализация глобальных переменных $iRC4$ и $jRC4$.)

$iRC4 := 0;\ jRC4 := 0;$

$j := 0;$

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ \{$

$j := (j + S[i] + K[i])\ mod\ 256;$ $S[i]$ и $S[j]$ меняются местами

$\}.$(Инициализация завершена.)

Текущее использование датчика:

$iRC4 := (iRC4 + 1)\ mod\ 256;$

$jRC4 := (jRC4 + S[iRC4])\ mod\ 256);$

$S[iRC4]$ и $S[jRC4]$ меняются местами;

$t := (S[iRC4] + S[jRC4])\ mod\ 256);$

$return(S[t]).$

($S[t]$ – байт, возвращаемый при обращении к функции $RC4$.)

Замечание. Возможные слабости алгоритма $RC4$ могут быть решены отбрасыванием начальной части ключевого потока. Надёжным считается отбрасывание первых $L = 256, 512, 768$ или 1024 байт. Число $256 \leq L \leq 1024$ можно рассматривать как дополнительный ключ, определяющий конкретную версию алгоритма.

Функция $RC4M$ ***получения псевдослучайного байта*** c ***в диапазоне*** $[0..k-1],\ k \leq 256$

$f := true;$

$y := 256 - (256\ mod\ k);$

$\boldsymbol{while}\ f\ \boldsymbol{do}\ \{$

$c := RC4;$ $\boldsymbol{if}\ c < y\ \boldsymbol{then}\ \{f := false;\ c := c\ mod\ k;\ return(c)\}$

$\}.$

2.1. Построение подстановок S_0, S_1, S_2 и S_3

Подстановка (перестановка) – это биективное (т.е. взаимно однозначное) отображение $\pi: M \longrightarrow M$ некоторого множества M на себя. Подстановка задается массивом $(\pi_0, \pi_1, \ldots, \pi_t)$, где $\pi_k \in M$ – образ элемента $k \in M$ при отображении π. Перемножаются подстановки по правилу: $\pi \circ \rho(x) = \pi(\rho(x))$.

Построение псевдослучайной подстановки на множестве $\{0, 1, \ldots, t\}$

Пусть $S = (s_0, s_1, \ldots, s_t)$ – список различных байтов, не превосходящих t (например, $s_i = i, i = 0, 1, \ldots, t$);

$R = (r_0, r_1, \ldots, r_t)$ – искомая подстановка.

Начальный шаг $i = 0$. Генерируем случайный байт $c \in \{0, 1, \ldots, t\}$. Полагаем $r_0 = s_c$. Исключаем s_c из списка S.

Очередной шаг $i = 1, 2, \ldots, t-1$. Пусть $(s_0, s_1, \ldots, s_{t-i})$ – текущее состояние списка S. Тогда генерируем случайный байт $c \in \{0, 1, \ldots, t-i\}$ и полагаем $r_i = s_c$. Исключаем s_c из списка S.

Другой алгоритм построения подстановки на множестве $\{\mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{t}\}$

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ S[i] := i;$

$f := true;$

$\boldsymbol{while}\ f\ \boldsymbol{do}$

$\{$

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}$

$\{$

генерируем случайные байты k и m;

$S[k]$ и $S[m]$ меняем местами

};

if каждый элемент массива S менял свою позицию

then

{

удаляем из S байты со значением $> t$;

$f := false$

}

}.

Одним из указанных способов строим подстановки S_0 и S_2 на множестве $\mathbb{Z}_{100} = \{0, 1, \dots, 99\}$.

Построение подстановок S_1 и S_3 на множестве $\mathbb{Z}_{100}$ на основе дискретных логарифмов.

Так как $p = 101$ – простое число, то множество $\mathbb{F}_{101} = \{0, 1, \dots, 100\}$ с операциями сложения и умножения по модулю 101 образует конечное поле. Его мультипликативная группа $\mathbb{F}_{101}^* = \{1, \dots, 100\} =< 2^k >$ циклическая с порождающими элементами 2^k, где н.о.д. $(k, 100) = 1$. Уравнение $i + 1 = 2^{x_i} (mod\ 101)$ имеет единственное решение $x_i \in \{100, 1, 2, \dots, 99\}$ для каждого $i = 0, 1, \dots, 99$. Массив чисел $x_i\ mod\ 100$ образует следующую подстановку S_1 на множестве $\mathbb{Z}_{100}$:

50, 00, 01, 69, 02, 24, 70, 09, 03, 38, 25, 13, 71, 66, 10,
93, 04, 30, 39, 96, 26, 78, 14, 86, 72, 48, 67, 07, 11, 91,
94, 84, 05, 82, 31, 33, 40, 56, 97, 35, 27, 45, 79, 42, 15,
62, 87, 58, 73, 18, 49, 99, 68, 23, 08, 37, 12, 65, 92, 29,
95, 77, 85, 47, 06, 90, 83, 81, 32, 55, 34, 44, 41, 61, 57,
17, 98, 22, 36, 64, 28, 76, 46, 89, 80, 54, 43, 60, 16, 21,
63, 75, 88, 53, 59, 20, 74, 52, 19, 51.

Подстановку S_3 можно построить аналогично, выбирая другой порождающий элемент, например, $g = 2^{13} = 11\ (mod\ 101)$. При этом, чтобы внести в подстановку элемент случайности можно выбрать порождающий элемент $g = 2^k\ (mod\ 101)$, где $k \in \mathbb{Z}_{100}$ – случайный элемент, взаимно простой с 100. Однако поступим по-другому, в качестве подстановки возьмем массив $y_i = k_1 x_i + k_2 (mod\ 100)$, $i = 0, 1, \dots, 99$, где $k_1, k_2 \in \mathbb{Z}_{100}$ – случайные элементы, причем н.о.д. $(k_1, 100) = 1$. Например, при $k_1 = 17$ и $k_2 = 23$ получаем следующую подстановку S_3:

73, 23, 40, 96, 57, 31, 13, 76, 74, 69, 48, 44, 30, 45, 93,
04, 91, 33, 86, 55, 65, 49, 61, 85, 47, 39, 62, 42, 10, 70,
21, 51, 08, 17, 50, 84, 03, 75, 72, 18, 82, 88, 66, 37, 78,
77, 02, 09, 64, 29, 56, 06, 79, 14, 59, 52, 27, 28, 87, 16,
38, 32, 68, 22, 25, 53, 34, 00, 67, 58, 01, 71, 20, 60, 92,
12, 89, 97, 35, 11, 99, 15, 05, 36, 83, 41, 54, 43, 95, 80,
94, 98, 19, 24, 26, 63, 81, 07, 46, 90.

2.2. ***Инволютивные матрицы над кольцом $\mathbb{Z}_{100}$ целых чисел с операциями сложения и умножения по модулю*** **100.**

Кольцо $\mathbb{Z}_{100}$ является коммутативным кольцом с делителями нуля. Мультипликативными обратными b^{-1} обладают только те $b \in \mathbb{Z}_{100}$, которые взаимно просты с числом 100, т.е. 40 нечетных чисел, не кратных 5. При этом $b^{-1} = b^{19}\ mod\ 100$. Инволютивная матрица совпадает со своей обратной. Например, матрицы

$$M_{ab} = \begin{pmatrix} a & b \\ b^{-1}(1 - a^2) & -a \end{pmatrix}, \qquad (3)$$

где $a, b \in \mathbb{Z}_{100}$, н.о.д. $(b, 100) = 1$, являются инволютивными. Пусть

$$M = diag\left(M_{ab}^{(0)}, M_{ab}^{(1)}, M_{ab}^{(2)}, M_{ab}^{(3)}\right),$$

где $M_{ab}^{(i)}$ – матрицы вида (3), обозначает квадратную матрицу порядка 8 с матрицами $M_{ab}^{(i)}$ по главной диагонали и нулями в остальных позициях, а V и W – соответственно нижняя и верхняя треугольные матрицы порядка 8, которые заполнены случайными элементами из $\mathbb{Z}_{100}$, причем диагональные элементы имеют мультипликативные обратные. Матрицы V и W являются невырожденными, следовательно, имеют обратные, которые нетрудно вычислить, применяя метод Гаусса-Жордана. Тогда матрица

$$A_8 = V\,W\,MW^{-1}V^{-1}$$

является квадратной инволютивной матрицей порядка 8.

2.3. Преобразования G и G^{-1} над элементами $a, b \in \mathbb{Z}_{100}$ под управлением ключей $k_1, k_2, k_3, k_4 \in \mathbb{Z}_{100}$ согласно 4-раундовой схеме Фейстеля

Преобразование G определяется согласно 4-раундовой схеме Фейстеля [1] следующим образом:

$G[k_1, k_2, k_3, k_4](a, b) \equiv$
{
$a := a \boxplus S_0[b \boxplus k_1];\ a \leftrightarrow b;$
$a := a \boxplus S_1[b \boxplus k_2];\ a \leftrightarrow b;$
$a := a \boxplus S_2[b \boxplus k_3];\ a \leftrightarrow b;$
$a := a \boxplus S_3[b \boxplus k_4]$
},

где $a \boxplus b$ − сложение a и b по модулю 100, а $a \leftrightarrow b$ обозначает, что a и b обмениваются значениями. Обратное преобразование G^{-1}, возвращающее a и b к исходным значениям, задается как

$G^{-1}[k_1, k_2, k_3, k_4](a, b) \equiv$
{
$a := a \boxminus S_3[b \boxplus k_4];\ a \leftrightarrow b;$
$a := a \boxminus S_2[b \boxplus k_3];\ a \leftrightarrow b;$
$a := a \boxminus S_1[b \boxplus k_2];\ a \leftrightarrow b;$
$a := a \boxminus S_0[b \boxplus k_1]$
},

где $a \boxminus b \equiv a + (100 - b)$ − вычитание по модулю 100.

3. Алгоритм генерации PIN-кода

Вход: $C = c_0 c_1 \dots c_{15} \in \mathbb{Z}_{10}^{16}$ − 16-разрядное десятичное число.

1. (Число C преобразуется в вектор $B = b_0 b_1 \dots b_7 \in \mathbb{Z}_{100}^{8}$ из 8 двухразрядных десятичных чисел.)
   $\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ b_i := c_{2i} + c_{2i+1} \cdot 10;$
2. (Начальное забеливание.)
   $\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ b_i := b_i \boxplus k_i;$
3. (4 раунда преобразований G над вектором B.)
   $\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 2\ \boldsymbol{do}$
   {
   $G[b_7, ke_{2i}, b_2, ke_{2i+1}ke_{2i+1}](b_0, b_1);$
   Циклический сдвиг B на два разряда вправо:
   $b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 := b_6 b_7 b_0 b_1 b_2 b_3 b_4 b_5$
   };
   $G[b_7, ke_6, d_2, ke_7](b_0, b_1);$
4. (Перемешивание путем умножение вектора B на инволютивную матрицу A_8.)
   $B := BA_8;$
5. (4 раунда преобразований G^{-1} над вектором B.)
   $\boldsymbol{for}\ i := 4\ \boldsymbol{to}\ 6\ \boldsymbol{do}$
   {

$G^{-1}[b_7, ke_{2i}, b_2, ke_{2i+1}](b_0, b_1)$;

Циклический сдвиг B на два разряда влево:

$b_0b_1b_2b_3b_4b_5b_6b_7 :=\ b_2b_3b_4b_5b_6b_7b_0b_1$

};

$G^{-1}[b_7, ke_{14}, b_2, ke_{15}](b_0, b_1)$;

6. (Заключительное отбеливание.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ b_i := b_i \boxminus k_i$;

7. (Вектор B преобразуется в 16-разрядное десятичное число $C = c_0c_1 \dots c_{15} \in \mathbb{Z}_{10}^{16}$.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}$

{

$c_{2i} \coloneqq b_i\ mod\ 10$;

$c_{2i+1} \coloneqq b_i\ div\ 10$

}.

Выход: $C = c_0c_1 \dots c_{15} \in \mathbb{Z}_{10}^{16}$ – 16-разрядный десятичный PIN-код.

4. Расписание ключей

После инициализации датчика псевдослучайных байтов и отбрасывания L байтов начальной части ключевого потока генерируются ключи

$$k_0, k_1, \dots, k_7,$$

затем генерируются ключи для преобразований G и G^{-1}:

$$ke_0, ke_1, \dots, ke_{15}.$$

Далее конструируются 4 матрицы M_{ab}, диагональная матрица M, нижняя и верхняя треугольные матрицы V и W, обратные к ним V^{-1} и W^{-1}, наконец, вычисляется матрица A_8.

Предложенный алгоритм обладает инволютивным свойством: его можно использовать для реализации обратного преобразования, если ключи ke_i, $i = 0, 1, \dots, 15$, используются при этом в обратном порядке.

5. Обобщение алгоритма

Алгоритм может быть обобщен для порождения n-разрядных десятичных PIN-кодов на случай, когда $n \geq 4$.

Вход: $C = c_0c_1 \dots c_{n-1} \in \mathbb{Z}_{10}^{n}$ – n-разрядное десятичное число.

1. (Начальное забеливание.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ c_i := (c_i + k_i)\ mod\ 10$;

2. (n раундов преобразований G над вектором C.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ n-2\ \boldsymbol{do}$

{

$b_0 := c_0 + c_1 \cdot 10$;

$b_1 := c_2 + c_3 \cdot 10$;

$G[ke_{4i}, ke_{4i+1}, ke_{4i+2}, ke_{4i+3}](b_0, b_1)$;

$c_0 := b_0\ mod\ 10$;

$c_1 := b_0\ div\ 10$;

$c_2 := b_1\ mod\ 10$;

$c_3 := b_1\ div\ 10$;

Циклический сдвиг C на один разряд вправо:

$b_0b_1 \dots b_{n-1} :=\ b_{n-1}b_0b_1 \dots b_{n-2}$

};

$b_0 := c_0 + c_1 \cdot 10$;

$b_1 := c_2 + c_3 \cdot 10$;

$G[ke_{4n-4}, ke_{4n-3}, ke_{4n-2}, ke_{4n-1}](b_0, b_1)$;

$c_0 := b_0\ mod\ 10$;

$c_1 := b_0\ div\ 10$;

$c_2 := b_1\ mod\ 10$;

$c_3 := b_1\ div\ 10$;

3. (Перемешивание путем умножение вектора C на инволютивную матрицу A_n.)

$B := BA_n$;

4. (n раундов преобразований G^{-1} над вектором C.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ n-2\ \boldsymbol{do}$

{

$b_0 := c_0 + c_1 \cdot 10$;

$b_1 := c_2 + c_3 \cdot 10$;

$G^{-1}[ke_{4i}, ke_{4i+1}, ke_{4i+2}, ke_{4i+3}](b_0, b_1)$;

$c_0 := b_0\ mod\ 10$;

$c_1 := b_0\ div\ 10$;

$c_2 := b_1\ mod\ 10$;

$c_3 := b_1\ div\ 10$;

Циклический сдвиг C на один разряд влево:

$b_0 b_1 \dots b_{n-1} := b_1 b_2 \dots b_{n-1} b_0$

};

$b_0 := c_0 + c_1 \cdot 10$;

$b_1 := c_2 + c_3 \cdot 10$;

$G^{-1}[ke_4, ke_3, ke_2, ke_1](b_0, b_1)$;

$c_0 := b_0\ mod\ 10$;

$c_1 := b_0\ div\ 10$;

$c_2 := b_1\ mod\ 10$;

$c_3 := b_1\ div\ 10$;

5. (Заключительное отбеливание.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ c_i := (c_i + (10 - k_i))\ mod\ 10$;

Выход: $C = c_0 c_1 \dots c_{n-1} \in \mathbb{Z}_{10}^{16} - n$-разрядный десятичный PIN-код.

6. Расписание ключей для обобщенного алгоритма

После инициализации датчика псевдослучайных байтов и отбрасывания L байтов начальной части ключевого потока генерируются ключи

$$k_0, k_1, \dots, k_{n-1} \in \mathbb{Z}_{10},$$

затем генерируются ключи для преобразований G и G^{-1}:

$$ke_0, ke_1, \dots, ke_{8n-1}.$$

Далее конструируются матрицы над $\mathbb{Z}_{10}$: инволютивные матрицы

$$M_{ab} = \begin{pmatrix} a & b \\ d & e \end{pmatrix},$$

$$d = (b^3(101 - a^2))\ mod\ 10;\ e = (10 - a)\ mod\ 10, a \in \mathbb{Z}_{10}, b \in \{1,3,7,9\}; \qquad (3)$$

Пусть

$$M = \begin{cases} diag\left(M_{ab}^{(0)}, M_{ab}^{(1)}, \dots, M_{ab}^{(m-1)}, M_{abc}^{(m)}\right), & \text{если } n = 2m+1, \\ diag\left(M_{ab}^{(0)}, M_{ab}^{(1)}, \dots, M_{ab}^{(m)}\right), & \text{если } n = 2m \end{cases}$$

где $M_{abc}^{(m)}$ – инволютивная матрица 3-го порядка, например,

$$M_{abc}^{(m)} = \begin{pmatrix} c & 0 & 0 \\ 0 & a & b \\ 0 & d & e \end{pmatrix}, \begin{pmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & e \end{pmatrix} \text{ или } \begin{pmatrix} a & b & 0 \\ d & e & 0 \\ 0 & 0 & c \end{pmatrix},$$

$$d = (b^3(101 - a^2))\ mod\ 10;\ e = (10 - a)\ mod\ 10;\ c \in \{1, 9\}, a \in \mathbb{Z}_{10}, b \in \{1,3,7,9\}.$$

Далее строим невырожденные нижнюю и верхнюю треугольные матрицы V и W над $\mathbb{Z}_{10}$ порядка n, обратные к ним матрицы V^{-1} и W^{-1}, наконец, вычисляем инволютивную матрицу порядка n

$$A_n = V\,WMW^{-1}V^{-1}.$$

Предложенный алгоритм обладает инволютивным свойством: его можно использовать для реализации обратного преобразования, если ключи ke_i, $i = 0, 1, \ldots, 8n - 1$, используются при этом в обратном порядке.

Литература

1. Панасенко С. П. ***Алгоритмы шифрования. Специальный справочник***. – СПб.: БХВ-Петербург, 2009.
2. Асосков А.В., Иванов М. А., Мирский А. А., Рузин А. В., Сланин А. В., Тютвин А. Н. ***Поточные шифры***. – М.: КУДИЦ-ОБРАЗ, 2003.
3. Сущевский Д.Г., Панченко О.В., Кугураков В.С. ***Современные криптосистемы и их особенности***. – Вестник Казан. технол. ун-та, 2015, т. 18, № 11, с. 194-198.
4. Кугураков В.С., Кирпичников А.П., Сущевский Д.Г. ***О генерации псевдослучайных PIN-кодов криптографическим методом***. – Вестник Казан. технол. ун-та, 2015, т. 18, № 17, с. 190-193.