УДК 004.054; 004.056.55; 004.421.5

КАЗАКОВА Н.Ф.
кандидат технических наук
декан факультета Компьютерных наук и инновационных технологий
Международного гуманитарного университета (г. Одесса)

ЩЕРБИНА Ю.В.
кандидат технических наук, доцент
доцент кафедры Информационной безопасности
Международного гуманитарного университета (г. Одесса)

ЧАСТОТНОЕ ТЕСТИРОВАНИЕ КРИПТОГРАФИЧЕСКИХ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Аннотация. *Рассмотрены теоретические принципы частотного тестирования псевдослучайных криптографических последовательностей и способы их практической реализации.*

Abstract. *The theoretical principles of the frequency of testing of cryptographic pseudo-random sequences and methods of their implementation.*

Постановка проблемы в общем виде и ее связь с научными и практическими задачами. В настоящее время, одной из проблем, связанных с защитой, передаваемой в информационно-телекоммуникационных системах, конфиденциальности информации, является потребность в генераторах псевдослучайных последовательностей (ПСП), отвечающих высоким требованиям к равномерности распределения вероятностей формируемых ими чисел. Эти требования были сформированы специалистами NIST, которые в 1999 г. в рамках проекта AES (*Advanced Encryption Standard*) разработали набор статистических тестов NIST STS (*NIST Statistical Test Suite*) для испытания псевдослучайных последовательностей [1].

Датчики псевдослучайных чисел находят применение в криптографических протоколах для формирования ключей, при хешировании паролей, а также в алгоритмах, заложенных в основу поточных симметричных криптографических систем, применяемых для защиты конфиденциальности передаваемой информации. Построение таких датчиков – совершенно нетривиальная задача и ее решение требует кропотливого труда математиков и аналитиков. Генератор, слабый с криптографической точки зрения, может значительно ослабить защищенность информационной системы и, по этой причине, для разработчиков криптографических систем важно иметь в наличии средство проверки их надежности.

Анализ научной и технической литературы показывает, что за последние десятилетия было разработано и исследовано большое количество «элементарных» генераторов ПСП, к числу которых относят линейные конгруэнтные генераторы [4, 5], генераторы Фибоначчи с запаздыванием [4], генераторы, построенные на линейных регистрах с обратной связью [5…8] и некоторые их разновидности. Многолетние исследования привели к заключению о том, что все они не являются криптографически стойкими и могут входить в состав формирователей ПСП только в качестве составных элементов.

Как показано в [5], идея построения составного генератора базируется на том факте, что комбинация двух и более выходных последовательностей от генераторов разного типа с помощью таких операций как «+», «–», «×», «⊕», приводит генератору с «лучшими свойствами случайности». Наиболее удачные составные генераторы (с точки зрения криптографии) подробно рассмотрены в работе Б. Шнаера [9]. С течением времени в алгоритмах, казавшихся ранее надежными, находят новые «слабые места». По этой причине в процессе разработки криптографических протоколов встает вопрос о поиске новых инженерных решений с целью построения новых, эффективных криптографически стойких генераторов, свободных от выявленных недостатков.

Наиболее удачным на сегодняшний день формирователем ПСП является алгоритм, реализованный в поточном шифре RC4 [9], однако все чаще появляются сообщения о том, что и в нем уже найдены уязвимости. Утверждается, что удалось установить статистическую зависимость характера генерируемой ПСП от первых символов ключа.

В отличие от иных инженерных задач, разработка нового алгоритма формирования случайных чисел отличается тем, что достоверный ответ на вопрос об эффективности найденного решения задачи может появиться только спустя некоторое время, когда для него будет разработан индивидуальный метод криптоанализа. Разработчику остается довольствоваться только результатами предварительного тестирования. Об этом прямо сказано в руководстве к пакету тестов, разработанных NIST [1]. Любой из предложенных тестов или даже целый пакет тестов не заменяет криптоанализа. При этом предварительное тестирование является обязательным. Генератор, не удовлетворяющий условиям тестирования непригоден. Каждый из входящих в пакет тестов ориентируется на поиск определенного вида аномалий в потоке формируемых символов.

К числу наиболее рекомендуемых из известных тестовых пакетов, относится уже упоминавшийся пакет NIST STS. Он включает набор из 16-ти тестов и методику их использования. Успешный результат испытаний проектируемого генератора с применением всего набора этих тестов дает основания надеяться на то, что формируемая генератором последовательность неотличима от «настоящей» случайной последовательности.

Известны и другие пакеты тестов, созданные для нужд криптографии. К их числу относится набор статистических тестов под названием Diehard [2], предназначенный для определения качества последовательности случайных чисел. Эти тесты были разработаны Джорджем Марсальей (George Marsaglia). Он включает 12 тестов и доступен в Интернете по адресу: http://stat.fsu.edu/pub/diehard/.

По адресу http://www.isi.qut.edu.au/resources/cryptx/ можно связаться с разработчиками пакета тестов Crypt-X [3] и получить программное обеспечение и руководство по их применению.

Однако использование отмеченных пакетов прикладных программ наталкивается на ряд серьезных препятствий. Первое из них заключается в том, что они предназначены для оценки уже готовых генераторов. В практической же работе такие устройства разрабатывают поэтапно, постепенно доводя их до уровня соответствия предъявляемым требованиям.

Вторая проблема заключается в том, что в основе каждого из входящих в предлагаемый пакет тестов лежит достаточно сложное теоретическое обоснование, требующее от разработчиков серьезной математической подготовки и знаний различных несмежных разделов математики. К сожалению, в прилагаемых к тестам руководствах, разработчики такого обоснования, как правило, не приводят.

Наконец, третья проблема, заключается в том, что, хотя к программному обеспечению и предоставлен свободный бесплатный доступ, воспользоваться ним сложно. Большинство тестов предполагают предварительное создание файла, в который записывается испытуемая псевдослучайная последовательность в виде 32-х битных слов, а затем запускается процедура тестирования. Это удобно не всегда и не для всех тестов, поскольку требует значительных программно-аппаратных ресурсов. К тому же, предлагаемые тесты рассчитаны на определенную программно-аппаратную платформу.

Перечисленные проблемы вынуждают разработчиков если не разрабатывать собственные тесты, то, по крайней мере, создавать собственное программное обеспечение, которое их реализует, удобно в работе и может эффективно использоваться в процессе поиска конструктивного решения разрабатываемого генератора. Всякий пакет тестов имеет свою внутреннюю логику. Предполагается, что испытание нового генератора должно начинаться с частотного тестирования. Как указывается в [1], если генератор не проходит частотный тест, то проведение всех прочих тестов уже не имеет смысла. Поэтому, учитывая все вышесказанное, ***целью статьи*** *является анализ выполнения частотного тестирования и методики ее проведения.*

Тестирование генератора, в том числе частотное, основано на сравнении этого генера-

тора с идеалом. Предполагается, что такой идеальный генератор формирует последовательность с равномерным распределением вероятностей единиц и нулей, причем такую, что следующий выходной бит невозможно предсказать по результатам наблюдения некоторого отрезка этой последовательности с вероятностью, отличающейся от 1/2.

В действительности, реальный генератор ПСП выдает «ненастоящую» случайную последовательность, а полностью определяемую значением секретного ключа. Степень его схожести с реальным формирователем случайной гаммы может быть установлена на основании выбранного эталона и критерия, позволяющего определить степень отличия полученного результата от ожидаемого равномерного распределения вероятностей.

Формальное определение критерия предполагает задание нулевой гипотезы H_0, в соответствии с которой тестируемая последовательность является случайной. С ней непосредственно связана альтернативная гипотеза H_A в соответствии с которой эта последовательность не может быть признана случайной. Принимая нулевую гипотезу, экспериментатор с вероятностью α, рискует ошибиться – совершить так называемую «ошибку первого рода». Соответственно, с вероятностью $1 - \alpha$, он будет прав. Обычно, величину α выбирают в пределах $0{,}01 < \alpha < 0{,}001$.

При частотном тестировании предполагается, что появление символов на выходе генератора полностью подчиняется распределению Бернулли, при котором вероятность единичного символа p равна вероятности появления нулевого символа $q = 1 - p$. В этом случае разница между числом единиц n_1 и числом нулей n_0, $S_n = n_1 - n_0$ в n-разрядной последовательности составляет $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, где ε_i – двоичный символ, принимающий значения {0,1}. Разница будет подчинена биномиальному закону распределения вероятностей, который в соответствии с теоремой Муавра-Лапласа, при достаточно большом значении n, хорошо апроксимируется стандартным нормальным законом $N_{0,1}$ с нулевым математическим ожиданием и единичной дисперсией. И, как показано в [1,10], эта разница, в соответствии с центральной предельной теоремой, удовлетворяет условию:

$$\lim_{n\to\infty} P\left(\frac{S_n}{\sqrt{n}} < z\right) = \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{u^2}{2}} du .$$

Здесь $\Phi(z)$ представляет собой функцию Лапласа, численно равную площади фигуры, ограниченной сверху кривой Гаусса, снизу осью абсцисс, а справа прямой $y = z$. В [1] показано, что для положительных значений z, будет справедливо выражение:

$$P\left(\frac{S_n}{\sqrt{n}} \le z\right) = 2\Phi(z) - 1.$$

По данным испытания n-разрядной двоичной последовательности вычисляют статистику:

$$s_{obs} = \frac{|n_1 - n_0|}{\sqrt{n}} = \frac{|S_n|}{\sqrt{n}}.$$

Значение статистики позволяет рассчитать вероятность того, что параметр z не выйдет за предел допустимого значения α, определяемого выбранным критерием. Эта вероятность может быть представлена как

$$2\left[1 - \hat{\Phi}\left(\frac{|s_{obs}|}{\sqrt{n}}\right)\right] = erfc\left(\frac{|s_{obs}|}{\sqrt{n}}\right). \qquad (1)$$

Учитывая, что дополнительная функция ошибки определяется как

$$erfc(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-u^2} du , \qquad (2)$$

то, с учетом пределов интегрирования, выражение (1) можно переписать в виде:

$$2\left[1 - \Phi\left(\frac{|S_{obs}|}{\sqrt{2n}}\right)\right] = erfc\left(\frac{|S_{obs}|}{\sqrt{2n}}\right).$$

Если, в результате испытаний вычисленная величина $P = erfc\left(\frac{|S_{obs}|}{\sqrt{2n}}\right) \geq \alpha$, то тестируемая последовательность признается случайной.

Если величина α выбрана равной 0,01, то это значит, что из ста тестируемых последовательностей не более чем одна из них может быть забракована как «неслучайная».

Программная реализация такого теста сталкивается с двумя сложностями. Первая из них заключается в том, что достаточно сложно реализовать эффективный подсчет числа единичных n_1 и, соответственно, нулевых n_0 символов в тестируемой последовательности. Это объясняется тем, что в большинстве современных вычислительных архитектур команд для работы с отдельными битами нет.

Вторая сложность состоит в необходимости вычисления в каждом тесте значения дополнительной функции ошибки *erfc*. В принципе, ее значение можно вычислить и с помощью прикладного программного пакета MatLab, но, в процессе тестирования большого числа последовательностей обращаться к отдельному программному пакету не удобно.

Поскольку размер тестируемой последовательности n невелик (в [1] рекомендуется выбирать n немногим более ста символов), их количество в каждом байте может быть подсчитано следующим образом.

Учитывая, что при двоичном исчислении в вес каждого двоичного разряда машинном слове больше суммы весов всех разрядов, стоящих слева от него (младших по отношению к этому разряду), значение двоичных символов, входящих в состав этого слова и их количество может быть определено по следующей методике.

Будем считать, что символы a_i, k-разрядного слова $a = \{a_k, a_{k-1}, \ldots, a_1\}$, выражающего число b, пронумерованы справа налево (от младшего разряда к старшему). Тогда значение каждого из них можно рассчитать по правилу:

$$a_i = \begin{cases} 1 \text{ при } b - 2^{k-1} \geq 0, \\ 0 \text{ при } b - 2^{k-1} < 0. \end{cases}$$

Вычисление этой процедуры напрямую нецелесообразно, поскольку возведение в степень – операция, трудоемкая для вычислительной системы. Если, например, считывание тестируемой последовательности осуществляется побайтно, процедура подсчета количества единичных n_1 и нулевых символов n_0, написанная на языке Delphi, может иметь следующий вид:

```
b := a - 128;
if b >= 0 then Begin a := a - 128; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 64;
if b >= 0 then Begin a := a - 64; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 32;
```

```
if b >= 0 then Begin a := a - 32; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 16;
if b >= 0 then Begin a := a - 16; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 8;
if b >= 0 then Begin a := a - 8; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 4;
if b >= 0 then Begin a := a - 4; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 2;
if b >= 0 then Begin a := a - 2; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;

b := a - 1;
if b >= 0 then Begin a := a - 1; n1 := n1 + 1 End
          else Begin n0 := n0 + 1 End;
```

Здесь счетчики **n1** и **n0** обнуляются в начале тестирования и, затем, накапливают информацию о количестве соответствующих символов до окончания тестирования всей последовательности. Такая процедура выглядит несколько громоздкой, однако работает быстро. Она легко может быть расширена и на случай блока большего размера.

Что касается вычисления показателя P, представляющего собой дополнительную функцию ошибки $erfc(x)$ вида (2), то ее можно представить так:

$$erfc(x) = 1 - erf(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt .$$

Брать такой интеграл в указанных пределах неудобно, поэтому лучше вычислить функцию

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^{x} e^{-t^2} dt ,$$

а затем перейти к функции $erfc(x)$.

Функция $erf(x)$ не может быть представлена через элементарные функции. Однако ее можно представить в виде ряда:

$$erf(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{n!(2n+1)} = \frac{2}{\sqrt{\pi}} \left(x - \frac{x^3}{3} + \frac{x^5}{10} - \frac{x^7}{42} + \frac{x^9}{216} - \ldots \right)$$

Этим разложением можно пользоваться, если значение аргумента функции $erf(x)$ не превышает значения $x = 3$. При большем значении аргумента можно воспользоваться асимптотическим разложением вида:

$$erfc(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} \left[1 + \sum_{n=1}^{\infty} (-1)^n \frac{1 \times 3 \times 5 \times \ldots \times (2n-1)}{(2x^2)^n} \right] = \frac{e^{-x^2}}{x\sqrt{\pi}} \sum_{n=0}^{\infty} (-1)^n \frac{(2n)!}{n!(2x)^{2n}}$$

и вычислить значение функции $erfc(x)$ напрямую. Если ряд для определения значения функции $erf(x)$ требует вычисления до 30-ти членов, то последний ряд, для вычисления функции $erfc(x)$, дает хорошее приближение уже при вычислении четырех членов.

Представленные разложения для функций $erf(x)$ и $erfc(x)$ можно найти на сайте functions.wolfram.com по адресу http://functions.wolfram.com/GammaBetaErf/.

Таким образом, для тестирования последовательности, формируемой проектируемым генератором, необходим программный продукт, в который этот генератор будет входить в качестве составной части, и который позволит формировать m n-разрядных последовательностей. Здесь m – заданное число тестов. При чем этот продукт должен иметь отдельный встроенный генератор ключей, с тем, чтобы обеспечить их независимость. В этом же продукте можно разместить и программу для тестирования, которая, с использованием изложенных приемов, позволит определять долю последовательностей, не прошедших тест.

В ***заключение*** отметим, что именно такой подход поэтапного тестирования и подбора составных элементов генератора позволяет сделать предварительные выводы о приемлемости предполагаемой архитектуры проектируемого генератора или шифра. Авторами реализован описываемый подход к тестированию в виде двух отдельных составляющих – модуля генератора и модуля теста. Это позволяет применять модуль тестирования для иных типов генераторов той же разрядности.

СПИСОК ИСПОЛЬЗОВАННЫХ ПЕРВОИСТОЧНИКОВ

1. 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.
2. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness// http://www.stat.fsu.edu/pub/diehard/
3. Statistical test suite Crypt-X //http://www.isi.qut.edu.au/resources/cryptx/
4. Кнут Д. Искусство программирования для ЭВМ. – Т.2. – М.: Мир, 1977. – 727 с.
5. Харин Ю. С., Берник В. И., Матвеев Г. В., Агиевич С. В. Математические и компьютерные основы криптологии. Учебное пособие. – М.: Новое издание, 2003. – 272 с.
6. Земор Ж. Курс криптографии. – Ижевск: НИЦ «Регулярная и хаотическая динамика»; Институт компьютерных исследований, 2006. – 256 с.
7. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации: Учебное пособие. – 2005.
8. Фомичев В.М. Дискретная математика и криптология. Курс лекций / Под общ. ред. д-ра физ.-мат. н. Н.Д. Подуфалова. – М.: ДИАЛОГ-МИФИ, 2003. – 400 с.
9. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: Триумф, 2002. – 816 с.
10. Кац М. Статистическая независимость в теории вероятностей, анализе и теории чисел. – М.: Издательство иностранной литературы, 1963. – 156 с.