УДК

ОЦЕНКА СТОЙКОСТИ ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЛОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Касьян В.А., студент гр. 253501, Новицкий З.Я., студент гр. 253501

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Стройникова Е.Д. – старший преподаватель каф. информатики

Аннотация. Генераторы псевдослучайных чисел (PRNG) представляют собой алгоритмы, которые генерируют числа, кажущиеся случайными, но при этом являются определенными. Они являются неотъемлемой частью многих приложений, от криптографии до статистических имитаций. Различные операционные системы и языки программирования имеют свои собственные реализации PRNG. Данное исследование направлено на анализ и сравнение алгоритмов, преимуществ и производительности PRNG в обеих операционных системах, а также предлагает тесты для оценки их эффективности.

Ключевые слова. Linux, Windows, PRNG, HRNG, CSPRNG, LCG, SHA-1, CryptoAPI, НИСТ, NIST

Определения: случайная числовая последовательность (СЧП) — числовая последовательность, полученная каким-либо способом, в которой невозможно предсказать следующее число ввиду его неопределенности до генерации. Псевдо-СЧП (ПСЧП) — СЧП, которую можно предсказать. Генератор случайных чисел (RNG) — реальное или виртуальное устройство, генерирующее случайные числовые последовательности. Аппаратный генератор случайных чисел (HRNG) — RNG, использующий физические эффекты, которые современная физика не может предсказать, например квантовые явления, или например физически неклонируемые функции [1]. Генератор псевдослучайных чисел (PRNG) — виртуальный RNG, являющийся комбинаций математических функций для генерации (ПСЧП).

Рассмотрим различные PRNG и HRNG. Все программные генераторы являются PRNG, кроме случая использования HRNG. Приведем пример. В операционной системе (OS) GNU/Linux основным генератором псевдослучайных чисел (PRNG) является файл устройства /dev/urandom, который реализует криптографически стойкий генератор псевдослучайных чисел (CSPRNG). CSPRNG использует энтропию, собранную из различных источников, таких как ввод с клавиатуры и мыши, активность диска и аппаратные события. Собранная энтропия используется вместе с криптографической хэш-функцией (SHA-1) и линейным конгруэнтным генератором (LCG) для создания псевдослучайных чисел. Он разработан с расчетом прежде всего на быстродействие. Для большей надежности используется файл /dev/random, который всегда набирает гораздо больший пул энтропии. Также оба файла используют доступные HRNG, если таковые есть в системе.

В OS Windows функция CryptGenRandom, входящая в состав Microsoft Cryptographic Application Programming Interface (CryptoAPI), отвечает за генерацию псевдослучайных чисел. Она полагается на поточный шифр RC4 в качестве алгоритма генерации псевдослучайных чисел. Аналогично GNU/Linux, Windows собирает энтропию из различных аппаратных источников и объединяет ее с заданным пользователем значением семени. Значение сида затем используется алгоритмом RC4 для генерации псевдослучайных чисел.

Во многих языках программирования (ЯП) используются случайные числа, представляемые OS, однако в некоторых применяется собственная генерация. Так, в языке C используется достаточно старый алгоритм LCG без добавления каких-либо усовершенствований вроде хеш-функций или собирания энтропии. В ЯП C++, Python, Java, Ruby для генерации ПСЧП реализован алгоритм Мерсенна – Твистера. Это генератор линейной обратной связи сдвигового регистра (LFSR), что означает, что он использует регистр сдвига для хранения последовательности битов и функцию обратной связи для генерации новых битов на основе содержимого регистра. Алгоритм использует большой вектор состояния (обычно 624 или 19937 бит) для генерации последовательности 32-битных целых чисел.

Теперь рассмотрим устройство HRNG. Наиболее часто используемая технология в HRNG — физически неклонируемые функции (Physical Unclonable Functions, PUF) — это тип криптографических функций, которые используются для генерации уникальных ключей на основе

физических 1 характеристик электронных компонентов. PUF — это аппаратные системы, которые генерируют уникальный ключ на основе физических отличительных особенностей микрочипа, таких как индивидуальные различия в параметрах транзисторов, шумах в кристаллической решетке и т.д.

PUF функции генерируют уникальный ключ, который невозможно воспроизвести в другом микрочипе, что делает PUF ключи полезными для криптографических приложений, таких как аутентификация и шифрование.

Существует несколько типов PUF, таких как термальные PUF, оптические PUF, акустические PUF и т.д., каждый из которых использует различные методы для генерации уникальных ключей. PUF функции являются привлекательной альтернативой для традиционных методов генерации ключей, т.к. они генерируют ключи на основе физических характеристик, которые трудно воспроизвести, и не требуют хранения ключей в памяти устройства, что делает их более устойчивыми к атакам по сравнению с традиционными методами. Таким образом, СЧП, полученные с помощью PUF или квантовых эффектов невозможно воспроизвести еще раз, что делает их идеальным решением для создания HRNG. Однако основная проблема и отслеживания квантовых эффектов, и PUF — дороговизна оборудования и относительно низкая скорость работы, поэтому в современном мире преимущественно используются PRNG. Однако стоит оценить различные PRNG, ведь если они ненадежные, то их нельзя уверенно использовать для генерации ПСЧП. В данной работе изучим различные PRNG и дадим им оценку стойкости.

Прежде всего, как любые HRNG, PRNG должны максимально соответствовать следующим критериям: быть криптографически стойкими, иметь источники энтропии, быть производительными.

Оценка алгоритмов генерации псевдослучайных чисел выполняется с использованием следующих основных критериев:

● Равномерность: распределение созданных чисел должно быть равномерным по всем возможным значениям.

● Независимость: созданные числа не должны иметь никакого заметного шаблона или корреляции.

Равномерность относится к распределению сгенерированных чисел по всем возможным значениям. Хороший PRNG должен производить числа, которые следуют равномерному распределению, обеспечивая равную вероятность генерации каждого значения. Это свойство является важным для поддержания случайности и непредсказуемости сгенерированных чисел. В PRNG GNU/Linux и Windows используются хеш-функции для обеспечения генерации равномерно распределенных ПСЧП (РСПСЧП). ЯП, реализующие алгоритм Мерсенна – Твистера также генерируют РСПСЧП. Алгоритмы, работающие только на основе LCG, не достигают равномерного распределения ПСЧП.

Независимость относится к отсутствию видимых закономерностей или корреляций между последовательными числами в сгенерированной последовательности. Качественный генератор псевдослучайных чисел должен производить числа, которые независимы друг от друга, обеспечивая то, что информация о предыдущих числах не дает никакой информации о будущих числах в последовательности. Это свойство особенно важно для криптографических приложений, где предсказуемость сгенерированных чисел может подорвать безопасность системы. В PRNG GNU/Linux и Windows используются пул энтропии для обеспечения независимости сгенерированных случайных чисел. Алгоритм Мерсенна – Твистера является предсказуемым, однако для этого необходимо получить сид генерации, что является проблематичной задачей, учитывая диапазон генерации алгоритма: $Z^p - 1$, p – простое число, обычно 19937.

Для проверки производительности и качества PRNG-алгоритмов воспользуемся несколькими хорошо известными тестовыми наборами. Эти тесты оценивают различные свойства генерируемых псевдослучайных чисел, такие как равномерность, независимость и воспроизводимость. Следующие тестовые наборы широко признаны своей строгостью и эффективностью при оценке PRNG:

Оперативное тестирование Diehard. Оперативное тестирование Diehard является еще одним широко используемым набором тестов для оценки PRNG. Он состоит из 18 тестов:

Тест на расстояние между днями рождения (Birthday Spacings Test), тест на перекрывающиеся перестановки из 5 элементов (Overlapping 5-Permutations Test), двоичный тест на ранг матриц размера 31x31 (Binary Rank Test for 31x31 Matrices), двоичный тест на ранг матриц

размера 32x32 (Binary Rank 2 Test for 32x32 Matrices), двоичный тест на ранг матриц размера 6x8 (Binary Rank Test for 6x8 Matrices), тест на битовую последовательность (Bitstream Test), тест на попарное пересечение разреженности (OPSO — Overlapping-Pairs-Sparse-Occupancy Test), тест на пересечение четверок разреженности (OQSO — Overlapping-Quadruples-Sparse-Occupancy Test), тест ДНК (DNA Test), тест на подсчет единиц в определенных байтах (Count the 1s Test for specific bytes), тест на подсчет единиц во всех байтах (Count the 1s Test for all bytes), тест парковки (Parking Lot Test), тест минимального расстояния (Minimum Distance Test), тест на сферу в 3D (3D Sphere Test), тест сжатия (Squeeze Test), тест на перекрывающиеся суммы (Overlapping Sums Test), тест на серии (Runs Test), тест на кости (Craps Test) [2].

TestU01 — это более новый и продвинутый набор тестов для тестирования ГПСЧ, предлагающий строгую оценку качества их выходных данных. Разработанный Пьером Лекье и Ричардом Симардом, TestU01 включает несколько заранее определенных наборов тестов, таких как:

1. SmallCrush: базовый набор тестов для предварительной оценки качества PRNG.
2. Crush: более расширенный набор тестов, обеспечивающий полную оценку качества PRNG.
3. BigCrush: самый строгий набор тестов, созданный для тщательной оценки качества PRNG и предназначенный для выявления даже незначительных недостатков в сгенерированных числах.

Набор тестов Национального института стандартов и технологий США (NIST) является всесторонним набором статистических тестов, разработанных для оценки случайности и качества PRNG. NIST предлагает несколько тестов для проверки свойств конечной последовательности, и для этого вычисляется статистика, которая может быть как одним значением, так и множеством значений. Затем эта статистика сравнивается с эталонной статистикой, которая представляет собой математический вывод идеально случайной последовательности. Для вывода эталонной статистики используется множество теорем и научных исследований.

Набор включает 15 тестов, каждый из которых нацелен на различные свойства и характеристики генераторов случайных чисел. Тесты, включенные в набор NIST:

Частотный (побитовый) тест, частотный блочный тест, тест на одинаковые идущие подряд биты, тест на максимальную длину серии, тест на ранг, дискретное преобразование Фурье (спектральный тест), тест совпадения шаблона без перекрытия, тест совпадения шаблона с перекрытием, универсальный статистический тест, тест на приближенную энтропию, тест на случайные блуждания, вариант теста на случайные блуждания, серийный тест, тест на линейную сложность, тест на кумулятивную сумму [3], [4].

Также было принято решение реализовать и исследовать несколько тестов для проверки последовательности чисел на случайность.

В статистических исследованиях используется понятие нулевой гипотезы, которая предполагает отсутствие взаимосвязи между рассматриваемыми факторами. Например, в случае исследования связи курения и заболевания раком легких нулевая гипотеза предполагает, что курение не вызывает рак легких. Также есть альтернативная гипотеза, которая опровергает нулевую гипотезу и предполагает наличие взаимосвязи между факторами.

Для проверки гипотез используется собранная статистика, которая сравнивается с эталонной статистикой, полученной по математическим методам. При этом вводится погрешность, например в 5%. Если отклонение собранной статистики от эталонной превышает заданный уровень погрешности, делается вывод о том, что нулевая гипотеза не верна. Существуют 4 возможных варианта вывода: правильный вывод о случайности исследуемой последовательности, ошибки первого рода (когда последовательность признается не случайной, хотя она является случайной), ошибки второго рода (когда последовательность признается случайной, хотя она не является таковой) и отбраковка.

В статистике вероятность ошибки первого рода обозначается как α. Это вероятность отбросить “хорошую” случайную последовательность. Значение α определяется областью применения, например, в криптографии α обычно берут от 0.001 до 0.01.

При проведении тестов на случайность генерируется т.н. P-значение, которое означает вероятность того, что подопытный генератор произведет последовательность не хуже, чем

гипотетический “истинный”. Если P-значение равно 1, это означает, что наша последовательность идеально случайна, а если оно равно 0, то последовательность полностью предсказуема.

Для проведения тестов в данном случае берется α = 0.01. Если P-значение больше или равно 0.01, то последовательность признается случайной с уровнем доверия 99%. Если P-значение меньше 0.01, то последовательность отбраковывается с уровнем доверия 99%. Было сгенерировано 4 последовательности:

Алгоритмом Мерсенна – Твистера

00110100110101111010011000010010110111111100011110100001101100100110110010001101111
11100100110010010000011011000101110001111001100010000100001000010000100011001111100110101
01001010001000110001100110100000011010110110111101101100101001100110100111010010001
01111000110010001000001011100001101001011010100110111011001010001010011010010101110 1
11001001001011010010101111100 01111000001011001110001011010001111

PRNG OS GNU/Linux/dev/random

10100110001101110110100011000001100111110000111111100010101111011000110000111001011 0
11001111010110011100001001110111000010100110011100000100001000011000010011000000001111
01100101110011011010001010111011101000101001111001000101011000111101100001100011011 0
00001111001001000011100101100100101101111011110011010100010001011110101000001000011 0
10100111100010001111101011011 00000000000011110000110000101011110

PRNG OS GNU/Linux/dev/urandom

00101110100011110001011111100110001001111111101101001110101110101101011010111000010 1
10100011110011000000111000111000101101101100100111010010110011010000000011000110010 1
01011001111010010111001001010010010001011001010100111111100110110001010010100011000 0
11011101010001110001001101101010111110111100111100000100000111000111111011001101110 1
00101111101110101101001010011 0 10110010100000100011001101011010 10

Устаревший метод в ЯП С

01010011000101100110011001101001100001101110010001001001111011011010100010000100101
10011010111101011011010011111010001111000111000110010111000001001111001110010000101 1
00000000011000001101010000101101101001000101011101101111101011111011110000101000010 1
01010111100001010011001011010000001010100010110011110011010000100001110111010011100 1
00001011001001011010111110000 00100101011010100010101010100000011

Частотный (побитовый) тест.

$S_n = \sum_{i=1}^{n} X_i$, где $X_i = 2x_i - 1$, где x_i – *i*-ый бит.

Статистика:

$S_{obs} = \frac{|S_n|}{\sqrt{n}}$, где *n* – количество битов.

P-значение:

$P_{value} = erfc(\frac{S_{obs}}{\sqrt{2}})$, где $erfc(x) = \frac{2}{\sqrt{n}}\int_x^{\infty} e^{-t^2}dt$ – дополнительная функция ошибок.

Рекомендуется тестировать последовательности длиной не менее 100 бит.

Таблица 1. Последовательность 400 битов частотным (побитовым) методом (Генерация: алгоритмом Мерсенна – Твистера, PRNG OS GNU/Linux /dev/random, PRNG OS GNU/Linux /dev/urandom, устаревшим методом в ЯП С)

Генерация	Sn	Sobs	P-значение
Алгоритм Мерсенна-Твистера	-8	0.4	0.69
PRNG OS GNU/Linux /dev/random	-16	0.8	0.42
PRNG OS GNU/Linux /dev/urandom	14	0.7	0.48
Устаревший метод в ЯП С	-22	1.1	0.27

Частотный блочный тест.

Этот тест делается на основе предыдущего, только теперь значения пропорции «1»/«0» для каждого блока анализируются методом Хи-квадрат. Ясно, что это соотношение должно быть приблизительно равным 1. Для проведения теста требуется разбить последовательность битов на N блоков по M битов, неполный блок отбрасывается.

$X_{obs}^2 = 4 \cdot M \cdot \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2$, ГДЕ $\pi_i = \frac{K}{M}$, ГДЕ K – количество бит, равных единице в данном блоке битов.

Р-значение: $P_{value} = Q\left(\frac{N}{2}, \frac{x_{obs}^2}{2}\right)$, где $Q(a,x) = \frac{1}{\Gamma(a)} \int_x^{\infty} e^{-t} t^{a-1} dt$, – неполная верхняя гамма-функция, $\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$ – стандартная гамма-функция.

Для достаточно высокой точности примем $n = 400$, $M = 40 \Rightarrow N = \frac{n}{M} = 10$.

Таблица 2. Последовательность 400 битов частотным блочным методом (Генерация: алгоритмом Мерсенна – Твистера, PRNG OS GNU/Linux /dev/random, PRNG OS GNU/Linux /dev/urandom, устаревшим методом в ЯП С)

Генерация	X_{obs}^2	Р-значение
Алгоритм Мерсенна-Твистера	9.2	0.51
PRNG OS GNU/Linux /dev/random	7.2	0.71
PRNG OS GNU/Linux /dev/urandom	7.3	0.70
Устаревший метод в ЯП С	14.1	0.17

Как видно из таблицы 2, во всех 10-ти тестах Р-значение было больше 0.01, значит, все последовательности битов прошли тест.

Тест на одинаковые идущие подряд биты.

Этот тест проверяет последовательности одинаковых битов в заданной последовательности и определяет, соответствуют ли количество и размеры этих последовательностей ожидаемым для случайной последовательности. Если смена 0 на 1 или наоборот происходит слишком редко, то такая последовательность не является случайной.

$\pi = \frac{\sum_{i=1}^{n} X_i}{n}$ – доля единиц в последовательности битов.

Далее проверяется условие:

$\left|\pi - \frac{1}{2}\right| < \frac{2}{\sqrt{n}}$.

Если оно не удовлетворяется, то весь тест считается неуспешным и на этом он заканчивается.

В противном случае вычисляем суммарное число знакоперемен V_n:

$V_n = \sum_{k=1}^{n-1} r(k) + 1$, где $r(k) = 0$, если $X_i = X_{i+1}$, или $r(k) = 1$ в противном случае.

Для вычисляем Р-значение:

$P_{value} = erfc\left(\frac{|V_n - 2n\pi(1-\pi)|}{2\sqrt{2n\pi}(1-\pi)}\right)$.

Таблица 3. Последовательность 400 битов на одинаковые идущие подряд биты (Генерация: алгоритмом Мерсенна – Твистера, PRNG OS GNU/Linux /dev/random, PRNG OS GNU/Linux /dev/urandom, устаревшим методом в ЯП С)

Генерация	π	$\left\|\pi - \frac{1}{2}\right\| < \frac{2}{\sqrt{n}}$.	V	P-значение
Алгоритм Мерсенна-Твистера	0.54	+	212	0.19
PRNG OS GNU/Linux /dev/random	0.48	+	188	0.24
PRNG OS GNU/Linux /dev/urandom	0.52	+	213	0.18
Устаревший метод в ЯП С	0.47	+	214	0.14

Эти наборы тестов могут применяться для проверки любых PRNG с целью оценки их производительности и сравнения качества их выходных данных. Анализируя результаты этих тестов, пользователи могут получить представление о сильных и слабых сторонах каждого PRNG и принимать обоснованные решения относительно их пригодности для конкретных приложений. Кроме того, регулярное тестирование PRNG с использованием этих наборов тестов помогает гарантировать, что любые обновления или модификации базовых алгоритмов не нарушают качество сгенерированных псевдослучайных чисел.

В целом, оценка стойкости генераторов случайных чисел является важной задачей, которая требует специальных знаний и опыта. Можно сделать вывод, что современные генераторы случайных числовых последовательностей проходят актуальные тесты на случайность, что означает пригодность их использования для генерации случайных чисел и числовых последовательностей для различных целей, например для криптографии. Лишь устаревшие PRNG проходят такие тесты, но и то не всегда.

Список использованных источников:

1. Хабр [Электронный ресурс]. — Режим доступа: URL: https://habr.com/ru/articles/343386/.

2. Duke. Department of physics[Электронный ресурс]. — Режим доступа: URL:https://webhome.phy.duke.edu/~rgb/General/dieharder.php.

3. Хабр [Электронный ресурс]. — Режим доступа: URL: https://habr.com/ru/companies/securitycode/articles/237695/.

4. NIST [Электронный ресурс]. — Режим доступа: URL: https://csrc.nist.gov/publications/detail/sp/800—22/rev—1a/final.

UDC

RANDOM NUMBER SEQUENCE GENERATORS SECURITY ESTIMATION

Kasyan V.A., Novickiy Z.Y.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Stroinikova E.D. – senior lecturer, Department of Informatics

Annotation. Pseudo-random number generators (PRNGs) are algorithms that generate numbers that appear to be random but are certain. They are an integral part of many applications, from cryptography to statistical simulations. Various operating systems and programming languages have their own PRNG implementations. This study aims to analyze and compare the algorithms, benefits, and performance of PRNGs on both operating systems, and proposes tests to evaluate their ef ectiveness.sequences

Keywords. Linux, Windows, PRNG, HRNG, CSPRNG, LCG, SHA-1, CryptoAPI, NIST