ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ ЭФФЕКТИВНОСТИ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ, БАЗИРУЮЩИХСЯ НА КРИПТОГРАФИЧЕСКИХ АЛГОРИТМАХ RC5 И RC6*

Б. Я. Рябко, В. С. Стогниенко, Ю. И. Шокин
Институт вычислительных технологий СО РАН
Новосибирск, Россия
e-mail: vss@ict.nsc.ru

Making use of cryptographic algorithms RC5 and RC6 is considered as the generators of pseudo-random numbers. The algorithms RC5 and RC6 are briefly described. Practical recommendations for the application are given.

В период стремительного развития вычислительной техники и вычислительных технологий необычайно актуальными становятся разработка, поиск и исследования "хороших", ориентированных на новые технологии генераторов псевдослучайных чисел. "Случайно выбранные" числа широко используются во многих приложениях и применяются для самых различных целей: моделирования, программирования, численного анализа, выборки, компьютерных игр и пр. Например, фирма Intel генератор псевдослучайных чисел использует в процессорах Pentium III [1].

В практических приложениях наиболее широкое применение нашли три способа генерации псевдослучайных чисел: аппаратный, табличный и алгоритмический [2].

Аппаратный способ предполагает использование специальной электронной приставки — генератора (датчика). В качестве физического эффекта, лежащего в основе таких генераторов, обычно используются шумы в полупроводниковых приборах, реже — явление распада радиоактивных элементов. Как правило, такие устройства позволяют получать равномерно распределенные на интервале [0, 1] числа. Реализация аппаратного способа не требует никаких специальных вычислительных процедур, однако он не гарантирует качества последовательности (см. п. 2 настоящей статьи).

При табличном способе эталонную таблицу псевдослучайных чисел оформляют в виде отдельного файла, который помещают в оперативную или внешнюю память ЭВМ, и обращаются к нему по мере необходимости, но такой способ получения псевдослучайных чисел целесообразно использовать только при небольшом объеме таблицы.

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, грант №98–01–00772.

© Б. Я. Рябко, В. С. Стогниенко, Ю. И. Шокин, 2000.

Алгоритмический способ основан на формировании псевдослучайных чисел в ЭВМ с помощью специальных алгоритмов и реализующих их программ. Каждое псевдослучайное число вычисляется обращением к соответствующей процедуре. К достоинствам способа относятся: достаточность единственной проверки качества последовательности, возможность многократного воспроизведения одной и той же последовательности, отсутствие необходимости использования внешних устройств, малый занимаемый объем памяти. Однако то, что запас чисел последовательности ограничен ее периодом и велики затраты машинного времени, можно отнести к недостаткам этого способа.

Указанные преимущества алгоритмического способа формирования базовой последовательности псевдослучайных чисел определяют его наибольшую пригодность по сравнению с двумя другими способами. В настоящей статье в качестве генераторов псевдослучайных чисел рассматриваются два криптографических алгоритма.

При оценке алгоритмов генераторов псевдослучайных чисел для электронных вычислительных устройств в первую очередь рассматриваются *качество* и *быстродействие*. Для оценки качества последовательности разработано большое количество тестов [3]. Высокая скорость изначально закладывается при разработке алгоритмов с учетом возможностей современных вычислительных платформ. В последнее время многие авторы, например, [4] рекомендуют использовать криптографические алгоритмы в качестве датчиков псевдослучайных чисел, поскольку они хорошо "перемешивают" входные данные и делают их почти независимыми, а также обрабатывают информацию с очень высокой скоростью (4 такта на байт сообщения) [5].

В 1994 году впервые появилось описание криптографического алгоритма шифрования RC5, а в 1998 г. — алгоритма RC6*. По сравнению с традиционными алгоритмами генерации псевдослучайных чисел эти алгоритмы обладают многими достоинствами, в том числе высоким быстродействием. Они рекомендованы для генерации последовательности псевдослучайных чисел [4], но на сегодняшний день каких-либо статистических данных по применению генераторов псевдослучайных чисел на основе этих алгоритмов не опубликовано. Мы попытались восполнить этот пробел. Целью настоящей работы является программная реализация и статистический анализ генераторов псевдослучайных чисел, базирующихся на алгоритмах RC5, RC6, и составление рекомендаций по их применению.

1. Алгоритм RC5

Внимание многих исследователей привлек алгоритм шифрования RC5, разработанный профессором Р. Ривестом для лаборатории RSA Data Security [6]. Для алгоритма характерны следующие свойства:

— адаптивность к аппаратным и программным средствам, которая обеспечивается использованием примитивных вычислительных операций, обычно присутствующих на типичных микропроцессорах;

— высокое быстродействие, так как в базисных вычислительных операциях операторы работают на полных словах данных;

— адаптивность к процессорам различных длин слова. Количество бит в слове w входит в число параметров алгоритма;

[0]В 2000 г. алгоритм RC6 в числе пяти других вышел в финал конкурса на лучший криптографический алгоритм шифрования Advanced Encryption Standard (AES), объявленный в 1997 г. Национальным институтом стандартов и технологий (США).

— наличие параметра r — количества раундов (итераций), — отвечающего за "степень перемешивания". Пользователь может выбирать между более высоким быстродействием и перемешиванием;

— низкое требование к памяти, что позволяет реализовывать алгоритм на устройствах с ограниченной памятью;

— применение циклических сдвигов, зависимых от данных, с "переменным" количеством сдвигов (на современных микропроцессорах время использования не зависит от количества циклических сдвигов);

— простота.

В алгоритме RC5 используются только три примитивных операции (и их инверсии), которые непосредственно и эффективно поддерживаются большинством процессоров:

1. Двоичное сложение слов по модулю 2^w ("+" — символ операции). Обратная операция вычитание ("−" — символ операции).

2. Побитовое исключающее ИЛИ ($\oplus$ — символ операции).

3. Левый сдвиг слов, $x <<< y$ ("<<<" — символ операции сдвига), — циклический сдвиг влево слова x на количество позиций, заданное в y. Здесь y интерпретируется по модулю w так, чтобы только $\lg(w)$ бит младшего разряда y использовались для определения количества циклических сдвигов ($\lg(w)$ обозначает двоичный логарифм от w). Правый сдвиг обозначен соответственно $x >>> y$.

Алгоритм RC5 включает алгоритм расширения ключа и алгоритм шифрования.

Алгоритм расширения ключа. Алгоритм расширения ключа заполняет расширенную таблицу ключа S размером $t = 2(r + 1)$ произвольных двоичных слов, он использует две "волшебные" константы P_w и Q_w (кратные слову) и состоит из трех простых алгоритмических частей. Эти константы определены для произвольного w следующим образом:

$$P_w = \mathrm{Odd}((e - 2)2^w);$$

$$Q_w = \mathrm{Odd}((\theta - 1)2^w);$$

где $e = 2.718281828459...$ (основание натурального алгоритма); $\theta = 1.618033988749...$ (золотое сечение); $\mathrm{Odd}(x)$ — нечетное целое число, самое близкое к x. Для $w = 16$, 32 и 64 эти константы даны ниже в двоичном и шестнадцатеричном виде:

```
P16 = 1011011111100001 = b7e1
Q16 = 1001111000110111 = 9e37
P32 = 10110111111000010101000101100011 = b7e15163
Q32 = 10011110001101110111100110111001 = 9e3779b9
P64 = 1011011111100001010100010110001010001010111011010010101001101011
    = b7e151628aed2a6b
Q64 = 1001111000110111011110011011100101111111010010100111110000010101
    = 9e3779b97f4a7c15
```

Первый шаг расширения ключа должен копировать секретный ключ K $[0...b - 1]$ в таблицу L $[0...c - 1]$ для $c = [b/u]$ слов, где $u = w/8$.

Второй шаг расширения ключа должен инициализировать таблицу S для частной фиксированной псевдослучайной комбинации двоичных разрядов, используя арифметическую прогрессию, определенную константами P_w и Q_w. Этот шаг реализуется следующим фрагментом алгоритма:

```
S[0]=Pw;
for i=1 to t-1 do
     S[i]=S[i-1]+Qw;
```

Третий шаг расширения ключа должен смешать ключ пользователя в трех проходах посредством таблиц S и L, но из-за различных размеров таблиц бóльшая из них будет обработана три раза, а другая — большее количество раз. Представим этот шаг в псевдокоде:

```
i=j=0;
A=B=0;
do 3 * max(t,c) times:
     A=S[i]=(S[i]+A+B)«<3;
     B=L[j]=(L[j]+A+B)«<(A+B);
     i=(i+1)mod(t);
     J=(j+1)mod(c);
```

Алгоритм шифрования. Пусть блок входа размещен в двух w-битных регистрах A и B, расширение ключа уже выполнено так, чтобы таблица S $[0...t-1]$, состоящая из $t = 2(r+1)$ w-битных слов, была вычислена. Представим алгоритм шифрования в псевдокоде [6]:

```
A=A+S[0];
B=B+S[1];
for i=1 to r do
     A=((A⊕B)«<B)+S[2i];
     B=((B⊕A)«<A)+S[2i+1];
```

Результат размещается в регистрах A и B.

Легко заметить исключительную простоту этого алгоритма.

2. Алгоритм RC6

Эволюционным усовершенствованием алгоритма RC5 является алгоритм RC6, разработанный в соответствии с требованиями стандарта AES (Advanced Encryption Standard). В нем так же, как и в RC5, активно применяются зависимые от данных циклические сдвиги. К новым свойствам алгоритма RC6 относятся использование четырех рабочих регистров вместо двух и включение целочисленного умножения как дополнительной простой операции, что значительно увеличивает рассеивание, достигнутое за раунд (цикл), позволяет обеспечить большую защиту, меньшее количество раундов и увеличивает производительность.

Алгоритм RC6 продолжает традицию использования примитивных операций (типа циклических сдвигов), которые эффективно реализованы на современных процессорах с учетом того, что 32-разрядное целочисленное умножение также реализовано на большинстве процессоров. Целочисленное умножение является очень быстрой простой операцией “рассеивания” и позволяет в алгоритме RC6 вычислять количество циклических сдвигов так, чтобы оно зависело от всех разрядов другого регистра, а не только от его младших разрядов, как в алгоритме RC5. В результате алгоритм RC6 имеет намного более быстрое рассеивание. Это также позволяет увеличивать производительность, задавая меньшее количество раундов.

Подобно алгоритму RC5 алгоритм RC6 представляет собой полностью параметризованное семейство алгоритмов шифрования. RC6-$w/r/b$ работает на четырех модулях w-битных слов с использованием следующих шести базисных операций:

$a + b$ — сложение целого числа по модулю 2^w;
$a - b$ — вычитание целого числа по модулю 2^w;
$a \oplus b$ — побитовое исключающее ИЛИ w-битных слов;
$a \times b$ — целочисленное умножение по модулю 2^w;
$a <<< b$ — левый сдвиг слов a на $\lg(w)$ бит младшего разряда b;
$a >>> b$ — правый сдвиг слов a на $\lg(w)$ бит младшего разряда b.

Алгоритм шифрования. Алгоритм расширения ключа RC6 идентичен алгоритму расширения ключа RC5 и подробно изложен выше.

Алгоритм RC6 работает с четырьмя w-битными регистрами A, B, C, D, которые содержат как входные данные, так и результат на выходе. Первый байт входных данных помещен в младший байт A, а последний — в старший байт D. Обозначим через $(A, B, C, D) = (B, C, D, A)$ параллельное переназначение (перемешивание) регистров. Пусть $S\,[0, ..., 2r+3]$ — таблица w-битных ключей.

Представим алгоритм шифрования в псевдокоде [7]:

```
B=B+S[0]
D=D+S[1]
for i=1 to r do
     {
       t=(B×(2B+1))«<lgw
       u (D×(2D+1))«<lgw
       A=((A⊕t)«<u)+S[2i]
       C=((C⊕u)«<t)+S[2i+1]
       (A,B,C,D)=(B,C,D,A)
     }
A=A+S[2r+2]
C=C+S[2r+3]
```

3. Статистические критерии

Статистическая теория дает некоторые количественные критерии случайности. Воспользуемся одним из хорошо известных тестов, который, будучи наиболее полезным, одновременно легко реализуется на вычислительных машинах [3].

Критерий χ^2 ("хи-квадрат"), вероятно, самый распространенный из всех статистических критериев. Он применяется в форме, предложенной Кнутом [3]. Его преимуществом является то, что одни и те же табличные значения (таблица χ^2-распределения) используются при любом числе n независимых испытаний и любых вероятностях p_s. Единственной переменной является $v = k - 1$ (число "степеней свободы"), где категория k — число возможных исходов испытаний.

На самом деле величина V (статистика χ^2) подчиняется распределению χ^2 только приближенно, причем приближение тем лучше, чем больше n. С учетом рекомендаций [3] и возможностей используемой вычислительной техники взято число $a = 10000$ для каждой категории k (при равномерном распределении каждое число должно в среднем встретиться 10000 раз). И только для длин слова $b = 24$ и $b = 26$ при $k_{24} = 16777216$ и $k_{26} = 67108864$ соответственно $a = 1000$, что связано с ограниченными вычислительными ресурсами ($b = 24$ и $b = 26$ рассматривается только для алгоритма RC6). Число испытаний $n = ak$, а длина последовательности псевдослучайных чисел (в байтах) определяется по

формуле $L = (ab/8)k$. Для длины слова $b = \{1, 2, 3, 4, 5, 6, 7, 8, 16, 24, 26\}$ (в битах) число возможных исходов испытаний равно 2^b.

Для проведения исследования были написаны программы шифрования с использованием алгоритмов RC5 и RC6, тест χ^2. В качестве языка программирования выбран язык Java, так как его универсальность и распространенность позволяют выполнять программы на различных платформах без дополнительных установок.

С помощью критерия χ^2 три раза проверялись разные части ряда псевдослучайных чисел, получаемого на выходе программной реализации алгоритмов RC5 и RC6. На вход подавались числа: в первом случае — начиная с 0, во втором — с n, в третьем — с $2n$. Испытания проводились на длинах слов $b = \{1, 2, 3, 4, 5, 6, 7, 8, 16, 24, 26\}$ для раундов с 4 по 12 (только после четвертого раунда вывод алгоритма является случайным [8]). Слова длиной $b = \{1, 2, 3, 4, 5, 6, 7, 8, 16\}$ выбирались последовательно из "непрерывного" выходного потока псевдослучайных чисел, слова длиной $b = \{24, 26\}$ — из каждого выходного слова (32 бита) алгоритмов со старшей битовой позиции x_{31}, ..., x_{32-b}, биты x_{32-b}, ..., x_0 игнорировались. Для любой длины слова w вероятность $p_s = 2^{-w}$.

Сначала исследовали генератор псевдослучайных чисел согласно предложенной в работе [4] рекомендации. Алгоритм шифрования RC5-32/8/0 использовался без секретного ключа. На вход подавались числа 0, 1, 2, ..., n_L, чтобы генерировать последовательность псевдослучайных чисел, которую можно использовать в рандомизированном вычислении [4]. Однако тесты по критерию χ^2 показали, что при использовании предложенного алгоритма на выходе получаются "плохие" числа, т. е. значения χ^2 явно слишком велики (зарегистрировано отклонение от нормального распределения). В связи с этим возникла задача дальнейшего анализа алгоритма шифрования RC5 как основы для генератора псевдослучайных чисел. Проведенные исследования позволили найти алгоритм задания входных данных, при котором RC5 дает неплохие результаты (рис. 1). Представим этот алгоритм в псевдокоде:

```
for (i={0,n,2n}; i<{n,2n,3n}; I=I+2)
A=i;
B=I+1;
далее следует алгоритм шифрования RC5...
```

На рис. 1 приведены результаты проверки последовательностей псевдослучайных чисел по критерию χ^2, полученных с помощью алгоритмов шифрования RC5 и RC6. На трех разных участках для каждой полученной последовательности псевдослучайных чисел проводилась проверка для заданной длины слова. Исходя из этих данных, можно сделать следующие выводы относительно алгоритма RC5:

— этот алгоритм можно использовать для генерации псевдослучайных чисел при условии специального подбора входных данных;

— как генератор псевдослучайных чисел он зависит от r*;

— для данного алгоритма длина получаемой последовательности равна $64 \cdot 2^{16}$ бит.

Исследование алгоритма шифрования RC6 для генерации последовательности псевдослучайных чисел проводилось аналогично. Эксперименты с различными алгоритмами входных данных показали неплохие результаты по критерию χ^2. Однако в настоящей работе принят алгоритм, предложенный в работе [4], который позволяет использовать максимальную длину получаемой последовательности псевдослучайных чисел — $128 \cdot 2^{128}$ бит. Представим его в псевдокоде:

[0]Испытания показали, что для обоих алгоритмов при r, большем, чем указано в табл. 1, качество получаемой последовательности псевдослучайных чисел не улучшается.

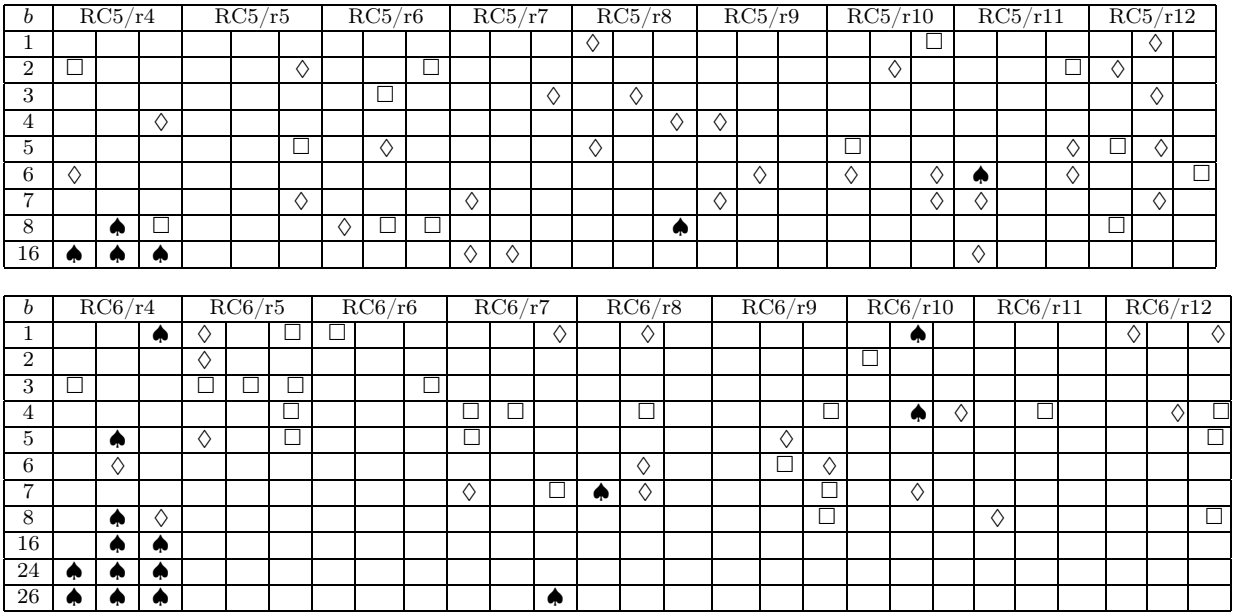

Рис. 1. Результаты испытаний алгоритмов шифрования RC5 и RC6 по критерию χ^2.

```
for (i={0,n,2n}; i<{n,2n,3n}; I=I+4)
{A,B,C,D}=i;
далее следует алгоритм шифрования RC6...
```

Криптографический алгоритм шифрования RC6, используемый для генерации псевдослучайных чисел, как и алгоритм RC5, зависит от r и для исследованных длин слов при $r = 6$ и $r = 11$ дает хорошие показатели по критерию χ^2. Дополнительные испытания поведения алгоритма при генерации многомерной последовательности проведены для $b = \{8, 16, 26\}$ и $r = 6$. При этом анализировалась плотность заполнения многомерного куба. В табл. 1 приведены результаты испытаний по критерию χ^2 с помощью алгоритма шифрования RC6, в ней отражена статистика равномерности заполнения при выбранной размерности x (первая графа) квадрата, куба, четырехмерного куба для длины слова 8, 16, 26 бит (первые, вторые и третьи строки соответственно).

Таблица 1

Результаты испытаний генерации многомерной последовательности по критерию χ^2

x	10	20	30	40	50	60	70	80	90	100
	103	392	993	1720	2570	3929	5431	6868	8501	11031
x^2	101	395	903	1700	2481	3649	4986	6220	8178	10264
	97	478	987	1669	2572	3813	5043	6449	8249	10309
	1139	8101	27984	65393	126188	219443	346991	516377	733823	1009271
x^3	967	7826	26883	64171	125382	216875	343892	513182	731504	1002365
	977	8177	27098	64693	124976	216398	343989	513510	731792	1000974
	10246	159744	811230	2562239	6254945	16248237	24000022	40961279	65615710	100026625
x^4	10217	160669	810346	2561106	6251952	12963800	24008500	40956500	65603520	99997464
	9681	159323	810460	2561489	6251869	12964883	24012199	40965239	65605048	100004235

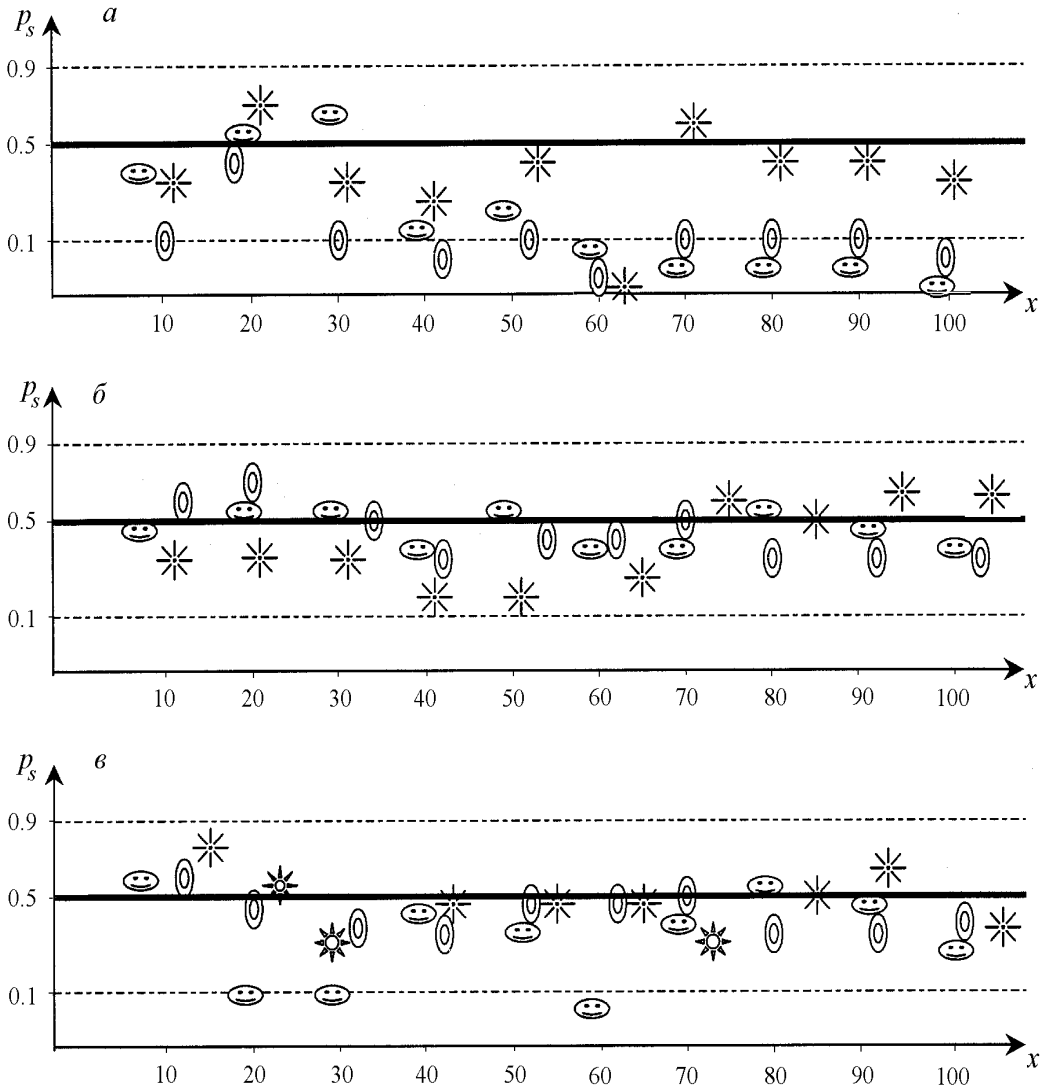

Рис. 2. Результаты испытаний генерации многомерной последовательности по критерию χ^2. — x^2, — x^3, — x^4, выбранное слово 8, 16, 26 бит — *а* – *в* соответственно.

Данные табл. 1 представлены в виде трех графиков (рис. 2), где по вертикальной оси отложена вероятность p_s появления полученного значения по критерию χ^2 и по горизонтальной — размерность многомерной последовательности. Каждый график отражает результаты испытаний для одной заданной длины слова по всем выбранным параметрам плотности заполнения многомерного куба, т. е. квадрата, куба и четырехмерного куба.

Результаты двух тестов для алгоритма RC6 позволяют сделать вывод о пригодности его использования как генератора псевдослучайных чисел.

4. Рекомендации

Проведенные исследования позволяют сделать вывод о том, что алгоритмы шифрования RC5 и RC6 можно применять в качестве генераторов псевдослучайных чисел для исследованных длин слов. Однако тестирование показало, что для алгоритма RC5 требуется специальный подбор входных данных. Временные характеристики алгоритма RC5 уступают характеристикам алгоритма RC6 (табл. 2). С помощью алгоритма RC6 можно получить

Таблица 2

Быстродействие алгоритмов шифрования RC6 и RC5

Алгоритм	Программная среда	Цикл/блок	Цикл/с	Мбайт/с
RC6	ANSI C	616	325 000	5.19
	Java (JDK)	16 200	12 300	0.197
	Java (JIT)	1010	197 000	3.15
	assembly	254	787 000	12.6
RC5	ANSI C	328	610 000	4.9
	Java (JIT)	1 140	175 000	1.4
	assembly	148	1 350 000	10.8

Данные округлены до трех значащих разрядов

последовательность псевдослучайных чисел значительно большей длины, чем с помощью алгоритма RC5. Выходная последовательность при одном обращении к алгоритму для RC6 (128 бит) в два раза больше, чем для RC5. Учитывая вышесказанное, рекомендуем для получения последовательности псевдослучайных чисел в качестве генератора использовать алгоритм шифрования RC6 с параметром $r = 6$ или $r = 11$. При выборе параметров алгоритма следует помнить, что число раундов влияет на количество проходов (перемешивание) и, следовательно, при большой выборке — на общее время счета.

Рассмотренные алгоритмы адаптированы для различных средств программирования за счет использования только примитивных вычислительных операций, благодаря чему они легко программируются. Выбор языка программирования индивидуален для каждого пользователя, и здесь нет никаких ограничений. Программная реализация алгоритмов RC5 и RC6 не представляется трудоемкой (алгоритм RC6 вместе с входными данными включает 12 строк простых операций на 128 бит выходной последовательности). При многократном использовании алгоритма с фиксированными начальными входными данными на выходе получаем фиксированную последовательность псевдослучайных чисел.

В статье дано полное описание алгоритма вычисления таблицы расширения ключа $S[0, ..., 2r+3]$, необходимое при работе алгоритма шифрования. Для реализации алгоритма RC6 в качестве генератора псевдослучайных чисел на языке Java для получения $S[\,]$ можно взять RC6Key.class на `http://cins.ict.nsc.ru/Class/RC6Key.class` и в свой программный код вставить следующие операторы:

```
int S[];
S=RC6Key.KeySchedule(r);
```

где r — заданное количество раундов.

Для иллюстрации временных характеристик алгоритмов RC5 и RC6 в зависимости от выбранного языка программирования приведем данные из работы [7].

Характеристики реализации алгоритма RC6 на оптимизированном ANSI C получены с использованием компилятора Borland C++ Development Suite 5.0. Измерения проводились на Pentium II 266 MHZ с 32 Mbytes RAM, под Windows 95. Для точности измерений временных характеристик маскируемые прерывания на процессоре были повреждены, в то время как установленные временные характеристики сохранялись.

Приведенные данные для реализации алгоритма RC6 на ассемблере получены на том же компьютере при идентичных условиях. Характеристики реализации алгоритма RC6 на Java измерялись на Pentium Pro 180 MHz с 64 Mbytes RAM, под Windows NT 4.0. Эта реализация компилировалась JavaSoft's JDK 1.1.6 компилятором, и характеристики результирующего кода измерялись JavaSoft's JDK 1.1.6 интерпретатором (с JIT поврежденной трансляцией) и Symantec

Corporation's Java! JustInTime Compiler, версия 210.054 для JDK. 1.1.2. Для достижения высокой точности измерений каждое множество временных тестов было выполнено 10 раз. В табл. 2 приведены не зависящие от размера ключа осредненные характеристики быстродействия алгоритмов (см. подробности в [7]).

Таким образом, алгоритм генерирования псевдослучайных чисел, базирующийся на использовании алгоритма шифрования RC6, обладает хорошим качеством и высокой скоростью и может быть рекомендован для практического применения.

Список литературы

[1] Спунер Дж. Intel выдвигает комплексный план защиты ПК // PC Week Online. 1999.

[2] Ферапонтов М. М. Моделирование случайных воздействий на ЭВМ. М.: Изд-во МТУ, 1995.

[3] Кнут Д. Искусство программирования для ЭВМ. М.: Мир, 1977.

[4] Rivest R. L. The RC5 Encryption Algorithm. http://www.rsa.com, 1998.

[5] Rogaway Phillip. Software-Optimized Encryption Algorithm // J. Cryptology. 1998. No. 11. P. 273–287.

[6] Rivest R. L. The RC5 encryption algorithm // Proc. 2nd Workshop on Fast Software Encryption. Springer, 1995. P. 86–96.

[7] Rivest R. L., Robshaw M. J. B., Sidney R. et al. The RC6^{TM} Block Cipher. Technology Square. 545. Cambridge, MA 02139, USA, 1998.

[8] Soto J., Basshам L. Randomness Testing of the Advanced Encryption Standard Finalist Candidates // Computer Security Division NIST.
`http://csrc.nist.gov/encryption/aes/aes_home.htm`, March 28, 2000.

Поступила в редакцию 26 апреля 1999 г.,
в переработанном виде — 28 июля 2000 г.