УДК 004.056

Н. Ф. Казакова, к.т.н., **Ю. В. Щербина**, к.т.н.

Одесский национальний економический университет, г. Одесса

ПРОБЛЕМЫ ОЦЕНКИ КАЧЕСТВА РАБОТЫ СОВРЕМЕННЫХ ЛИНЕЙНЫХ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Проведен анализ проблем, связанных с теоретическим и практическим обоснованием качества создаваемых генераторов случайных чисел, применяемых в криптографических приложениях и моделировании.

Ключевые слова: *качество, генератор ПСП, криптография, моделирование, многомерность.*

Постановка проблемы в общем виде и ее связь с важными научными и практическими задачами. В настоящее время, генераторы псевдослучайных последовательностей (ПСП) находят самое широкое применение в задачах криптографии и моделирования. Они включены практически во все стандартные программные библиотеки, которыми можно воспользоваться в случае необходимости. Однако, как показывают серьезные исследования, у пользователей часто возникают проблемы из-за дефектов в таких программных продуктах. Наличие этих дефектов обусловлено, во-первых, отсутствием научно обоснованного определения псевдослучайности, которым можно было бы руководствоваться при разработке генераторов и, во-вторых, различием подходов к их оценке и разработке.

Актуальной проблемой в настоящий момент является поиск компромиссного способа определения меры случайности, учитывающей многомерность распределения вероятностей в числах, формируемых генераторами ПСП и определение показателей, позволяющих судить о качестве предлагаемых алгоритмов.

К сожалению, исследователям и разработчикам пока еще не удалось выработать общепринятую точку зрения на способы исследования алгоритмов формирования ПСП. В реальной жизни каждая группа исследователей исповедует собственные взгляды на то, каким образом следует разрабатывать генераторы и как их оценивать. При этом зачастую применяемые на практике подходы к исследованию входят в противоречие с иными методами, решающими аналогичные задачи. Поэтому ***целью*** данной статьи является анализ и оценка существующих методов создания генераторов ПСП и выявление имеющихся в них дефектов. К сожалению, требования к генераторам формулируют не столько пользователи, сколько разработчики, которые их конструируют. Именно поэтому в общедоступных программных библиотеках, наряду с качественными генераторами ПСП, можно обнаружить много дефектных реализаций. Для решения поставленных задач было бы целесообразно объединять усилия заказчиков и разработчиков, однако, учитывая сложность и специфику задачи, на практике этого, как правило, не происходит.

Изложение основного материала. Первой и наиболее солидной работой посвященной анализу сформулированных проблем, принято считать второй том книги Д. Кнута «Искусство программирования» [1], которую большинство серьезных программистов называют «Библией информатики». В ней дан полный анализ и математическое обоснование большинства известных на момент ее публикации алгоритмов, которые сегодня используются как элементарные блоки для строительства сложных генераторов.

Первым простым и наиболее широко используемым генератором псевдослучайных последовательностей был предложенный Лехмером (Lehmer) около 1960 года, линейный конгруэнтный генератор (*Linear Congruential Generator* – LCG), формирующий рекурсивным способом последовательность целых чисел x_0, x_1,... от 0 до $N - 1$ по правилу – $x_{j+1} = ax_j + c \bmod N$ ($j = 0, 1, 2,...$).

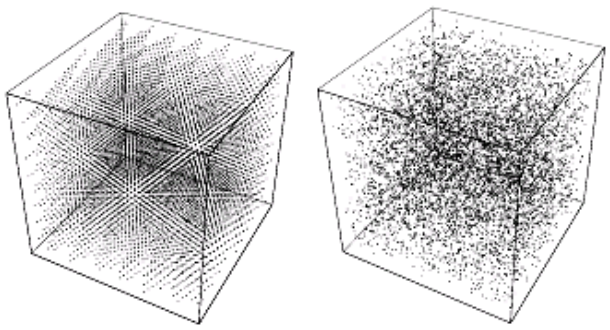
Значение x_0, с которого начинает работать генератор, является ключом к формируемой последовательности. Изменяя ключ, можно получать другие последовательности. Если на каком-либо шаге возникнет ситуация, при которой $x_j = x_{j+p}$, при некотором j и $p > 0$, эта ситуация будет повторяться бесконечно. Поскольку существует не более N возможных значений для x_j, период формируемой последовательности не превышает величины N.

Иногда пользователь нуждается в последовательности вещественных чисел, значения которых равномерно распределены в полуоткрытом интервале [0,1). Обычно они получаются путем преобразования по правилу

$$x_j \mapsto \frac{x_j}{N} \in [0,1).$$

Желательно, чтобы N было как можно больше, поскольку его величина определяет верхнюю границу периода, формируемой генератором последовательности. С другой стороны, проблема возникает при выполнении операции привидения по модулю N. Если величина N превышает максимальный размер машинного слова микропроцессора, эта операция становится слишком затратной с точки зрения вычислительных ресурсов. По этой же причине, чаще всего, величина N выбирается равной 2^{32}. В середине 80-х годов такой генератор вошел с состав С – пакета **rand** с параметрами a = 1103515245, c = 12 345, $N = 2^{31}$. Еще одной проблемой LCG является создание многомерной структуры распределения случайных точек для оценки равномерности распределения формируемых чисел. С целью контроля этой равномерности используется следующий алгоритм тестирования:

1. Генерируется три псевдослучайных числа в единичном полуинтервале [0, 1), которые затем используются как координаты точки в трехмерном единичном кубе.
2. Итерация пункта 1, повторяется 2^{31} раза.
3. В основном единичном кубе вырезаются подкубы со стороной, равной 0,0153 от стороны единичного куба, а затем изображается картинка с размещенными внутри такого фрагмента точками.



(а) (б)

Рисунок 1 – Иллюстрация равномерности распределения чисел, формируемых LCG

На рис. 1, (а) представлена структура решетки, образованной разделениями единичного куба, а на рис. 1, (б) – геометрическая интерпретация результатов тестирования LCG. Чем ближе результат к ожидаемому распределению, тем менее заметны различия между этими рисунками [1].

Сегодня LCG уже практически не отвечают современным требованиям, однако они, по-прежнему, присутствуют в распространенных пакетах прикладных программ.

В 1973 году Льюисом (Lewis) и Пауни (Payne) [2] впервые были предложены генераторы, создаваемые на основе регистров сдвига с обобщённой обратной связью (*generalized feedback shift register* – GFSR). Также как и LCG, они широко применяются, реализованы в большинстве прикладных программных пакетов, однако, точно также у них имеются свои дефекты. Идея такого генератора состояла в том, чтобы увеличить длину периода формируемой последовательности, поскольку от этого зависит ее случайность. С этой целью значение выходного символа x_j следовало сделать зависимым не только от предыдущего символа x_{j-1}, но и еще от одного символа x_{j-m}, сформированного ранее.

Если процессор вычислительной системы использует w-битные слова то все слова на выходе GFSR можно рассматривать как w-мерное распределение горизонтальных векторов с коэффициентами из множества {0, 1}. GFSR формирует псевдослучайную последовательность из слов x_0, x_1, x_2,… по рекуррентному правилу

$$x_j = x_{j-1} \oplus x_{j-k} \bmod p \ (j = 0, 1, \ldots), \qquad (2)$$

где $\oplus$ обозначает сложение «по модулю два» (сложение векторов с компонентами из двух элементов поля $\mathbf{F}_2$ = {0, 1}).

Такой генератор, обычно, дает последовательность, период которой превышает p.

Первые n w-битных слов x_0, x_1, x_2,..., x_{n-1} используются в качестве некоторого начального значения. Если целые числа $n > m > 0$ выбраны так, что $t^n + t^m + 1$ представляют собой «примитивный» многочлен над $\mathbf{F}_2$ [3], то верхняя оценка периода формируемой таким генератором последовательности достигает $2^n - 1$. В часто используемых программных приложениях величина n изменяется в пределах $60 \le n \le 1000$. Реализация такого генератора требует сохранения в памяти n последних слов, являющихся частью рекурсии и, следовательно, необходимый объем памяти составляет именно n слов.

При относительно большом периоде, для создания одного выходного слова GFSR требуется всего несколько инструкций процессору. Этим объясняется факт широкого применения подобных генераторов, однако проводимые исследования показали, что соотношение между числом 0 и 1 в выходной последовательности

существенно отклоняется, а это противоречит постулатам Голомба [3]. Если выбрать какой-либо бит в выходных словах, например, самый старший бит (*Most Significant Bit* – MSB) и зафиксировав целое *N*, разделить выходную последовательность на кортежи, содержащие *N* слов, то, для хорошей последовательности распределение единиц в таких кортежах будет соответствовать биномиальному распределению *B* (*N*, 1/2).

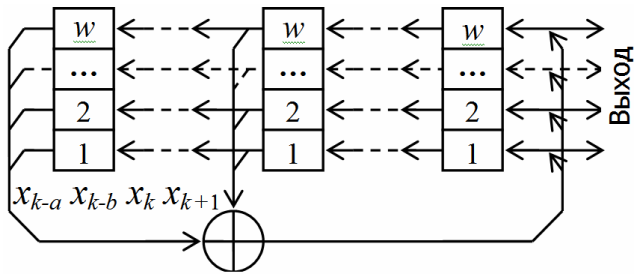


Рисунок 2 – Генератор Фибоначчи с запаздыванием

Однако исследования показывают, что при $N \leq n$ это условие не выполняется. Это происходит по причине того, что в процессе формирования выходных слов суммируются по модулю два только два из *n* слов, которые были сформированы ранее. Проще говоря, с учетом работы сумматора по модулю два, преобладание на некотором участке единиц или нулей не «сглаживается».

В 1958 году Дж. Ж. Митчеллом (G. J. Mitchell) и Д. Ф. Муром (D. P. Moore) был предложен модифицированный алгоритм Фибоначчи, работающий по правилу

$$x_n = (x_{n-24} \oplus x_{n-55}) \bmod p, \ (n \geq 55,\ldots).$$

Числа 24 и 55 выбраны таким образом, младшие значащие двоичные разряды имели период повторения, равный $2^{55} - 1$. Точная длина периода выходной последовательности определяется выражением

$$T = 2^{l-1}(2^{55} - 1) \text{ при } p = 2^l.$$

Первым тестом, который применяют для оценки соответствия распределения символов в словах на выходе генератора биномиальному закону *B* (*N*, 1/2) распределения является тест χ-квадрат. В тестовом пакете NIST этот тест называют «частотным тестом» [4]. Если этот тест не выполняется, проведение остальных тестов считается нецелесообразным. Многочисленные исследования показали, что данный тест для GFSR не выполняется [5 - 7]. Однако, не смотря на то, что этот дефект был обнаружен более 30 лет назад, пользователи зачастую не обращают на этот факт должного внимания. Из-за отсутствия научного определения практической псевдослучайности, в различных разработках генераторов находят воплощение «догмы» самих разработчиков. Зная о наличии дефектов, они пытаются частично их нейтрализовать путем «косметических» модификаций, однако полностью удалить их не удается, как это оказалось в случае с алгоритмом Фибоначчи.

Среди пользователей распространено мнение, что если необходимо получить не более 10^7 случайных чисел, то можно воспользоваться реальным источником, а не PRNG. Это мнение не кажется бесспорным. Во-первых, это практически не применимо в криптографии, из-за невозможности многократного воспроизводства такой ПСП. И, во-вторых, потому, что, при всех упомянутых негативных моментах, хорошие GFSR все же существуют. Например, в настоящее время, широко известен генератор, под названием «Вихрь Мерсенна» (Mersenne Twister), предложенный Макото Мацумото (Matsumoto M.) и Такудзи Нисимурой (Nishimura T.) в 1997 году [8]. Этот генератор имеет огромный период, равный числу Мерсенна $2^{19937} - 1$, и относится к классу, так называемых, витковых генераторов на регистрах сдвига с обобщенными обратными связями (*twisted generalized feedback shift register* – TGFSR). По сути это модификация генератора Фибоначчи с запаздыванием. Его упрощенная схема приведена на рис. 3.

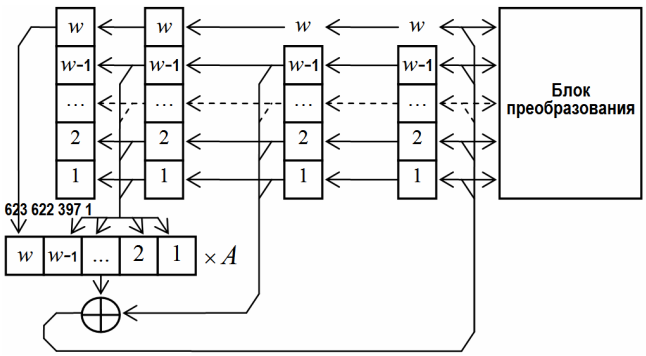


Рисунок 3 – Генератор MT19937

Существуют несколько вариантов этого алгоритма, различающихся размером используещегося простого числа Мерсенна. Среди них наибольшее распространение получил MT19937. Он состоит из 623 ячеек памяти, в каждой из которых хранится целое 32 битное число. При этом рекуррентная последовательность выходных слов, формируемых по правилу: $x_i = x_{397-i} \oplus ((x_{i-624}$

& 0×80000000) | ($x_{i\text{-}623}$ & 0×7f f f f f f f)) A (i = 0, 1 2, …).

Суть этого преобразования заключается в том, что на каждом i-том шаге выбирается старший бит из слова $x_{i\text{-}623} = x_{w-1}, x_{w-2},..., x_0$, ($w$ = 32), принятого на предыдущем 623-ем шаге, и 31 бит из слова $x_{i\text{-}622}$, принятого на предыдущем 622-ом шаге, а затем осуществляется конкатенация обеих полученных частей с последующим умножением полученного результата на матрицу

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 \\ a_{w-1} & a_{w-2} & \dots & \dots & a_0 \end{pmatrix}.$$

При этом умножение выполняется по правилу:

$$xA = \begin{cases} \text{schiftrigth}(x), & (\text{if the last significant bit } x = 0); \\ \text{schiftrigth}(x) \oplus a, & (\text{if the last significant bit } x = 1), \end{cases}$$

где a – это постоянный вектор: $a = (a_{31}, a_{30},..., a_0)$ = = 0×9908B0DF в шестнадцатеричном исчислении.

Форма и значения элементов матрицы A была выбрана авторами из условия обеспечения максимальной скорости выполнения умножения x A.

Далее, результат этого преобразования посимвольно складывается по модулю два со словом, принятого на предыдущем 397-ом шаге, после чего содержимое всех ячеек сдвигается на шаг влево и полученный результат записывается в младшую ячейку. После этого, в зависимости от модификации генератора, часть полученного слова подвергается дальнейшему преобразованию и поступает на выход генератора. Цель выходного преобразования сводится к дальнейшей рандомизации выходного пока.

Для испытания MT19937 использовался пакет тестов DIEHARD [9], результаты тестирования оказались вполне удовлетворительными. Обладая впечатляющим периодом, этот генератор обеспечивает хорошую равномерность распределения вероятностей выходной последовательности символов. Еще одним его достоинством является наличие в алгоритме таких наиболее экономичных операций, как операции сдвигов и суммирования. Генератор разрабатывался для нужд моделирования и, к сожалению, не считается криптографически стойким. Тем не менее, он, наряду с другими, рассмотренными выше линейными генераторами, может использоваться для нужд криптографии в качестве составной части.

Т. о., в **заключение** следует отметить, что на текущий момент существует достаточно много инженерных решений, позволяющих создать алгоритм формирования псевдослучайных чисел, удовлетворяющий заданным требованиям. Однако существуют серьезные проблемы теоретического характера, которые вызывают существенные трудности в этой деятельности, и они сводятся к ответу на следующие вопросы:

- *как формировать случайные числа?*
- *как обеспечить их случайность в выходном потоке?*

К сожалению, на эти вопросы полноценных ответов пока нет и, вероятно, не будет в будущем. Одной из причин такого положения дел является отсутствие достаточного практического и разумного определения псевдослучайности. В свое время, Колмогоров определил случайную последовательность, как такую последовательность, для генерации которой необходим алгоритм, длина которого не менее длины самой последовательности. Однако практическая значимость этого определения для разработчиков PRNG мала. Поэтому они проектируют генераторы, имеющие большой период, а, уже затем, ищут способы рандомизации выходного потока. Существует еще статистическое определение псевдослучайности, но полное удовлетворение его требований, представляет собой неподъемную задачу для современных ЭВМ.

Основой для предварительной теоретической оценки проектируемого генератора может быть рекуррентное аналитическое выражение, определяющее порядок формирования выходной последовательности. Одним из показателей, который может быть из него вычислен, является период. Подлинно случайные последовательности бесконечны. Но величина периода никак не отображает характер распределения символов. Поэтому, на данный момент, для определения качества выходной последовательности остается только статистическое тестирование. Известно множество строгих тестов, которые по различным показателям подтверждают (или не подтверждают) равномерность распределения вероятностей выходных символов. В работе [1] Д. Кнут утверждает, что увеличение числа различных тестов лишь «укрепляет уверенность» разработчика в качестве проделанной работы, поэтому, чем

больше выполнено различных тестов, тем лучше.

Вывод. В работе рассмотрены подходы к созданию линейных рекуррентных алгоритмов, обеспечивающих высокую производительность формирования случайных чисел. Известно множество нелинейных алгоритмов, применяемых, в основном для криптографических целей, но они сложны и работают достаточно медленно.

Список использованных источников

1. Кнут, Д. Искусство программирования для ЭВМ : монография. Т.2 / Д. Кнут. – М. : Мир, 1977. – 727 с.

2. Lewis, T. G. Generalized feedback shift register pseudorandom number algorithms : монография / T. G. Lewis, W. H. Payne. – J. ACM 20, 1973. – 468 с.

3. Golomb, S. W. Shift Register Sequences : монография / S. W. Golomb. – San Francisco : Holden Day, 1967 (and also reprint : Aegan Park Press, 1982). – ISBN 978-3-540-44523-4, ISSN 0302-9743.

4. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications // NIST Special Publication 800-22. – May 15, 2001. – С. 117-123.

5. Matsumoto, M. Twisted GFSR Generators / M. Matsumoto, Y. Kurita // ACM Transactions on Modeling and Computer Simulation. – 1992. – № 2. – С. 179-194.

6. Lindholm, J. H. An analysis of the pseudo-randomness properties of subsequences of long m-sequences / J. H. Lindholm // IEEE Trans. Inform. – 1968. – Theory IT-14. – С. 569-576.

7. Fredricsson, S. A. Pseudo-randomness properties of binary shift register sequences / S. A. Fredricsson // IEEE Trans. Inform. – 1975. – Theory IT-21. – С. 115-120.

8. Matsumoto, M. Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator / M. Matsumoto, T. Nishimura // ACM Trans. on Modeling and Computer Simulation. – 1998. – № 8. – С. 3-30.

9. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness [Електронный ресурс] // Портал : без названия. – Режим доступа /www/ URL : http://www.stat.fsu.edu/ pub/diehard. – Заглавие с экрана, доступ свободный, 18.05.2013.

Поступила в редакцию 21.05.2013

Рецензент: д.т.н., доцент Скопа А. А., Одесский национальный экономичный университет, г. Одесса.

Н. Ф. Казакова, к.т.н., **Ю. В. Щербина,** к.т.н.

ПРОБЛЕМИ ОЦІНКИ ЯКОСТІ РОБОТИ СУЧАСНИХ ЛІНІЙНИХ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

Проведено аналіз проблем, пов'язаних з теоретичним та практичним обґрунтуванням якості створюваних генераторів випадкових чисел, застосовуваних у криптографічних додатках та моделюванні.

Ключові слова: *якість, генератор ПВП, криптографія, моделювання, багатовимірність.*

N. F. Kazakova, PhD, **Yu. V. Scherbina,** PhD

EVALUATION OF QUALITY PROBLEMS OF MODERN LINEAR GENERATOR PSEUDO-RANDOM SEQUENCE

The analysis of the problems of theoretical and practical study of quality of random number generators. Generators are designed for use in cryptographic applications and solutions for the problems of modeling.

Keywords: *quality, pseudo random number generator, cryptography, modeling, multidimensional.*