Использование суперкомпьютеров для решения некоторых задач криптоанализа.

А.И. Миненко
магистрант СибГУТИ г. Новосибирск, Россия
minenkoai@mail.ru

Аннотация

Проведено исследование эффективности тестов “Стопка книг” и 15 тестов NIST для проверки генераторов случайных чисел. Протестированы 18 линейных конгруэнтных генераторов, RC4 и функция rand() в компиляторе C++ gcc 4.3.2 Linux. Показано, что “Стопка книг” может эффективнее находить отклонения от случайности, чем другие тесты.

1 Введение

Одной из задач криптоанализа является тестирование генераторов случайных чисел. Эта задача представляет большой интерес для разработчиков криптосистем. Во многих протоколах и шифрах случайные слова и числа используются как секретные ключи. Более того, с развитием криптографии выяснилось, что многие фундаментальные проблемы этой науки тесно связаны с генерированием и тестированием случайных чисел. Генераторы случайных чисел также находят широкое применение в вычислительных методах и при имитационном моделировании. Одно из требований, предъявляемое к современным генераторам: генерируемая случайная последовательность должна быть статистически неотличима от абсолютно случайной. Методы, используемые для проверки этого условия, рассматриваются в рамках математической статистики. Таким образом, необходимо проверить гипотезу H_0 о том, что источник порождает символы алфавита равновероятно и независимо, против альтернативной гипотезы H_1, являющейся отрицанием H_0.

Эта задача привлекает внимание многих исследователей в силу её теоретической и практической важности. Так, Национальный институт стандартов США (NIST) разработал пакет статистических тестов для проверки бинарных последовательностей, созданных либо аппаратными, либо программными генераторами случайных чисел. Кроме этих тестов, существуют и другие. Одним из них является статистический тест “Стопка книг”, который был предложен в [3,8].

Рассматриваемая задача имеет высокую трудоёмкость вычислений, так как каждым тестом нужно проверить несколько последовательностей, которые могут быть очень

большими. Каждая из последовательностей может быть проанализирована на отдельном вычислителе, поэтому для решения этой задачи хорошо подходит суперкомпьютер. Кроме того, распараллелить можно выполнение самих тестов.

2 Тесты

Тест “Стопка книг”, предложенный в [3,8], базируется на адаптивном коде с таким же названием. Основная идея метода в упорядочивании элементов как в стопке книг. Книга вынимается из стопки и кладётся наверх. Её номер становиться первым; книги, которые до этого были над ней сдвигаются вниз, а остальные остаются на месте. Дадим краткое описание этого теста.

Пусть некоторый источник порождает буквы из алфавита $A = \{a_1, a_2, \ldots, a_s\}$, $S > 1$, и требуется по выборке $x_1, x_2, ..., x_n$ проверить гипотезу $H_0 : p(a_1) = p(a_2) = ... = p(a_s) = 1/S$ против альтернативной гипотезы H_1 являющейся отрицанием H_0.

При тестировании по предлагаемому методу буквы алфавита A упорядочены (и занумерованы в соответствии с этим порядком от 1 до S), причём этот порядок меняется после анализа каждого выборочного значения x_i следующим образом: буква x_i, которую мы обозначаем через a, получает номер 1, номера тех букв, которые были меньше, чем номер a, увеличиваются на 1, а у остальных букв номера не меняются. Для более формального описания этого преобразования обозначим через $v^t(a)$ номер буквы $a \in A$ после анализа $x_1, x_2, ..., x_{t-1}$, и пусть начальный порядок $v^1()$ на буквах A задан произвольно. Тогда нумерация после анализа x_t определяется следующим образом:

$$v^{t+1}(a) = \begin{cases} 1, & \text{если } x_t = a, \\ v^t(a) + 1, & \text{если } v^t(a) < v^t(x_t), \\ v^t(a), & \text{если } v^t(a) > v^t(x_t). \end{cases} \quad (1)$$

(Как в стопке книг, если считать, что номер книги совпадает с положением в стопке. Книга извлекается и кладётся наверх. Её номер становится первым; книги, которые первоначально были над ней, сдвигаются вниз, а остальные остаются на месте.) Основная идея метода - подсчитывается не частота встречаемости букв в выборке $x_1, x_2, ..., x_n$, а частота встречаемости номеров букв (при описанном упорядочивании). В том случае, когда выполнена гипотеза H_1, вероятность (и частота встречаемости в выборке) некоторых букв больше $1/S$, и их номера в среднем будут меньше, чем у букв с меньшими вероятностями. (Другими словами, книги, к которым обращаются чаще, проводят в верхней части стопки значительно большее время, чем остальные. Следовательно, вероятность обнаружить требуемую книгу в верхней части стопки больше, чем в нижней.) Если же выполнена гипотеза H_0, то, очевидно, вероятность появления в выборке буквы с любым номером равна $1/S$.

При применении описываемого теста множество всех номеров $\{1, ..., S\}$ заранее, до анализа выборки, разбивается на $r > 1$ непересекающихся частей $A_1 = \{1, 2, ..., k_1\}$,

$A_2 = \{k_1 + 1, ..., k_2\}$, ..., $A_r = \{k_{r-1} + 1, ..., k_r\}$. Затем по выборке $x_1, x_2, ..., x_n$ подсчитывается количество номеров $v^t(x_t)$, принадлежащих подмножеству A_j, которое мы обозначим через n_j, $j = \overline{1,r}$. При выполнении гипотезы H_0 вероятность того, что $v^t(x_i)$ принадлежит множеству A_j, пропорциональна количеству его элементов, т.е. равна $|A_j|/S$. Затем по критерию χ^2 проверяется гипотеза H_0^*:

$$P\{v^t(x_t) \in A_j\} = \frac{|A_j|}{S} \tag{2}$$

против альтернативной гипотезы $H_1^* = \neg H_0^*$. Очевидно, при выполнении исходной гипотезы H_0 выполняется H_0^*, и наоборот, при выполнении гипотезы H_1^* выполняется H_1. Поэтому применение описанного критерия корректно.

При применении критерия χ^2 вычисляется величина

$$x^2 = \sum_{j=1}^{r} \frac{(n_j - nP_j^0)^2}{nP_j^0}, \tag{3}$$

где $P_j^0 = |A_j|/S$. Известно, что распределение случайной величины x^2 асимптотически приближается к распределению χ^2 с $r-1$ степенью свободы при выполнении H_0.

Необходимо отметить, что критерий применим, если объём выборки n удовлетворяет неравенству

$$nP_j^0 \geq 10, \quad j = \overline{1,r} \tag{4}$$

В состав пакета NIST входят 15 статистических тестов[7]. Перечислим их:

1) Frequency (Monobit)
2) Frequency within a Block
3) Runs
4) For the Longest Run of Ones in a Block
5) Binary Matrix Rank
6) Discrete Fourier Transform (Spectral)
7) Non-overlapping Template Matching
8) Overlapping Template Matching
9) Maurer's "Universal Statistical"
10) Linear Complexity
11) Serial
12) Approximate Entropy
13) Cumulative Sums (Cusum)
14) Random Excursions
15) Random Excursions Variant

Эти тесты основаны на различных особенностях, присущих только неслучайным последовательностям.

Во всех тестах, если вычисленное в ходе теста значение P-value < 0.01, то данная двоичная последовательность не является истинно случайной. В противном случае, последовательность носит случайный характер.

3 Генераторы

В этом исследовании протестированы 18 линейных конгруэнтных генераторов [2,5,6], RC4 [4] и функция rand() в компиляторе C++ gcc 4.3.2 Linux.

Все генераторы предназначены для создания равномерно распределённых целых чисел. Известно, что младшие знаки чисел, порождаемые линейными конгруэнтными генераторами, часто далеки от случайных [1]. Следуя этой рекомендации, из созданных генератором значений выделялись старшие 8 бит. Для генератора rand() также выделялись старшие 8 бит. RC4 выдавал по 8 бит.

Линейные конгруэнтные генераторы вычисляются по формуле

$$X_{n+1} = (aX_n + c)\ mod\ m, \tag{5}$$

где a, c, m, X_0 - параметры метода. Полное описание в [1]. Будем обозначать эти генераторы LCG(m, a, c). Параметр $X_0 = 1$ для всех генераторов. Приведём список линейных генераторов, которые протестированы:
1) LCG($2^{24}, 16598013, 12820163$), этот генератор используется в Microsoft VisualBasic 6.0.
2) LCG(2^{31}, 65539, 0), RANDU долгое время использовался во многих компьютерах в 1960-х - 1970-х годах.
3) LCG(2^{32}, 1099087573, 0), этот генератор предложил Fishman.
4) LCG(2^{32}, 69069, 1), этот генератор предложил Marsaglia.
5) LCG(2^{32}, 69069, 5) использовался в компиляторах GNU.
6) LCG(2^{32}, 1664525, 1013904223) предложен в Numerical Recipes.
7) LCG(2^{32}, 22695477, 1) используется в Borland C/C++.
8) LCG(2^{32}, 1103515245, 12345) используется в Digital Mars.
9) LCG(2^{32}, 134775813, 1) используется в Borland Delphi.
10) LCG(2^{32}, 214013, 2531011) используется в Microsoft Visual/Quick C/C++.
11) LCG(2^{46}, 5^{13}, 0) использовался для аэродинамического моделирования в НАСА в исследовательском центре Ames.
12) LCG(2^{48}, 25214903917, 11) это генератор drand48 из стандартной библиотеки Unix.
13) LCG(2^{48}, 5^{19}, 0) это традиционный генератор, использующийся в Национальной лаборатории США Los Alamos.
14) LCG(2^{48}, 33952834046453, 0) это один из генераторов, который предложил Fishman.
15) LCG(2^{48}, 44485709377909, 0) использовался в системе CRAY.
16) LCG(2^{59}, 13^{13}, 0) базовый генератор в математической библиотеке NAG, также он присутствует в векторной статистической библиотеке(VSL), которая находится в библиотеке математического ядра Intel.
17), 18) LCG(2^{63}, 5^{19}, 1), LCG(2^{63}, 9219741426499971445, 1) рекомендовано использовать на будущее в Национальной лаборатории США Los Alamos.

4 Описание экспериментов

Каждый генератор выдавал 100 последовательностей одинаковой длины. Если генератор создаёт действительно равномерно распределённые последовательности, то в сред-

нем 1 последовательность из 100 может быть забракована при уровне значимости $\alpha = 0.01$. Доверительный интервал можно вычислить с помощью критерия χ^2. Прямое вычисление показывает, что он равен от 0 до 4 включительно. Если количество забракованных последовательностей не попадает в этот интервал, то это говорит о том, что генератор выдаёт последовательности, при данной длине выборки, статистически отличимые от случайных. В этом случае будем говорить, что генератор не прошёл испытания, то есть забракован.

Тестирования проводились для последовательностей, выдаваемых генераторами, от 2^8 до 2^{23} бит. Для некоторых генераторов тестирование было проведено при больших длинах последовательности и только некоторыми тестами.

Стопка книг. Последовательность $x_n \in \{0, 1\}$ разбивалась на блоки длины l и при тестировании рассматривалась как выборка из алфавита размера $S = 2^l$. Множество всех позиций в "Стопке книг" разбивалось на два подмножества $A_1 = \{a_1, a_2, ..., a_k\}$, $A_2 = \{a_{k+1}, ..., a_S\}$. Второе подмножество не хранилось в памяти компьютера.

Frequency Test within a Block. M - длина каждого блока. Этот параметр выбирался из соотношения $n \geq MN$, $M \geq 20$, $M > 0.01n, N < 100$, где n - количество бит во входной последовательности.

Binary Matrix Rank Test. $M = Q = 32$.

Non-overlapping Template Matching Test. m - количество бит в каждом шаблоне. $m = 9$. B - шаблон. $B = 000000001$.

Overlapping Template Matching Test. m - количество бит в каждом шаблоне. $m = 9$. B - шаблон. $B = 111111111$. $M = 1032$.

Maurer's "Universal Statistical" Test. Выбирались рекомендованные значения в [11,12].

Linear Complexity Test. M = 500.

Serial Test. $m = \lfloor \log_2 n \rfloor - 3$, где n - количество бит во входной последовательности.

Approximate Entropy Test. $m = \lfloor \log_2 n \rfloor - 7$ до длины 2^{16} бит входной последовательности, где n - количество бит во входной последовательности. С 2^{16} m = 10.

Для остальных тестов параметры выбирались по умолчанию.

5 Результаты

В таблице 1 приведены результаты тестирования, перечисленных выше генераторов псевдослучайных чисел. В этой таблице представлены длины последовательностей в битах, выдаваемые генератором, с которых начинаются первые отклонения от случайности, определяемые данным тестом. Если отклонения обнаружены, то с увеличением длины входной последовательности отклонения возрастают.

Тестом "Стопка книг" найдены отклонения от случайности на 11 генераторах. Забракованы линейные генераторы с 1 по 10 номер в выше приведённом списке на различных длинах последовательности, начиная от 2^{13} и до 2^{23}. И RC4 при длине последовательности 2^{32}. Тестом Spectral на 4 генераторах. Забракованы линейные генераторы с 1 по 3 номер и номер 6. Тестом Serial на 3 генераторах. Забракованы линейные генераторы с 1 по 3

номер. Во всех случаях, если отклонения обнаружены, то с увеличением длины входной последовательности отклонения возрастают.

Из результатов видно, что тест “Стопка книг” забраковал большее количество генераторов. Во многих случаях это сделано при меньшей длине последовательности. А в некоторых случаях отклонения находит только “Стопка книг”.

Таблица 1: Тестирование генераторов

Генератор/Тест	Стопка книг	1. Frequency	2. Frequency Block	3. Runs	4. Longest Run	5. Binary Matrix	6. DFT	7. Non-overlapping	8. Overlapping	9. Maurer's "Universal"	10. Linear Complexity	11. Serial	12. Approximate Entropy	13. Cumulative Sums	14. Random Excursions	15. Random Exc Variant
LCG($2^{24}, 16598013, 12820163$)	2^{16}						2^{21}					2^{23}				
LCG($2^{31}, 65539, 0$)	2^{13}						2^{22}					2^{20}				
LCG($2^{32}, 1099087573, 0$)	2^{20}						2^{23}					2^{23}				
LCG($2^{32}, 69069, 1$)	2^{20}															
LCG($2^{32}, 69069, 5$)	2^{20}															
LCG($2^{32}, 1664525, 1013904223$)	2^{23}						2^{23}									
LCG($2^{32}, 22695477, 1$)	2^{20}															
LCG($2^{32}, 1103515245, 12345$)	2^{23}															
LCG($2^{32}, 134775813, 1$)	2^{20}															
LCG($2^{32}, 214013, 2531011$)	2^{19}															
LCG($2^{46}, 5^{13}, 0$)																
LCG($2^{48}, 25214903917, 11$)																
LCG($2^{48}, 5^{19}, 0$)																
LCG($2^{48}, 33952834046453, 0$)																
LCG($2^{48}, 44485709377909, 0$)																
LCG($2^{59}, 13^{13}, 0$)																
LCG($2^{63}, 5^{19}, 1$)																
LCG($2^{63}, 9219741426499971445, 1$)																
RC4	2^{32}															
rand (C++ gcc 4.3.2)																

6 Выводы

Проведённые исследования позволяют дать следующие выводы и рекомендации по применению рассмотренных тестов и генераторов.

Тест “Стопка книг” может эффективнее находить отклонения от случайности, чем другие тесты.

Линейные генераторы с параметром $m \leq 2^{32}$ не рекомендуется использовать в современных приложениях. RC4 рекомендуется использовать для создания последовательностей длиной до 2^{32} бит. Остальные генераторы можно использовать, так как до 2^{23} не было найдено отклонений выше перечисленными тестами.

Рекомендуемые параметры для “Стопки книг”: l - длину блока рекомендуется выбирать из ряда 8, 16, 20, 24, 32, 40 и так далее; размер одного или нескольких подмножеств $|A_i| = b * \sqrt{2^l}$, где b число из ряда 2, 4, 5, 8, 10, 16, 20, 32, 40, 64, 80, 128, 160 и так далее.

Литература

[1] Кнут Д. Искусство программирования на ЭВМ. Т. 2. Получисленные алгоритмы. М.: Мир, 1977.

[2] Линейный конгруэнтный метод. // Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс] URL: http://ru.wikipedia.org/wiki/Линейный_конгруэнтный_метод (дата обращения: 04.05.2010).

[3] Рябко Б. Я., Пестунов А. И. "Стопка книг" как новый статистический тест для случайных чисел. Проблемы передачи информации. Том 40. Вып. 1, 2004.

[4] Рябко Б. Я., Фионов А. Н. Криптографические методы защиты информации: Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2005. – 229 с.: ил.

[5] Karl Entacher A collection of classical pseudorandom number generators with linear structures - advanced version. June 16, 2000. [Электронный ресурс] URL: http://random.mat.sbg.ac.at/results/karl/server/server.html (дата обращения: 04.05.2010).

[6] L'Ecuyer P. and Simard R., TestU01: A C Library for Empirical Testing of Random Number Generators, ACM Transactions on Mathematical Software, 2007.

[7] Rukhin A. and others A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22 Revision 1a April 2010.

[8] Ryabko B. Ya., Monarev V. A. Using information theory approach to randomness testing. Journal of Statistical Planning and Inference, 2005, v. 133, n.1, pp. 95-110.