Розглянуті теоретичні принципи частотного тестування псевдовипадкових криптографічних послідовностей і способи їх практичної реалізації. Приведений новий метод поетапного тестування і підбору складових елементів генераторів псевдовипадкових послідовностей

Ключові слова: ПСП, криптографічна система, тестування

Рассмотрены теоретические принципы частотного тестирования псевдослучайных криптографических последовательностей и способы их практической реализации. Приведен новый метод поэтапного тестирования и подбора составных элементов генераторов псевдослучайных последовательностей

Ключевые слова: ПСП, криптографическая система, тестирование

Theoretical principles of the frequency testing of pseudocasual cryptographic sequences and methods of their practical realization are considered. The new method of the stage-by-stage testing and selection of component elements of generators of pseudocasual sequences is resulted

Keywords: PSP, cryptographic system, testing

УДК 621.391:519.72

ПОЭТАПНОЕ ТЕСТИРОВАНИЕ И ПОДБОР СОСТАВНЫХ ЭЛЕМЕНТОВ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Н.Ф. Казакова
Кандидат технических наук, декан
Факультет компьютерных наук и инновационных технологий
Международный гуманитарный университет
Фонтанская дорога, 33, г. Одесса, Украина, 65009
Контактный тел.: (048) 719-88-48, 703-64-18
E-mail: kaz2003@ukr.net

1. Введение

Постановка проблемы в общем виде и ее связь с научными и практическими задачами. Постоянное развитие криптографических систем, применяемых в современных информационно-телекоммуникационных системах, объясняет интерес специалистов, работающих в области криптографии, к разработке инструментальных средств их тестирования. Одним из критериев, на которых основывается оценка стойкости симметричных шифров, является равномерность распределения символов в гаммирующих последовательностях, а также в последовательностях, из которых формируются сеансовые ключи. С этой целью к настоящему моменту времени разработано большое количество тестирующих программных пакетов, позволяющих осуществлять комплексные испытания новых криптографических систем. К сожалению работы, опубликованные в этой области, разрознены и эта область криптоанализа, базирующаяся на фундаментальных положениях математической статистики, еще окончательно не сформировалась как самостоятельный раздел. Это создает серьезные проблемы и для специалистов, работающих в области криптографической защиты информации.

Анализ научной и технической литературы (например, [2...10]) показывает, что к настоящему времени разработано достаточно большое количество инструментальных средств, позволяющих осуществлять предварительный анализ пригодности последовательностей, порождаемых генераторами ПСП, для нужд криптографии. Эти пакеты прикладных программ реализуют наборы тестов, призванные дать ответ на вопрос, возможно ли, зная некоторый участок формируемой гаммы, предсказать последующий (или предыдущий) ее символ с вероятностью, отличной от 1/2? При отрицательном ответе формируемая гамма признается действительно случайной. Программное обеспечение и описание наборов этих тестов является общедоступным и может быть получено через Интернет-сайты своих разработчиков.

Проблема, однако, состоит в том, что использование предлагаемого программного обеспечения предполагает глубокий комплексный анализ готового продукта. Прилагаемые описания этих тестов содержат комплексные программы испытаний датчиков ПСП и процедуры вычисления обобщенного показателя качества, учитывающего результаты от всех тестов, входящих в состав пакета. Эти тесты позволяют выявлять различные виды аномалий в псевдослучайной

последовательности, которые в принципе могут быть использованы как уязвимости криптоалгоритма для организации на их основе атаки со стороны злоумышленников.

В то же время, инженерам-разработчикам необходимы инструментальные средства, реализующие простые и надежные процедуры тестирования на начальных и промежуточных этапах создания генераторов, позволяющих убедится в правильности выбранного пути. По этой причине разработчики алгоритмов формирования ПСП, как правило, зачастую предлагают собственные средства испытания, подтверждающие качество созданного программного продукта и далеко не всегда используют комплекты тестов, рекомендованные NIST [11] или другими авторитетными организациями. Отчасти это объясняется еще и тем, что разработчику важен не столько обобщенный показатель «случайности» сформированной датчиком гаммы, сколько вид отдельной аномалии в ее составе, приводящей к ухудшению этой самой «случайности». По этой причине в каждом конкретном случае приходится искать свой подходящий способ тестирования, и это обстоятельство объясняет повышенное внимание многих исследователей этой области к построению новых тестов. В подтверждение этого свидетельствует тот факт, что например, для шифра RC4, созданного в 1987 году, и в котором, в последствии, были обнаружены некоторые отклонения от случайности, никто не мог предложить практически реализуемого подходящего теста в течение многих лет, несмотря на большое число работ, посвященных этому вопросу. Все это говорит о том, поиск новых эффективных тестов, пригодных для использования в качестве инструментальных средств в процессе проектирования остается пока *актуальной и не решенной до конца задачей*.

Целью статьи является акцентирование внимания на теоретическиих принципах частотного тестирования псевдослучайных криптографических последовательностей и способах их практической реализации, а также рассмотрение нового метода поэтапного тестирования и подбора составных элементов генераторов псевдослучайных последовательностей.

2. Инструментальные средства тестирования

Современный симметричный шифр должен формировать шифрующую гамму при минимальных затратах вычислительных ресурсов системы, обеспечивая при этом заданную криптографическую стойкость. Эта стойкость, в свою очередь, определяется видом шифрующей последовательности. В общем виде, требования к таким последовательностям сводятся к следующему:

– отсутствие аналитической зависимости между последовательно сформированными числами;

– наблюдая предыдущие числа на некотором интервале, криптоаналитик не может предсказать следующие число с вероятностью, отличной от 1/2 (атака из прошлого);

– наблюдая последующие числа на некотором интервале, криптоаналитик не может предсказать следующие число с вероятностью, отличной от 1/2 (атака из будущего);

– все числа в формируемой последовательности равновероятны.

Что касается первого требования, то оно, в случае формирования последовательности с помощью вычислительного устройства, естественно, выполнено быть не может. Выполнение последних трех из перечисленных требований является обязательным условием для хорошей гаммирующей последовательности, и при их выполнении генерируемая последовательность будет неотличима от истинно случайной последовательности.

За последние десятилетия было разработано и исследовано большое количество «элементарных» генераторов псевдослучайных последовательностей (ПСП), к числу которых относят линейные конгруэнтные генераторы [1, 2], генераторы Фибоначчи с запаздыванием [1], генераторы, построенные на линейных регистрах с обратной связью [2, 3, 4, 5] и некоторые их разновидности. Общие результаты исследований перечисленных генераторов сводятся к тому, что все они не являются криптографически стойкими и, могут входить в состав формирователей ПСП, только в качестве составных элементов.

Как показано в [2], идея построения составного генератора базируется на том факте, что комбинация двух и более выходных последовательностей от генераторов разного типа с помощью линейных операций, таких как +, –, ×, ⊕ приводит генератору с «лучшими свойствами случайности». Наиболее удачные, с точки зрения криптографии, составные генераторы, подробно рассмотрены в работе Б. Шнаера [6].

Все чаще, в казавшихся ранее надежными алгоритмах, 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. В 1987 году Роном Ривестом был разработан потоковый шифр RC4 [6]. Он является наиболее удачным на сегодняшний день формирователем ПСП, однако в печати все чаще появляются сообщения о том, что и в нем уже найдены уязвимости.

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. По этой причине, шифры, предлагаемые на роль государственных стандартов, или уже принятые в качестве таковых принято считать стойкими, до тех пор, пока не станет известно об их эффективном взломе.

Разработчику остается довольствоваться только результатами предварительного тестирования. Об этом прямо сказано в руководстве к пакету тестов, разработанных NIST [7]. 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.

Наиболее рекомендуемым из известных тестовых пакетов, является 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.

Кроме того, известны и другие пакеты тестов, созданные для нужд криптографии. К их числу относится набор статистических тестов под названием Diehard [8], предназначенный для определения качества последовательности случайных чисел. Эти тесты были разработаны Джорджем Марсальей (George Marsaglia). Он включает 12 тестов и доступен 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 [9] и получить программное обеспечение и руководство по их применению.

Перечисленные пакеты предназначены для оценки уже готовых генераторов. В практической же работе такие устройства разрабатывают поэтапно, постепенно доводя их до уровня соответствия предъявляемым требованиям. Это обстоятельство является серьезным препятствием для применения на этапе создания архитектуры нового алгоритма.

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.

Эти 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 [7], если генератор не проходит частотный тест, то проведение всех прочих тестов уже не имеет смысла. Поэтому, учитывая вышесказанное, можно сделать вывод о том, что у разработчика должна быть методика, позволяющая быстро и эффективно оценивать «случайность» формируемой последовательности на различных этапах разработки генератора ПСП.

Частотное тестирование, основано на сравнении формируемой генератором последовательности с некоторым идеалом. Предполагается, что такой идеальный генератор формирует последовательность с равномерным распределением вероятностей единиц и нулей, такую, что следующий выходной бит невозможно предсказать по результатам наблюдения некоторого отрезка этой последовательности с вероятностью, отличающейся от 1/2.

В реальном случае 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 $0.01 < \alpha < 0.001$.

Предполагается, что при частотном тестировании 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

$$\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n,$$

где ε_i – двоичный символ, принимающий значения {0, 1}, будет подчинена биномиальному закону распределения вероятностей, который в соответствии с теоремой Муавра – Лапласа, при достаточно большом значении n, хорошо аппроксимируется стандартным нормальным законом $N_{0,1}$ с нулевым математическим ожиданием и единичной дисперсией. И, как показано в [7, 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 [7] 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:

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

Ее значение позволяет рассчитать вероятность того, что параметр z не выйдет за предел допустимого значения α, определяемого выбранным критерием. Эта вероятность может быть представлена в виде:

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

Учитывая, что дополнительная функция ошибки определяется как

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

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

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

Если, в результате испытаний вычисленная величина $P=\mathrm{erfc}\left(\frac{|S_{obs}|}{\sqrt{2n}}\right)\geq\alpha$, 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 [7] 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 разряду). 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 это трудоемкая для вычислительной системы операция. 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 n_1 и n_0 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 в виде:

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

Брать такой интеграл в указанных пределах неудобно, поэтому лучше вычислить функцию $\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^{x}e^{-t^2}dt$, а затем перейти к функции erfc (x).

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

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

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

можно воспользоваться асимптотическим разложением вида

$$\operatorname{erfc}(x)=\frac{e^{-x^2}}{x\sqrt{\pi}}\left[1+\sum_{n=1}^{\infty}(-1)^n\frac{1\times3\times5\times\ldots\times(2n-1)}{(2x^2)^n}\right]=$$

$$=\frac{e^{-x^2}}{x\sqrt{\pi}}\sum_{n=0}^{\infty}(-1)^n\frac{(2n)!}{n!(2x)^{2n}}$$

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 независимость тестируемых последовательностей. В этом же продукте можно разместить и программу для тестирования, которая, с использованием изложенных приемов, позволит определять долю последовательностей, не прошедших тест.

3. Заключение

Предлагаемый подход поэтапного тестирования и подбора составных элементов генератора позволяет сделать предварительные выводы о приемлемости предполагаемой архитектуры проектируемого генератора или шифра. Описываемый подход хорошо работает для генераторов ПСП с различной разрядностью машинного слова – от 8-ми до 32-х бит. При этом, его лучше реализовать в виде двух отдельных модулей для самого генератора и модуля теста. Это позволит применять модуль тестирования для иных типов генераторов той же разрядности.

Литература

1. Кнут, Д. Искусство программирования для ЭВМ. Т.2. [Текст] / Д. Кнут.– М. : Мир, 1977. – 727 с.
2. Харин, Ю. С. Математические и компьютерные основы криптологии [Текст] : учеб, пособие / Ю. С. Харин, В. И. Берник, Г. В. Матвеев, С. В. Агиевич.— М. : Новое издание, 2003. – 272 с.
3. Земор, Жиль. Курс криптографии [Текст] / Жиль Земор. М.– Ижевск : НИЦ «Регулярная и хаотическая динамика»; Институт компьютерных исследований, 2006. – 256 с.
4. Рябко, Б.Я. Криптографические методы защиты информации [Текст] : учеб, пособие / Б. Я. Рябко, А. Н. Фионов ; ТГГУТ.– Томск : ГПД, 2005.– 112 с.
5. Фомичев, В. М. Дискретная математика и криптология [Текст] : Курс лекций / В. М. Фомичев ; под общ. ред. д-ра физ.-мат. н. Н.Д. Подуфалова. – М. : ДИАЛОГ-МИФИ, 2003. – 400 с.
6. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си [Текст] / Б. Шнайер.– М. : Триумф, 2002. – 816 с.
7. 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.
8. The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness [Электронный ресурс].– Режим доступа : \www/ URL: http:// www.stat.fsu.edu/pub/diehard/ – 10.01.2010 г. – Загл. с экрана.
9. Statistical test suite Crypt-X [Электронный ресурс].– Режим доступа : \www/ URL: http://www.isi.qut.edu.au/resources/cryptx/ – 10.01.2010 г. – Загл. с экрана.
10. Кац, М. Статистическая независимость в теории вероятностей, анализе и теории чисел [Текст] / М. Кац.– М. : Издательство иностранной литературы, 1963.– 156 с.
11. 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.