МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский технологический университет»

МИРЭА

Институт кибернетики

Кафедра информационной безопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине

«Криптографические протоколы»

На тему:

«Реализация ГПСЧ»

Подготовил

студент группы ККСО–01–14 А.С. Першин

Руководитель работы

А.П. Никитин

Москва, 2019

Оглавление

1. Основные понятия о ГПСЧ

Для определения понятия ГПСЧ введем некоторые понятия:

Случайное число – число, представляющее собой реализацию случайной величины.

Детерминированный алгоритм – алгоритм, который возвращает те же выходные значения при тех же входных значениях.

Псевдослучайное число – число, полученное детерминированным алгоритмом, используемое в качестве случайного числа.

Физическое случайное число (истинно случайное) – случайное число, полученное на основе некоторого физического явления.

Как правило, генерация случайного числа состоит из двух этапов:

1) генерация нормализованного случайного числа (то есть равномерно распределенного от 0 до 1);
2) преобразование нормализованных случайных чисел r_i в случайные числа x_i, которые распределены по заданному закону распределения или в необходимом интервале.

Генератор случайных бит (ГСБ) — это устройство или алгоритм, который выдает последовательность статистически независимых и несмещенных бит (то есть подчиняющихся закону распределения).

Генератор случайных бит может быть использован для генерации равномерно распределенных случайных чисел. Например, случайное целое число в интервале $[0;n]$ может быть получено из сгенерированной последовательности случайных бит длины $\lfloor \lg n \rfloor + 1$ путем конвертации её в соответствующую систему исчисления.

Если полученное в результате целое число превосходит n, то его можно отбросить и сгенерировать еще одну последовательность бит. Поэтому далее мы будем использовать термин генератор случайных чисел наравне с термином генератор случайных бит.

Генератором псевдослучайных бит (детерминированным ГПСБ) – будем называть детерминированный алгоритм (функция), который получает на вход двоичную последовательность длины k и выдает на выходе двоичную последовательность длины $l \gg k$ (l значительно больше k), которая «выглядит случайной»[1]. Входное значение ГПСБ называется начальным вектором (также

[1] Поясним понятие «выглядит случайной». Понятно, что последовательность, сгенерированная детерминированным алгоритмом, не является случайной. Однако цель алгоритма в том, чтобы взять некоторую маленькую последовательность истинно случайных чисел и использовать её для генерации длинной последовательности, не отличимой от истинно случайной последовательности чисел той же длины. Убедиться в том, что последовательность

называют инициализационным вектором и обозначают *IV*), а выход называется псевдослучайной последовательностью бит.

Говорят, что ГПСБ проходит все полиномиальные по времени вероятностные тесты на статистическую случайность, если не существует полиномиального по времени[2] вероятностного алгоритма, который бы мог корректно отличить выходную последовательность генератора от истинно случайной последовательности той же длины с вероятностью превышающей $\frac{1}{2}$.

Говорят, что ГПСБ успешно проходит тест на следующий бит, если не существует полиномиального по времени алгоритма, который может по входным l битам последовательности s предсказать (l + 1)-й бит s с вероятностью превышающей $\frac{1}{2}$.

2. Виды ГПСЧ

Генераторы случайных чисел по способу получения чисел делятся на:

- аппаратные;
- табличные;
- алгоритмические.

Аппаратные генераторы (истинно) случайных последовательностей должны обладать источником энтропии[3]. Разработка генераторов, использующих источники энтропии, генерирующих некоррелированные и статистически независимые числа – достаточно сложная задача. Кроме того, для большинства криптографических приложений такой ГПСЧ не должен быть предметом изучения и воздействий стороны противника.

Табличные генераторы в качестве источника случайных чисел используют заранее подготовленные таблицы, содержащие проверенные некоррелированные числа и не являются генераторами в строгом понимании этого понятия. Недостатки такого способа очевидны: использование внешнего ресурса для хранения чисел, ограниченность последовательности, предопределенность значений. В качестве примера табличного метода можно привести книгу.

Алгоритмический генератор является комбинацией физического генератора и детерминированного алгоритма. Такой генератор использует

чисел случайна (или не случайна) можно либо при помощи статистических тестов, выявляющих специфические особенности случайных последовательностей, либо аналитико-вычислительными методами.

[2] То есть время выполнения теста ограничено сверху значением полинома, вычисленного от длины l выходной последовательности. Полиномиальным алгоритмом или алгоритмом полиномиальной временной сложности называется алгоритм, у которого временная сложность равна O(p(n)), где p(n) - некоторая полиномиальная функция, а n - входная длина.

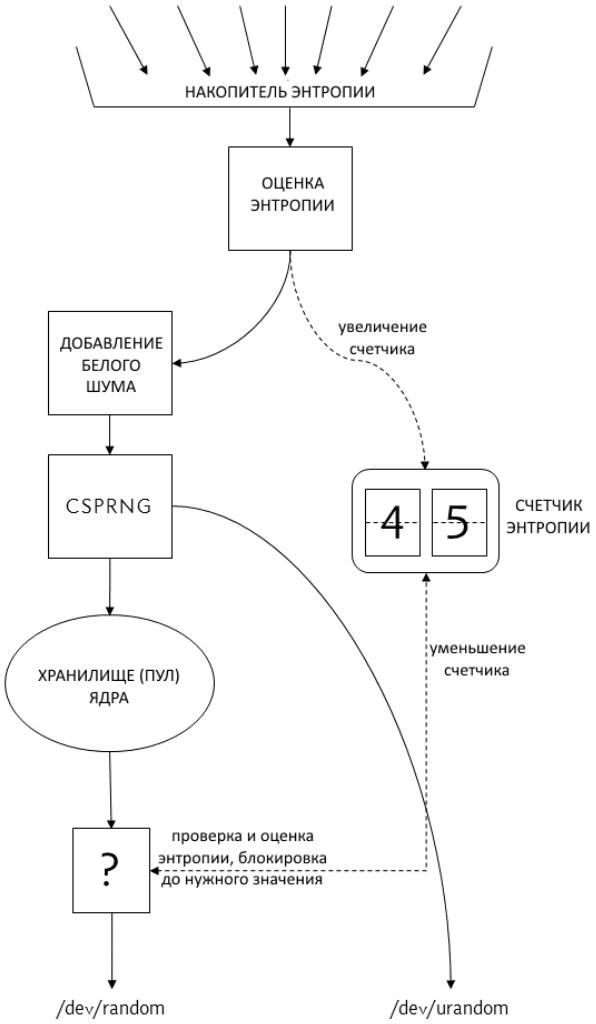
[3] Источники энтропии используются для накопления энтропии, с последующим получением из неё начального значения. Под энтропией понимают меру, определяющую «неопределенность», то есть то, насколько полученная из системы информация говорит о «неизвестности» работы самой системы выработки последовательности.

ограниченный набор данных, полученный с выхода физического генератора для создания длинной последовательности чисел преобразованиями исходных чисел. Данный вид генераторов представляет наибольший интерес в силу его очевидных преимуществ над генераторами случайных чисел других видов.

3. Пример реализации ГПСЧ

В качестве реализуемого ГПСЧ возьмем структурную схему ГПСЧ из библиотек /dev/random и /dev/urandom ядра операционной системы Linux.

Структура Linux's ГСЧ/ГПСЧ



Работа схемы заключается в следующем. Существует три накопителя энтропии:

- первичный;
- для /dev/random;
- для /dev/urandom.

Последние два накопителя получают данные из первичного. У каждого накопителя присутствует свой счетчик энтропии, однако для /dev/random и /dev/urandom они близки к 0 (нулю). Для увеличения их энтропии при запросе

пользователя они используют в качестве источника энтропии первичный накопитель.

Первичный накопитель энтропии собирает её из различных источников (физических/аппаратных датчиков [USB контроллер подключаемых временных устройств, датчики температуры, встроенные часы, положение указателя мыши, время нажатия клавиш на клавиатуре и другие]). Вычисляется энтропия этой накопленной информации и немедленно это значение добавляется к значению счетчика энтропии. Если энтропия принятой информации имеет малое значение, то происходит коррекция до того момента, пока значение энтропии будет в пределах нормы, установленной в параметрах генератора.

После исправления недостатков происходит привнесение белого шума (равномерно распределенных битов).

CSPRNG представляет собой стойкий криптографический ГПСЧ, то есть ГПСЧ с определенными свойствами, позволяющими использовать его в криптографии. Одна из возможных реализаций CSPRNG основывается на использовании криптографических алгоритмов.

Примером такой реализации может выступать безопасный блочный шифр, который преобразуется в режиме счетчика (Counter mode [CTR] или Output Feed Back mode [OFB]) в ГПСЧ (работает по принципу поточного шифра). Таким образом, выбрав случайный ключ, можно получать следующий случайный блок. Очевидно, что периодом такого генератора будет не больше, чем 2^n для n-битного блочного шифра. Также очевидно, что безопасность такой схемы полностью зависит от секретного ключа.

В роли CSPRNG может выступать и криптографически стойкая хеш-функция. В таком случае исходное значение счетчика должно оставаться в секрете.

Поточные шифры работают на основе генерации псевдослучайного потока бит, которые некоторым образом комбинируются (с помощью операции XOR) с битами открытого текста. Запуск такого шифра на входной последовательности даст новую псевдослучайную последовательность, возможно, даже с более длинным периодом. Такой метод безопасен, только если в самом поточном шифре используется надежный криптографически стойкий ГПСЧ. При этом, начальное состояние счетчика должно оставаться секретным.

После прохождения CSPRNG информация попадает в хранилище (пул) ядра, откуда /dev/urandom берет псевдослучайные числа, получая их из пула напрямую, если у счетчика энтропии имеется запрашиваемое количество чисел (бит). Для /dev/random происходит оценка энтропии полученной информации и только после принятия решения результат поступает на /dev/random к пользователю.

Существует большое множество криптографически стойких блочных шифров. Один из них – AES256 с размером блока 128 бит, который можно использовать в режиме OFB, чтобы получить хорошую ПСП.

Проверим получаемую ПСП, реализуемую данным алгоритмом в режиме гаммирования с обратной связью, на статистические тесты, входящие в пакет статистических тестов Dieharder, предлагаемые NIST в документе NIST SP 800-22.

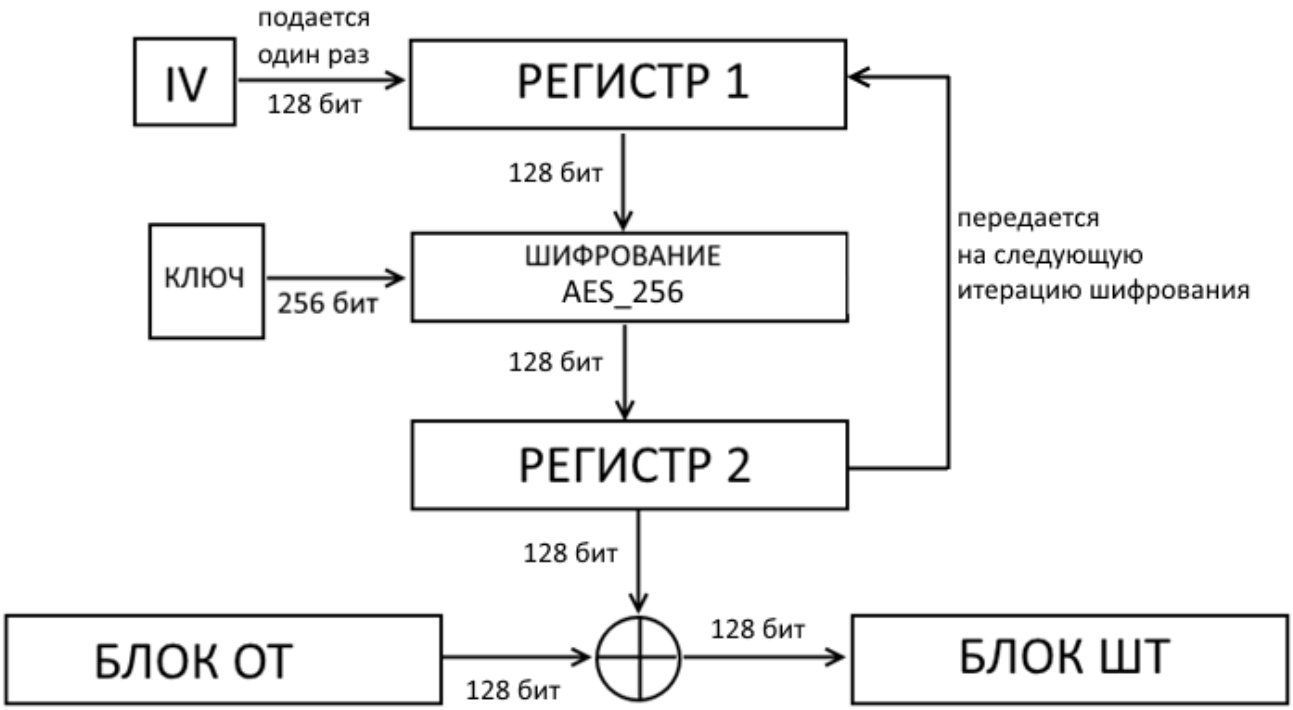
Статистические тесты NIST – пакет статистических тестов, разработанный Лабораторией информационных технологий, являющейся главной исследовательской организацией Национального института стандартов и технологий (NIST). В его состав входят 15 статистических тестов, целью которых является определение меры случайности двоичных последовательностей, порожденных либо аппаратными, либо программными ГСЧ.

В пакет тестов входят:

- Частотный побитовый тест;
- Частотный блочный тест;
- Тест на последовательность одинаковых битов;
- Тест на самую длинную последовательность единиц в блоке;
- Тест рангов бинарных матриц;
- Спектральный тест;
- Тест на совпадение неперекрывающихся шаблонов;
- Тест на совпадение перекрывающихся шаблонов;
- Универсальный статистический тест Маурера;
- Тест на линейную сложность;
- Тест на периодичность;
- Тест приблизительной энтропии;
- Тест кумулятивных сумм;
- Тест на произвольные отклонения;
- Другой тест на произвольные отклонения.

4. ГПСЧ на основе AES256_OFB

Рассмотрим работу блочного шифра AES256 в режиме OFB (гаммирования с обратной связью) в виде структурной схемы:



Режим гаммирования с обратной связью работает следующим образом. Содержимое РЕГИСТР 1 сначала получает вектор инициализации (IV), затем перед каждым шифрованием получает содержимое из РЕГИСТРА 2 (результат работы алгоритма AES256).

Открытый текст не шифруют напрямую: вначале шифруется вектор инициализации (IV), а уже полученный в результате шифртекст ксорится (XOR) с блоком открытого текста. Затем шифруется результат работы алгоритма AES256 на предыдущем шаге и ксорится (XOR) со следующим блоком открытого текста и так далее.

Таким образом. работа в режиме OFB заключается в следующем:

ВХОД:

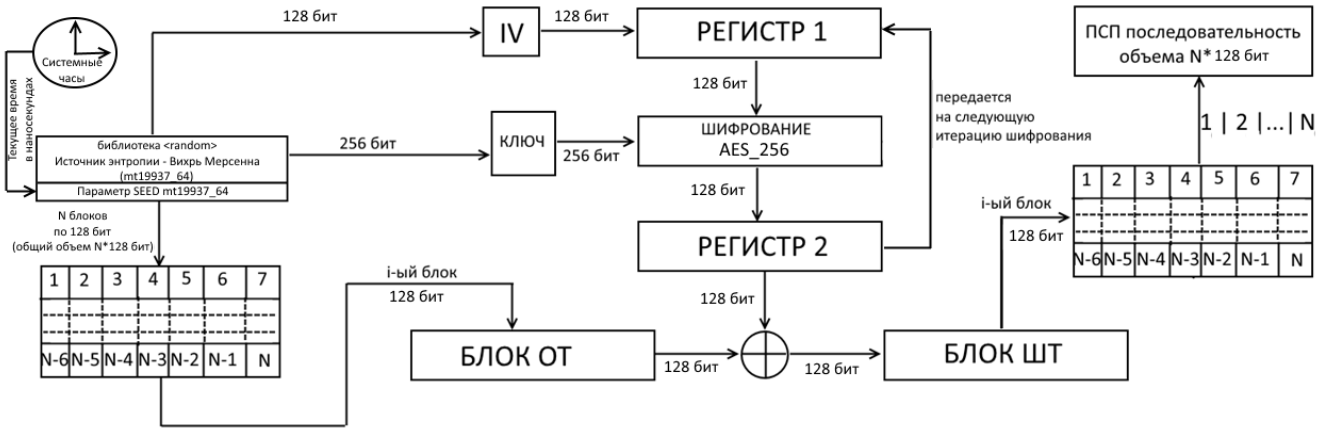
- вектор инициализации IV (128 бит);
- ключ (256 бит);
- блоки открытого текста [ОТ] (128 бит).

ВЫХОД:

- блоки шифртекста [ШТ] (128 бит).

Конкатенируя блоки ШТ на выходе алгоритма мы получаем ПСП, которую можно проверить на тесты NIST SP 800-22.

Структурная схема предложенного ГПСЧ будет выглядеть следующим образом:



Для моделирования источника накопления энтропии будем использовать библиотеку языка C++11 <random>, в котором реализован криптографически нестойкий ГПСЧ Вихрь Мерсенна (mt19937_64), который для генерации ПСП принимает на вход значение SEED (семени). Семя, как вариант, можно получать из текущего значения системных часов (в наносекундах). Вихрь Мерсенна будет вырабатывать в нашем эксперименте:

- вектор инициализации IV (объемом 128 бит = 16 байт);
- энтропия [которая на схеме отмечается как ОТ] (объемом 83886080 бит = 10485760 байт = 655360 блоков размером 128 бит);
- ключ (объемом 256 бит = 32 байта).

Далее был применен алгоритм выработки ПСП через алгоритм AES256 в режиме OFB (гаммирования с обратной связью).

5. Результаты реализации ГПСЧ AES256_OFB

Разработка производилась в IDE Microsoft Visual Studio 15 Pro. Для реализации задания лабораторной работы было создано общее решение с именем CryptoProtocols. Реализация алгоритма ГПСЧ входит в проект CSPRNG решения CryptoProtocols.

Для тестирования корректности разрабатываемых проектов в решении CryptoProtocols был создан отдельный проект GoogleTestingSolutionProject модульного тестирования gtest (для unit testing) и gmock (для проверки корректности вызовов методов). Данные пакеты устанавливались через менеджер пакетов NuGet для Visual Studio.

Результат выполнения тест кейсов для проверки корректности работы функции выработки ПСП и времени выполнения для подсчета производительности работы (т.к. gtest замеряет работу вызовов кейсов в микросекундах, то для повышения точности была использована библиотека <chrono> c++11 с точностью до микросекунд) приведены на Рис. 1.

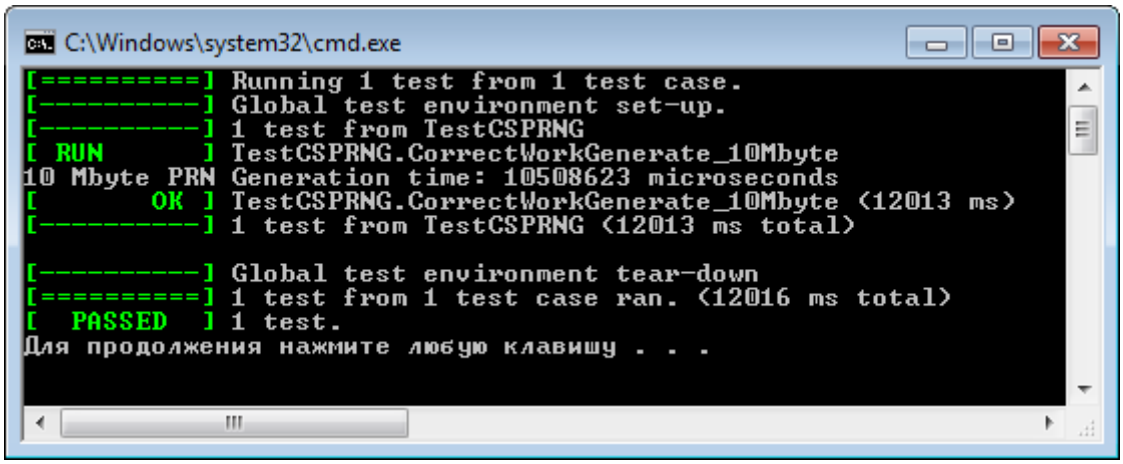


Рис. 1. Результат тестирования реализованного алгоритма CSPRNG

Запускался тест на ЦП AMD A6-3410MX (4 ядра, 4 потока) на Рис. 2. По полученным данным посчитаем время выработки ПСП для данного ЦП. Данные приведены в Табл. 1.

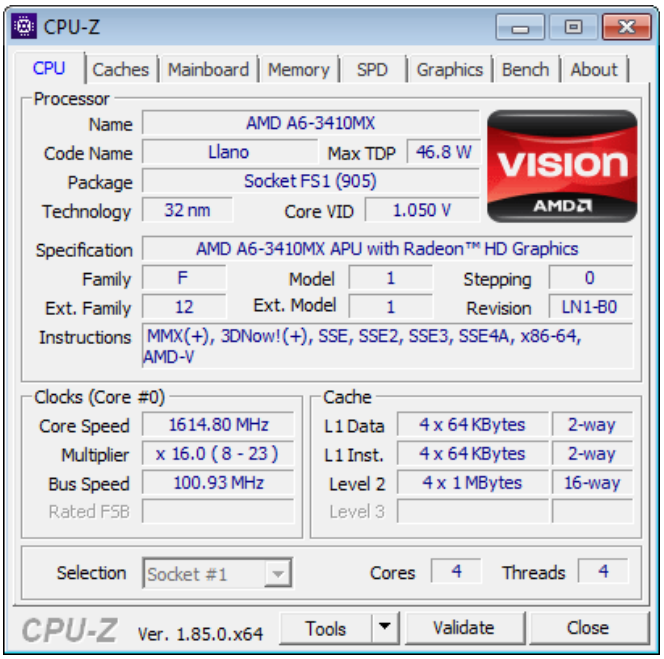


Рис. 2. ЦП AMD A6-3410MX (4 ядра, 4 потока)

Табл. 1. Скорость выполнения выработки ПСП алгоритма CSPRNG

Алгоритм	Размер данных [Мбайт]	Скорость [Мбайт/с]
AES256_OFB	10	0,95159946265

Проверим полученную последовательность на батарею тестов NIST SP800-22 из пакета Dieharder.

Запуск проверки сгенерированной последовательности осуществляется командой:

```
dieharder -g 201 -f testrands.txt -a
```

Где:

- -g 201 (формат тестируемых данных – полученный на выходе ГПСЧ файл ASCII с ПСП);
- -f (указывает путь к файлу testrands.txt);
- -a (выполнить проверку по всем тестам, которые есть в сборке библиотеки, посмотреть конкретные тесты можно флагом -l, запуск через флаг -d [номер теста])

Батарея тестов NIST SP800-22 в пакете Dieharder имеет номер -d 102

```
dieharder -g 201 -f testrands.txt -d 102
```

Результат проверки последовательности длиной 83886080 бит = 10 Мбайт приведен на Рис. 3.

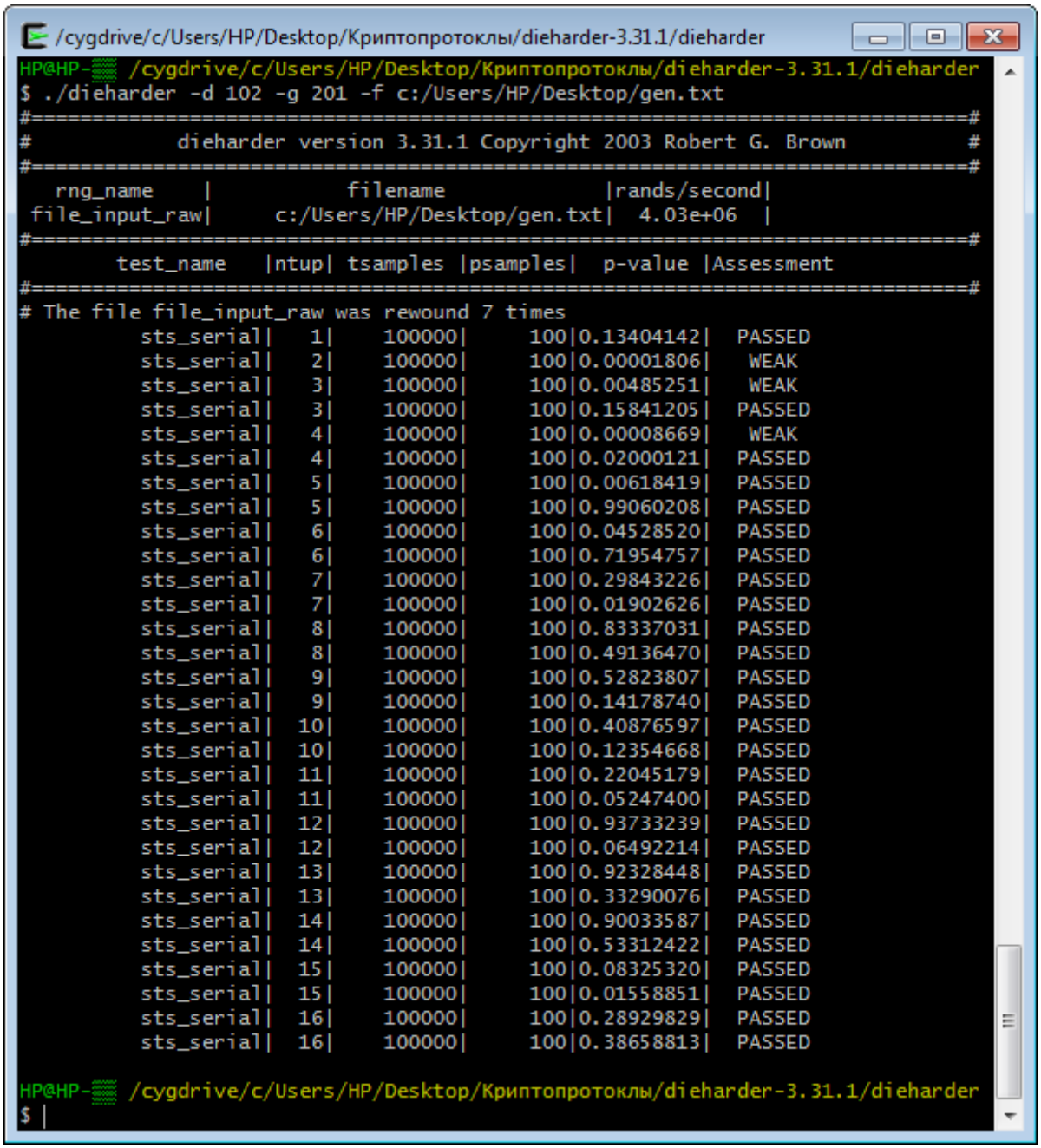


Рис. 3. Результат проверки последовательности длиной 83886080 бит = 10 Мбайт

Таким образом, последовательность, выработанная реализованным CSPRNG, прошла проверку на случайность, значит данный ГПСЧ пригоден для дальнейшего использования.

Литература

1. Зязин В.П. «Курс лекций ПСП», РТУ (МИРЭА), 2018 - 2019 г.
2. Режимы шифрования блочных шифров. [Интернет ресурс], ссылка https://ru-wiki.ru/wiki/Режим_шифрования.
3. Meths about /dev/urandom. [Интернет ресурс], ссылка https://www.2uo.de/myths-about-urandom/#structure.

4. FIPS PUB 197 «ADVANCED ENCRYPTION STANDARD (AES)» [Интернет ресурс], ссылка https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf