

пользователей  $t$ .

Покажем корректность предложенного алгоритма КЭЦП. Подставив подпись  $(r, s)$  в проверочное уравнение (14) убеждаемся, что выполняется равенство  $R' = R$ :

$$\begin{aligned} R' &= sP + rQ = \left( \sum_{i=1}^t s_i \right) P + r \sum_{i=1}^t Q_i = \sum_{i=1}^t (s_i P + r Q_i) = \\ &= \sum_{i=1}^t ((k_i + d_i r) P + r(-d_i P)) = \sum_{i=1}^t (k_i P) = \sum_{i=1}^t R_i = R. \end{aligned}$$

Поскольку  $R' = R$ , то также выполняется и  $r' = r$ .

Таким образом, корректность предложенного алгоритма КЭЦП на основе стандарта ДСТУ 4145-2002 доказана.

#### IV Заключение

Применение понятия коллективного открытого ключа, вычисляемого как свертка подмножества индивидуальных открытых ключей, позволяет построить протоколы КЭЦП, перспективные для практического применения в технологиях электронного документооборота, благодаря обеспечению одновременности формирования подписи и ее целостности. Для произвольной совокупности пользователей может быть легко выработан соответствующий им коллективный ключ, на основе которого можно проверить их коллективную подпись. Достоинством предложенных протоколов является возможность их практической реализации на основе стандартной инфраструктуры открытых ключей и стандартов электронной цифровой подписи ГОСТ 34.310-95 и ДСТУ 4145-2002. Учитывая аналогию стандартов ГОСТ 34.310-95 и ГОСТ Р 34.10-2001, легко показать, что рассматриваемый подход формирования КЭЦП может быть реализован также и на основе последнего стандарта.

Использование КЭЦП представляет удачное решение известной проблемы одновременного подписания контракта [10]. Представляет интерес использование КЭЦП и для построения протоколов «множественной подписи» [10], что составляет самостоятельную задачу дальнейшего развития протоколов на основе понятия коллективного открытого ключа.

*Литература:* 1. Венбо Мао. Современная криптография. Теория и практика. - М., СПб., Киев: Издательский дом «Вильямс», 2005. - 763 с. 2. Молдовян Н. А. Введение в криптосистемы с открытым ключом. - Санкт-Петербург: БХВ-Петербург, 2005. - 286 с. 3. Min-Shiang Hawng, Cheng-Chi Le: Research issues and challenges for multiple digital signature, Int. J. of Network Security, 2005. Vol. 1. No 1, pp. 1-7. 4. Карякин Ю. Д. Технология «AXIS-2000» защиты материальных объектов от подделки // «Управление защитой информации». - М.: 1997. - Т.1. - № 2. - С. 90-97. 5. Молдовян Н. А., Молдовян П. А. Новые протоколы слепой подписи // «Безопасность информационных технологий». - М.: 2007. - № 3. - С. 17-21. 6. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. Межгосударственный стандарт ГОСТ 34.310-95. - 16 с. 7. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. Держстандарт України ДСТУ 4145-2002. - 94 с. 8. Беспалов А. В., Телиженко А. Б. Криптосистемы на эллиптических кривых: Учеб. пособие. - К.: ИВЦ "Видавництво «Політехніка»", 2004. - 224 с. 9. Информационная технология. Криптографическая защита информации. Функция хэширования. Межгосударственный стандарт ГОСТ 310. - 12 с. 10. B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code (Second Edition) - New York.: John Wiley & Sons. - 1996. - 758 p.

УДК 681.3.06

## ТЕСТИРОВАНИЕ СЛУЧАЙНЫХ И ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ С ИСПОЛЬЗОВАНИЕМ КОНТЕКСТНОГО МОДЕЛИРОВАНИЯ

**Виталий Шарапов**

Физико-технический институт Национального технического университета Украины «КПИ»

**Анотація:** Пропонується новий алгоритм тестування та оцінки якості випадкових та псевдовипадкових двійкових послідовностей, що ґрунтується на контекстному моделюванні.

**Summary:** In the work the new algorithm of testing and an estimation of quality of the random and pseudo-random binary sequences, based on context modeling are offered.

**Ключевые слова:** Контекстное моделирование, контекстное сжатие, случайные и псевдослучайные последовательности, тестирование случайных последовательностей.

## I Тестирование случайных и псевдослучайных последовательностей

Среди множества различных тестов проверки качества случайных и псевдослучайных последовательностей, особую группу составляют тесты, основанные на сжатии информации [1, 2]. Ввиду того, что существующие тесты, построенные по такому принципу, хорошо себя зарекомендовали и в них используется лишь отдельные методы сжатия, в статье предлагается опробовать другие методы сжатия для построения тестов анализирующих свойства случайности последовательностей.

На сегодняшний день одними из самых эффективных алгоритмов сжатия информации являются методы контекстного моделирования [2]. В данной работе метод контекстного сжатия взят за основу для построения способа тестирования двоичных последовательностей на случайность.

## II Прогнозируемость символов случайной последовательности

Согласно [3] двоичная последовательность  $s, s \in \{0,1\}^n, n \in \mathbb{N}$  будет непредсказуемой, если для произвольного полиномиального вероятностного алгоритма  $A$  выполняется следующее условие. Пусть  $l\zeta$  – некоторый префикс последовательности  $s$  (то есть первые  $k-1$  бит), где  $\zeta \in \{0,1\}, l \in \{0,1\}^{k-1}$ , а  $k$  – это случайное натуральное число, не превышающее  $n$ . Тогда

$$\left| P[A(l) = \zeta] - \frac{1}{2} \right| < \frac{1}{n^c}, \quad (1)$$

для любого  $c$  при достаточно больших  $n$ .

Запишем условие (1) менее формально. Последовательность не является случайной, если анализируя эту последовательность слева направо и накапливая при этом информацию мы получаем возможность прогнозировать последующие символы. Последовательность является случайной, если она не идентифицирована неслучайной с вероятностью превышающей  $\frac{1}{2}$ .

Пусть задан некоторый полиномиальный вероятностный алгоритм  $A$ , и имеется двоичная строка  $s, s \in \{0,1\}^n, n \in \mathbb{N}$ , такая, что  $s = l\zeta, l \in \{0,1\}^{n-1}, \zeta \in \{0,1\}$ . Будем называть прогнозом  $n$ -го символа  $A(l)$  и говорить, что:

- символ  $\zeta$  – предсказан, если  $A(l) = \zeta$ ;
- символ  $\zeta$  – непредсказан, если  $A(l) \neq \zeta$ ;
- символ  $\zeta$  – непредсказуем, если  $A(l)$  не определено.

При построения критерия были использованы следующие определения и положения:

- любой символ, находящийся в строке, непосредственно следует за некоторой подстрокой; возможно пустой. Такая подстрока называется *контекстом* [2] символа;
- *порядок контекста* – это длина подстроки.
- имеется библиотека контекстов – это объект который, хранит в себе контексты и позволяет выполнять над ними операции добавления и поиска; также в библиотеке контекстов кроме самих контекстов хранится следующая сопутствующая информация:
  - сколько раз встретился именно этот контекст;
  - сколько раз за данным контекстом встречался нулевой бит;
  - разница между количеством правильных и неправильных прогнозов, сделанных по данному контексту.

Библиотека контекстов должна обеспечивать чтение и модификацию этой информации.

- Критерий реализуется с помощью алгоритма, результатом работы которого является количество предсказанных, непредсказанных и непредсказуемых символов.

### III Неформальный алгоритм подсчета и классификации прогнозируемых символов

Для начала рассмотрим неформальное описание алгоритма тестирования.

На вход алгоритму подается:

- битовая строка  $s$  длины  $n$ ;
- порядок максимально рассматриваемого контекста –  $M$ ;

Результатом работы алгоритма являются:

- количество предсказанных символов –  $l_1$ ;
- количество непредсказанных символов –  $l_2$ ;
- количество непредсказуемых символов –  $l_3$ ;

В работе алгоритма используются следующий вспомогательный объект – библиотека контекстов,  $B$ .

Суть алгоритма состоит в следующем.

0. Обнуляем все результирующие счетчики.

*Входная строка посимвольно просматривается слева направо и выполняются следующие действия.*

1. Последовательно для каждого бита строки  $s$ , начиная с первого и заканчивая последним, просматриваются контексты различных порядков, начиная с максимально возможного порядка для данного бита (для бита в позиции с номером, большим порядка  $M$  – это порядок контекста  $M$ , а для остальных – это «номер позиции бита» минус 1) и заканчивая нулевым порядком.
  - 1.1. Если рассматриваемый в данный момент контекст уже присутствует в библиотеке  $B$ , то анализируем информацию о нем:
    - если «разность количества правильных и неправильных прогнозов» больше 2 и по контексту можно сделать прогноз (нулевой и единичный биты встречался за данным контекстом разное число раз), то делаем прогноз (если текущий бит в строке совпадает с битом, который чаще следовал за данным контекстом, то увеличиваем счетчик предсказанных, в противном случае увеличиваем счетчик непредсказанных символов) и прерываем анализ по текущему контексту (переходим к пункту 3).
  2. Мы просмотрели все контексты и ни по одному из них не смогли сделать прогноз, значит текущий бит в строке непредсказуем – увеличиваем счетчик непредсказуемых символов.
  3. Снова последовательно для каждого бита строки  $s$ , начиная с первого и заканчивая последним, просматриваются контексты, начиная с максимально возможного порядка для данного бита (для символов в позиции с порядком, большим порядка  $M$  – это порядок  $M$ , а для остальных – это «номер позиции бита» минус 1) и заканчивая нулевым порядком.
    - 3.1. Если рассматриваемый в данный момент контекст уже присутствует в библиотеке  $B$ , то модифицируем информацию о нем:
      - если нулевой бит встречался за данным контекстом чаще единичного и текущий бит в строке нулевой, то уменьшаем счетчик разности количества правильных и неправильных прогнозов на 1;
      - если нулевой бит встречался за данным контекстом реже единичного и текущий бит в строке единичный, то увеличиваем счетчик разности количества правильных и неправильных прогнозов на 1;
      - в остальных случаях счетчик разности количества правильных и неправильных прогнозов уменьшаем на 1;
      - увеличиваем счетчик, показывающий сколько раз данный контекст вообще встречался;
      - если текущий символ в строке нулевой, то увеличиваем счетчик количества следования нулевых битов на 1.
    - 3.2. Если рассматриваемый в данный момент контекст не находится в библиотеке  $B$ , то заносим его в библиотеку  $B$  и устанавливаем информацию о нем следующим образом:
      - сколько раз встретился именно этот контекст  $\leftarrow 0$ ;
      - сколько раз за данным контекстом встречался нулевой бит  $\leftarrow 0$ , если текущий бит в строке единичный и  $\leftarrow 1$  в противном случае;
      - разница данному контексту между количеством правильных и неправильных прогнозов  $\leftarrow 1$ .

#### IV Формальный алгоритм подсчета и классификации прогнозируемых символов

Введем несколько условных обозначений:

- $\Sigma(a, b)$  – подстрока строки  $\Sigma$ , начиная с символа в позиции  $\alpha$  и заканчивая символом в позиции  $\beta$ ;
- $x \in B$  – присутствует ли контекст  $x$  в библиотеке  $B$ ;
- $r \leftarrow B(x)$  – занести в  $r$  информацию о контексте  $x$  из  $B$ .

Запишем алгоритм III в формальном виде.

Входные данные:

- Битовая строка  $s$  длины  $n$ ;
- Порядок максимально рассматриваемого контекста,  $M$ .

Результатом работы алгоритма являются:

- Количество предсказанных символов,  $l_1$ ;
- Количество непредсказанных символов,  $l_2$ ;
- Количество непредсказуемых символов,  $l_3$ ;

В работе алгоритма используются следующие вспомогательные объекты и переменные:

- Библиотека контекстов,  $B$ ;
- $i, j, m$  – целые;
- $x$  – строка (временный контекст);
- $r$  – информация из библиотеки о контексте ( $r.S$  – количество нулевых битов,  $r.U$  – общее количество прогнозов,  $r.R$  – количество правильных прогнозов);

1. Подготовка:  $l_1 \leftarrow 0, l_2 \leftarrow 0, l_3 \leftarrow 0$ .

2. Цикл по строке:  $i \leftarrow i + 1$ .

3. Если  $i > n$  то завершить работу.

4. Цикл обновления счетчиков: Если  $m < M$ , то  $j \leftarrow i$  иначе  $m \leftarrow M, j \leftarrow 1, l_3 \leftarrow l_3 + 1$ .

5.  $x \leftarrow S(i - j, i - 1), j \leftarrow j + 1$ .

6.  $x \in B$ ? Если нет, то перейти на 11. Иначе  $r \leftarrow B(x)$ .

7.  $2 \times r.R - 1 \geq r.U$ ? Нет – перейти на 11.

8.  $l_3 \leftarrow l_3 - 1$ .

9. Если  $(2 \times r.S < r.U \wedge S(i) = "0") \vee (2 \times r.S \geq r.U \wedge S(i) = "0")$ , то  $l_1 \leftarrow l_1 + 1$  иначе  $l_2 \leftarrow l_2 + 1$ .

10.  $j \leftarrow j + 1$ . Если  $j \geq M$ , то перейти на 4.

11. Цикл обновления сопутствующей информации: Если  $m < M$ , то  $j \leftarrow i$  иначе  $m \leftarrow M, j \leftarrow 1$ .

12.  $x \leftarrow S(i - j, i - 1), j \leftarrow j + 1$ .

13.  $x \in B$ ? Если да, то перейти к 18.

14.  $r \leftarrow B(x)$ .

15.  $r.U \leftarrow 1, r.S \leftarrow 0, r.R \leftarrow 0$ .

16. Если  $S(i) = "0"$ , то  $r.S \leftarrow r.S + 1$ .

17. Перейти к 21.

18.  $r.U \leftarrow r.U + 1$ .

19. Если  $(2 \times r.S < r.U \wedge S(i) = "0") \vee (2 \times r.S \geq r.U \wedge S(i) = "0")$ , то  $r.U \leftarrow r.U + 1$ , иначе  $r.U \leftarrow r.U - 1$ .

20. Если  $S(i) = "0"$ , то  $r.S \leftarrow r.S + 1$ .

21.  $j \leftarrow j + 1$ .

22. Если  $j \geq M$ , то перейти к 11.

23.  $i \leftarrow i + 1$ .

24. Если  $i < n$ , то перейти к 2.

## V Статистика

Результаты работы алгоритма IV являются исчерпывающими по информативности, но не удобными для сопоставления нескольких различных последовательностей по мере случайности или для проверки проходит последовательность тест или нет. Формально работу вышеприведенного алгоритма можно представить в виде функции  $A$ :

$$A: S^n \rightarrow M^n, \quad (2)$$

где  $S^n = \{0,1\}^n$ ,  $M^n = \bigcup_{\substack{n_1, n_2, n_3 \in \mathbb{Z}^+ \\ n_1 + n_2 + n_3 = n}} (n_1, n_2, n_3)$

Для таких сравнений гораздо более удобной будет одна единственная скалярная величина. Такую величину, применительно к исходным строкам, можно получать с помощью функции следующего вида:

$$f: S^n \rightarrow [0,1]. \quad (3)$$

Условимся, что если имеются две различные последовательности, из которых одна является «более случайной» чем другая, то «более случайная» последовательность должна отображаться в большее число.

То есть, для реализации критерия в виде функции  $f(x)$ , необходимо использовать дополнительную функцию  $g$  такую, что

$$f(x) = g(A(x)). \quad (4)$$

где  $g: M^n \rightarrow [0,1]$ ,  $x \in S^n$ ,  $f(x) \in [0,1]$ .

Предлагается использовать следующий вариант реализации функции  $g$

$$g(c_1, c_2, c_3) = \begin{cases} 1, & c_1 = 0 \text{ и } c_2 = 0 \\ \frac{1}{2}, & c_1 c_2 = 0 \text{ и } c_1 + c_2 \neq 0 \\ \left( \arctg\left(2 \frac{c_3}{c_1 + c_2}\right) + \arctg\left|\frac{c_1}{c_2} - \frac{c_2}{c_1}\right| \right), & \text{иначе.} \end{cases} \quad (5)$$

Введем функцию  $U$ ,  $U: S^n \rightarrow [0,1]$  следующим образом:

$$U(x) = g(A(x)), \quad (6)$$

где  $g(c_1, c_2, c_3)$  функция (5).

Применение этой функции к строке следует понимать, как применение алгоритма к строке, а затем сведение многомерных результатов этого алгоритма с помощью функции  $g$  к скалярной величине.

## VI Экспериментальные данные

Для проверки пригодности алгоритма было выполнено много различных экспериментов над псевдослучайными двоичными последовательностями, полученными с помощью различных генераторов. В данном разделе приводятся наиболее интересные из них.

В экспериментах каждая используемая строка длины  $n$  генерировалась одним из следующих генераторов.

- Модулярный генератор.
  1. Берется  $\lceil \frac{n}{32} \rceil + 1$  значений модулярного генератора из стандартной библиотеки Delphi 2007 и конкатенируются в одну строку в своем непосредственном двоичном представлении (одно значение генератора это 32 бита).
  2. Полученная на этапе 1 строка усекается до длины  $n$ .
- Генератор AES.
  1. Берется  $\lceil \frac{n}{32} \rceil + 1$  значений модулярного генератора из стандартной библиотеки Delphi 2007 и конкатенируются в одну строку в своем непосредственном двоичном представлении;

2. Генерируется этим же самым генератором четыре значения, которые затем конкатенируются в один ключ.
  3. Строка, созданная на этапе 1, шифруется ключом, сгенерированным на этапе 2 алгоритмом AES в режиме обратной связи по гамме.
  4. Полученная на этапе 3 строка усекается до длины  $n$ .
- Генератор AES /число.
    1. Генерируем строку длины  $n$  с помощью генератора AES.
    2. В сгенерированной строке заменяем биты в позициях кратных «числу» (параметр из названия генератора) на значение предыдущего бита в строке.
  - Генератор DJVU.
    1. Данные берутся из файла с типом «djvu» (Формат djvu используется для хранения сжатых книг. На данный момент это формат с самой лучшей степенью сжатия среди форматов, использующихся для хранения отсканированных книг). Если из одного файла оказывается недостаточно для выборки нужного размера, то берется еще один или несколько различных файлов с таким же типом. Каждый использованный файл имел размеры в диапазоне 512–1024 кб.

Над разработанным тестом были выполнено следующее исследование – бралась серия различных псевдослучайных последовательностей одинаковой длины и от них вычислялась статистика  $U$  (6). По полученным значениям строилась гистограмма.

Ниже приводится ряд графиков (рисунки 1 – 3). На каждом отдельном графике приводятся данные по четырем различным генераторам для одной и той же длины. Вот параметры, с которыми были построены эти графики:

- максимальный порядок контекста – 8;
- размер выборки – 1000;
- количество интервалов на гистограмме – 50.

На рисунке 1 рассматриваются результаты исследования последовательностей с длиной 128 бит. Из графика видно, что статистика  $U$  от генераторов AES и модулярного очень напоминает нормальное распределение с параметрами: среднее – 0,66 и среднеквадратичное отклонение 0,14. От генератора AES /5: статистика  $U$  тоже достаточно похожа на нормальное распределение, но со средним 0,54 и среднеквадратичным отклонением 0,17. Про генератор DJVU, также можно сказать, что статистика от него ведет себя как нормально распределенная величина, но с оговоркой о дополнительных «шумах» в левой части. Эти «шумы» можно пояснить тем, что формат DJVU не совсем однородный и содержит в себе кроме достаточно случайных данных еще и служебные, которые никак нельзя назвать случайными.

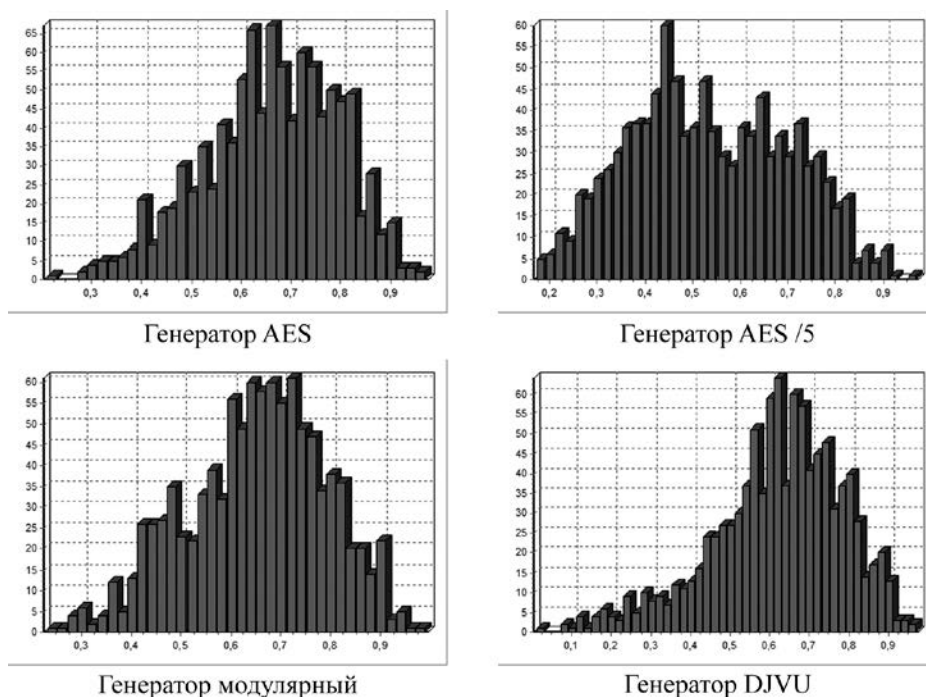


Рисунок 1 – Длина 128 бит

На рис. 2 рассматриваются результаты экспериментов от последовательностей длиной 1024 бита. В отличие от предыдущего случая вместо генератора AES /5 использовался генератор AES /9. Это связано с тем, что если в хорошей последовательности с длиной порядка тысячи бит испортить каждый 5-й бит, то она станет очень плохой, а для эксперимента нужна была не очень плохая, но и не хорошая последовательность. Схожесть поведения статистики как нормально распределенной величины сохранилась для всех четырех генераторов. Изменились только параметры:

- для AES среднее 0,62, среднеквадратичное отклонение 0,1;
- для модулярного среднее 0,60, среднеквадратичное отклонение 0,11;
- для AES /9 среднее 0,48, среднеквадратичное отклонение 0,09;
- для DJVU среднее 0,61, среднеквадратичное отклонение 0,11.

Принципиальным отличием от предыдущего графика является, то, что теперь модулярный и AES генераторы отличаются по среднему значению. Имеющееся различие между ними не позволяет их отличать с помощью статистики  $U$  с достаточной достоверностью. Но уже заметно нарастание ухудшения качества модулярного генератора. Подробный анализ модулярного генератора дан в [4].

На рис. 3 приводятся результаты экспериментов над последовательностями с длиной 16384 бита. Статистики, приведенные на этом рисунке (кроме генератора AES /17, использованного вместо AES /9 с предыдущего эксперимента, по тем же причинам, что и AES /9 был введен на смену AES /5) уже ближе к логнормально распределенной случайной величине.

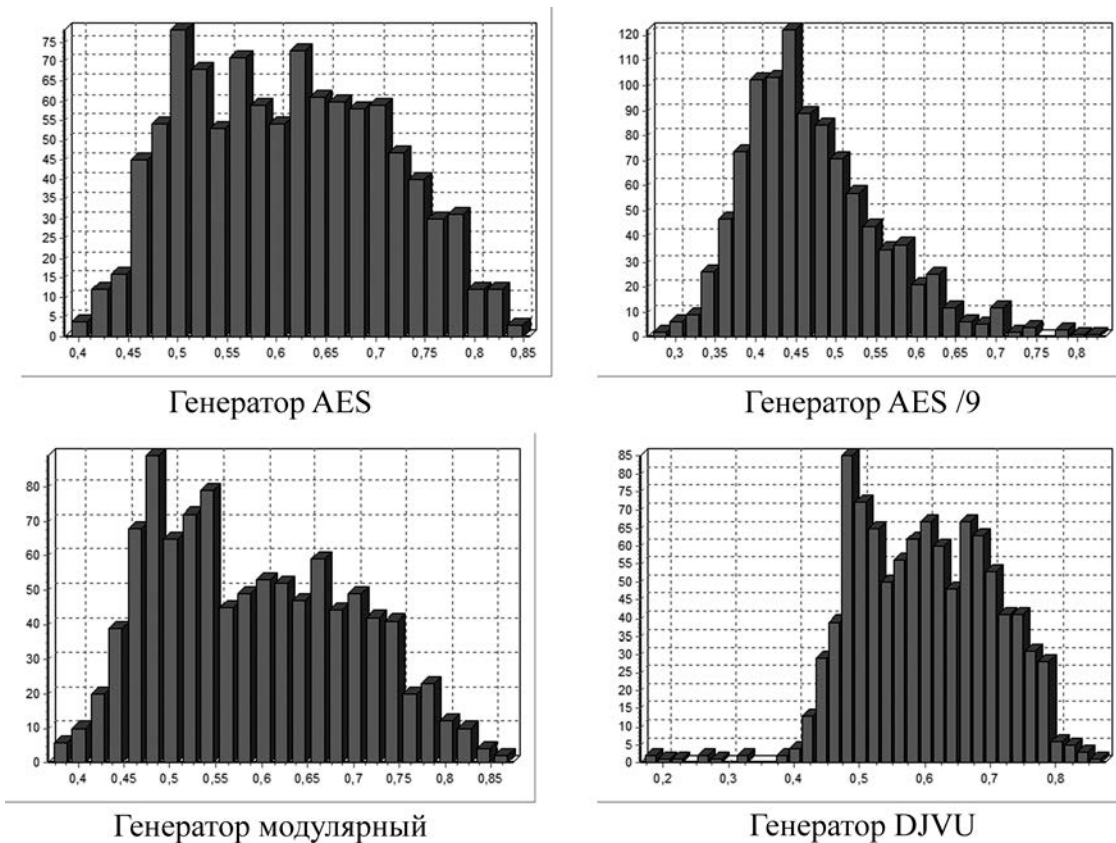


Рисунок 2 – Длина 1024 бита

Факт перехода к другому распределению является очень неудобным. Такого перехода можно избежать, если данные перед построением усреднять. Например брать по 50 значений, вычислять от них среднее и с этим средним пользоваться как одним значением. Гистограммы построенные по усредненным данным для длины 16384 бита приведены на рис. 4. Эксперименты с различными последовательностями показали, что можно пренебречь отличием в распределении даже в случае, когда данные не усредняются – то есть пользоваться нормальным распределением, параметры которого рассчитываются обычным способом.

Теперь приведем значения среднего и среднеквадратичного отклонения для длины 16384:

- для AES среднее 0,57, среднеквадратичное отклонение 0,06;
- для модулярного среднее 0,50, среднеквадратичное отклонение 0,03;
- для AES /17 среднее 0,46, среднеквадратичное отклонение 0,02;
- для DJVU среднее 0,56, среднеквадратичное отклонение 0,06.

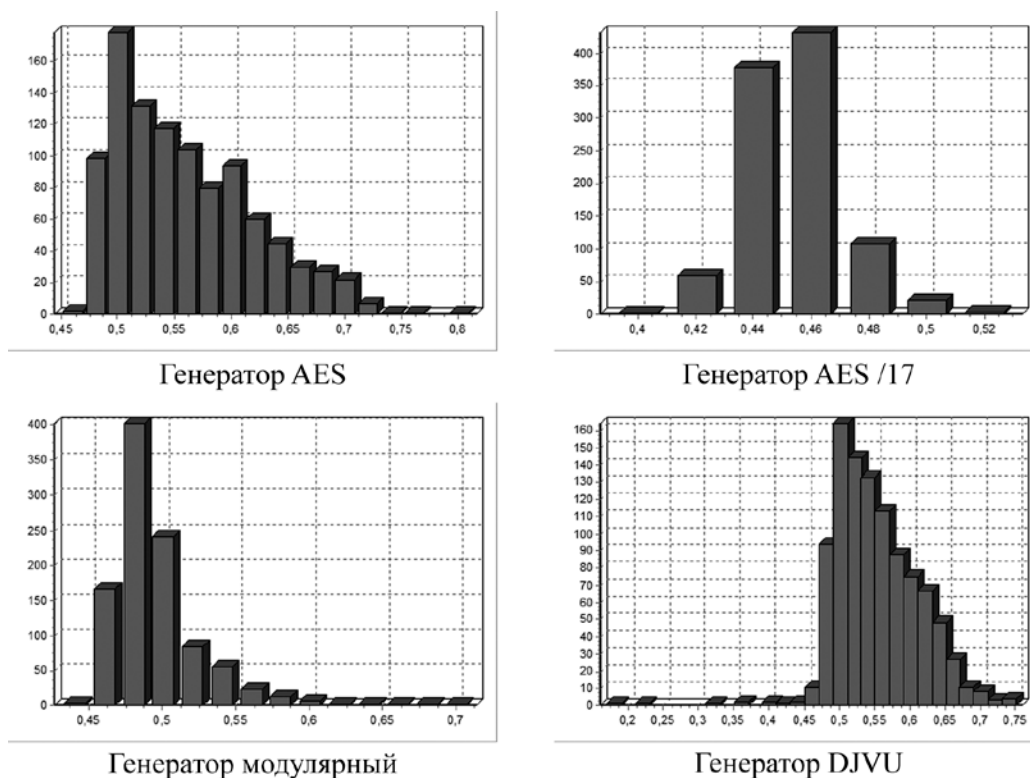


Рисунок 3 – Длина 16384 бита

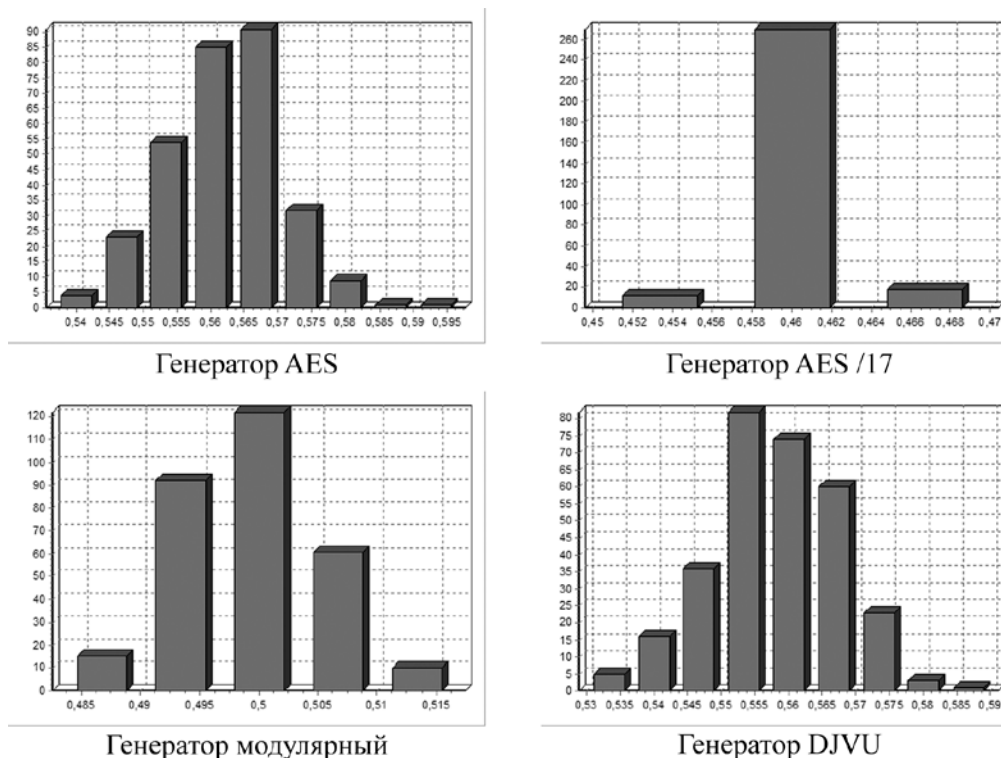


Рисунок 4 – Длина 16384 бита, с усреднениями



На рисунке 4 представлены усредненные данные выборки статистики  $U$ . Было произведено 15000 экспериментов и результаты каждых 50 подряд выполненных экспериментов были усреднены и помещены на гистограмму как одно значение. Остальные параметры эксперимента не отличаются от предыдущих графиков.

Значения среднего и среднеквадратичного отклонения для длины 16384 с усреднениями:

- для AES среднее 0,57, среднеквадратичное отклонение 0,008;
- для модулярного среднее 0,50, среднеквадратичное отклонение 0,006;
- для AES /17 среднее 0,46, среднеквадратичное отклонение 0,002;
- для DJVU среднее 0,56, среднеквадратичное отклонение 0,009.

## VII Обработка экспериментальных данных

Для практического применения критерия необходимо знать функцию распределение статистики  $U$ . По результатам экспериментов было установлено, что статистика  $U$  при использовании ее с последовательностями длиной до 4096 хорошо описывается нормальным распределением. Экспериментально установлено, что при больших длинах также можно пользоваться нормальным приближением для распределения. Параметры распределения зависят от длины и качества тестируемых последовательностей. При анализе экспериментов были выявлены следующие закономерности в параметрах функции распределения статистики  $U$ :

- при использовании одного и того же случайного генератора с ростом длины последовательности математическое ожидание и среднеквадратичное отклонение уменьшаются;
- если имеются два различных генератора, один из которых гарантированно лучше другого, то при одинаковой длине последовательности у более плохих генераторов значения математического ожидания и среднеквадратичного отклонения будут меньше, чем у хорошего.

Также эксперименты показали, что генераторы, построенные на основании стойких блочных шифров, например ранее рассматриваемый AES, очень хорошо себя ведут и показывают максимальные значения математического ожидания и среднеквадратичного отклонения. Поэтому результаты экспериментов с этим генератором были взяты в качестве эталонных данных для оценки параметров распределения статистики  $U$ . В таблице 1 приводятся результаты экспериментов для генератора AES. Таблица 1 содержит данные без усреднений.

Таблица 1 – Результаты экспериментов для генератора AES

$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$
24	0,732	0,213	384	0,647	0,124	8192	0,571	0,064	294192	0,554	0,047
48	0,664	0,168	512	0,639	0,113	16384	0,564	0,061	556336	0,553	0,046
64	0,663	0,154	768	0,622	0,103	32768	0,563	0,054	16777216	0,550	0,045
96	0,663	0,148	1024	0,610	0,099	65536	0,560	0,053	33554432	0,549	0,044
128	0,662	0,143	2048	0,604	0,088	98304	0,559	0,051	50331648	0,538	0,042
192	0,660	0,138	3072	0,592	0,081	131072	0,557	0,049	67108864	0,536	0,041
256	0,653	0,149	4096	0,583	0,068	262144	0,555	0,048	1073741824	0,525	0,031

В таблице 2 содержатся данные для среднеквадратичного отклонения, полученные с усреднением по 50-ти элементам (математическое ожидание совпадает с таблицей 1).

Таблица 2 – Результаты экспериментов для генератора AES с усреднениями

$n$	$\sigma(n)$	$n$	$\sigma(n)$	$n$	$\sigma(n)$	$n$	$\sigma(n)$
24	0,03228	384	0,01729	8192	0,00909	294192	0,00630
48	0,02268	512	0,01681	16384	0,00866	556336	0,00598
64	0,02160	768	0,01512	32768	0,00821	16777216	0,00574
96	0,01979	1024	0,01376	65536	0,00806	33554432	0,00539
128	0,01943	2048	0,01187	98304	0,00746	50331648	0,00507
192	0,01859	3072	0,01024	131072	0,00714	67108864	0,00463
256	0,01809	4096	0,00970	262144	0,00682	1073741824	0,00312

В таблице 3 приводятся результаты тестирования таких же по длине последовательностей с помощью модулярного генератора. Эти результаты предоставляются для наглядного сравнения между двумя различными по качеству генераторами.

Таблица 3 – Результаты экспериментов для модулярного генератора

$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$	$n$	$\mu(n)$	$\sigma(n)$
24	0,733	0,208	384	0,631	0,124	8192	0,528	0,054	294192	0,468	0,003
48	0,659	0,165	512	0,618	0,118	16384	0,508	0,033	556336	0,465	0,002
64	0,657	0,150	768	0,607	0,111	32768	0,492	0,016	16777216	0,461	0,0002
96	0,652	0,144	1024	0,601	0,104	65536	0,478	0,008	33554432	0,461	0,0001
128	0,648	0,142	2048	0,571	0,089	98304	0,477	0,007	50331648	0,460	0,00005
192	0,647	0,129	3072	0,550	0,073	131072	0,476	0,005	67108864	0,457	0,00003
256	0,644	0,125	4096	0,546	0,067	262144	0,469	0,003	1073741824	0,446	0,00001

На основании таблиц 1 и 3 можно построить следующий график (рис. 5)

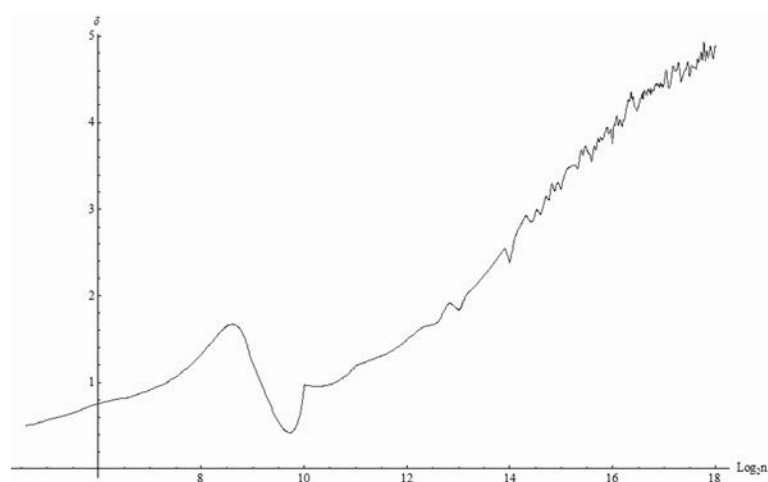


Рисунок 5 – отличие между модулярным и AES генераторами

где  $\delta = \frac{\mu(AES) - \mu(MOD)}{\sigma(AES) + \sigma(MOD)}$ ,  $\mu(AES)$ ,  $\mu(mod)$  – выборочное среднее для AES и модулярного генераторов

соответственно;  $\sigma(AES)$ ,  $\sigma(mod)$  – среднеквадратичное отклонение для AES и модулярного генераторов соответственно. На графике видно, что с помощью разработанного теста можно отличить генератор AES от модулярного генератора на последовательностях с длиной более  $2^{11}$  бит. Такое различие никак не связано с допущением о нормальности распределения. Если бы графики строились по усредненным данным, то различия были бы даже более выраженными.

«Эталонные» данные нам понадобятся в дальнейшем, причем не только приведенные в таблице значения, но и промежуточные значения. Поэтому данные, приведенные в табл. 1, с помощью регрессионного анализа, были приведены к простой для вычисления формуле следующего вида:

$$a + b \log(c + dx) \quad (7)$$

То есть, можно приближенно вычислять эталонные  $\mu$  и  $\sigma$  для «чисто случайной» последовательности с помощью следующих формул:

$$\mu(n) = 0.722 - 0.0088 \log(114.274n - 11433.364) \quad (8)$$

$$\sigma(n) = 0.226 - 0.0133 \log(6.727n - 1866.31) \quad (9)$$

Для среднеквадратичного отклонения, приведенного в табл. 2, также приводится упрощенная формула:

$$\sigma(n) = 0.0302 - 0.001465 \log(-70.325n - 1687.5) \quad (10)$$

### VIII Критерий оценки качества случайной последовательности

Суть критерия состоит в том, что по входной последовательности вычисляется статистика  $U$ , а затем проверяется гипотеза: может ли полученное значение соответствовать «хорошей» последовательности.

Для проверки того, является ли тестируемая последовательность последовательностью равновероятных независимых бит, необходимо воспользоваться приведенным ниже алгоритмом.

Входные данные:

- входная двоичная последовательность  $s$  длины  $m$  ( $m$  не менее 48);
- уровень значимости критерия  $\alpha$ .

Выходные данные:

- ответ: входная последовательность удовлетворяет критерию с заданным уровнем значимости или: нет.

Для проверки последовательности необходимо выполнить следующие шаги.

1. Если  $m < 48$ , то завершить тест с выводом: «последовательность слишком мала для применения теста».
2. Вычислить  $V = U(s)$ .
3. Руководствуясь формулами (8) и (9) вычислить (или использовать экспериментальными данные из таблицы 1 и вычислить другим образом) среднее  $\mu(m)$  и среднеквадратичное отклонение  $\sigma(m)$  для длины последовательности  $m$ .
4. Если выполняется неравенство  $u_{\alpha/2} \leq \frac{V - \mu(m)}{\sigma(m)} \leq u_{1-\alpha/2}$ , где  $u_q$  – квантиль стандартного нормального распределения уровня  $q$ , то входная последовательность **прошла** тест с заданной вероятностью, иначе последовательность тест **не прошла**.

### IX Критерий оценки качества генератора псевдослучайных последовательностей

Суть критерия состоит в том, что с помощью тестируемого генератора порождается некоторое количество последовательностей одинаковой длины. Затем проверяется гипотеза, о том, могут ли эти значения быть описаны законом, описывающим поведение эталонной последовательности, или нет.

Для проверки того, создает ли генератор псевдослучайных последовательностей последовательности равновероятных независимых бит, необходимо воспользоваться приведенным ниже алгоритмом.

Входные данные:

- генератор случайных последовательностей –  $R$  (запись  $R(l)$  следует понимать как порожденная генератором  $R$  последовательность длины  $l$ );
- длина последовательности –  $m$  ( $m$  не менее 48);
- количество экспериментов –  $n$  ( $n$  не менее 10);
- уровень значимости критерия  $\alpha$ .

Выходные данные:

- ответ: входная последовательность удовлетворяет критерию с заданным уровнем значимости или нет.

Для проверки генератора необходимо выполнить следующие шаги.

1. Если  $m < 48$ , то завершить тест с выводом «размер последовательности слишком мал для применения теста».
2. Если  $n < 10$ , то завершить тест с выводом: «выборка слишком мала для применения теста».
3. Построить случайную выборку  $\vec{X} = (X_1, X_2, \dots, X_n)$  на основании статистики  $U$ :  
 $X_j = U(R(m)), j = \overline{1, n}$ .
4. Руководствуясь формулами (8) и (10) вычислить (или использовать экспериментальными данные из таблиц 1 и 2 и вычислить другим образом) среднее  $\mu(m)$  и среднеквадратичное отклонение  $\sigma(m)$  для длины последовательности  $m$ .

5. Если выполняется неравенство  $u_{\alpha/2} \leq \frac{\sum_{j=1}^n X_j - n\mu(m)}{\sqrt{n\sigma(m)}} \leq u_{1-\alpha/2}$ , где  $u_q$  – квантиль стандартного нормального распределения уровня  $q$ , то генератор **прошел** тест с заданной вероятностью, иначе – генератор тест **не прошел**.

Генератор последовательностей следует испытывать на последовательностях, имеющих длину не меньшую, чем при планируемом практическом использовании!

Входной параметр алгоритма  $n$  является долговременным. Рекомендуемое значение – от 25-ти до 50-ти.

## Х Заключение

В данной работе приводится алгоритм тестирования случайных и псевдослучайных последовательностей, построенный с использованием контекстного моделирования. За время разработки алгоритма было выполнено множество экспериментов, в статье приводится лишь небольшая их часть.

Результаты этих экспериментов показали, что алгоритм способен отличать не только «плохие» последовательности от «хороших», но и различать «хорошие» и «испорченные» (полученные путем незначительных искажений из «хороших») последовательности.

Среди особенностей алгоритма можно выделить, то что он применим для очень коротких последовательностей – порядка 48 бит длиной. Также немаловажной особенностью алгоритма является то, что он, в отличие от большинства тестов, способен сопоставить любые две последовательности одинаковой длины между собой, а не просто сказать, проходят они тест или нет.

*Література: 1. Тесты NIST и их описание <http://www.csrc.nist.gov/rng/>. 2. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройства архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ–МИФИ, 2003. – 384 с. 3. Вербіцький О. В. Вступ до криптології. – Львів: ВТНЛ, 1998. – 250 с. 4. Кнут, Дональд, Эрвин Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд. :Пер. с англ. :Уч. пос. – М. : Издательский дом «Вильямс», 2000. – 832 с. : ил.– Парал. тит. англ.*