Scientific and Practical Cyber Security Journal (SPCSJ) 4(1): 41 - 55 ISSN 2587-4667 Scientific Cyber Security Association (SCSA)

АНАЛИЗ КРИПТОГРАФИЧЕСКОЙ СТОЙКОСТИ КОДИРОВЩИКА HiSNeC НА ОСНОВАНИИ СТАТИСТИЧЕСКИХ ТЕСТОВ NIST STS

Пенкин Юрий, доктор физико-математических наук, профессор,
Национальный фармацевтический университет, Харьков, Украина
Хара Георгий, кандидат математических наук, доцент,
Национальный фармацевтический университет, Харьков, Украина
Федосеева Алина, кандидат технических наук,
Харьковский радиотехнический колледж, Харьков, Украина

ANALYSIS OF THE CRYPTOGRAPHIC STRENGTH OF HiSNeC ENCODER BASED ON NIST STS STATISTICAL TESTS

Penkin Yuriy, Doctor of Science degree, Full professor,
National University of Pharmacy, Kharkiv, Ukraine
Khara Georgiy, PhD in Mathematical Sciences, Associate Professor,
National University of Pharmacy, Kharkiv, Ukraine
Fedoseeva Alina, PhD in Computer Science,
Kharkiv Radiotechnical College, Kharkiv, Ukraine

АННОТАЦИЯ: Представлено описание нового алгоритма генерации ключей шифрования, основанного на нелинейных трансформациях матричных структур в виде сеток Судоку. Рассмотрена реализация такого способа построения криптографических примитивов в работе блочного кодировщика *HiSNeC*. С помощью стандартных тестов *NIST STS* проведен анализ статистических свойств кодовых последовательностей шифротекстов, сгенерированных кодировщиком в режиме динамически изменяющихся входных ключей. Результатами тестирования подтверждена высокая криптографическая стойкость кодировщика, что позволяет рекомендовать *HiSNeC* для использования в системах низкоресурсной (lightweight) криптографической защиты.

ABSTRACT: The description of a new encryption key generation algorithm based on non-linear transformations of matrix structures in the form of Sudoku grids presented. The implementation of such a method for constructing cryptographic primitives in the work of the *HiSNeC* block encoder is considered. With standard NIST STS tests, we analyzed the statistical properties of cipher-text code sequences generated by the encoder in the mode of dynamically changing input keys. The test results is confirmed the high cryptographic strength of the encoder. This allows us to recommend the *HiSNeC* encoder for use in low-resource cryptographic protection systems.

КЛЮЧЕВЫЕ СЛОВА: *низкоресурсная защита данных; криптографический алгоритм; нелинейные операции матричных трансформаций; динамическая генерации ключа; блочный кодировщик HiSNeC; статистическое тестирование.*

KEYWORDS*: lightweight data protection, cryptographic algorithm, nonlinear operations of matrix transformations, dynamic key generation, block encoder HiSNeC, statistic testing.*

ВВЕДЕНИЕ. В последние годы отмечается стремительный рост объема передаваемого интернет-трафика. Наряду с традиционными интернет-устройствами, такими как персональные компьютеры, ноутбуки, смартфоны, стали появляться устройства бытовой техники, транспорта, а также различные датчики, имеющие доступ в Интернет. Это явление получило название «Internet-things». Согласно прогнозам экспертов из Gartner в 2020 году количество устройств

интернета вещей должно достигнуть отметки в 7 миллиардов единиц. Интернет для «вещей» представляет собой самоконфигурированную беспроводную сеть между объектами типа бытовых приборов, транспортных средств, различных сенсоров и датчиков (Wireless sensors), а также меток радиочастотной идентификации (Radio Frequency Identification, RFID). Уже сейчас абсолютное большинство всех изготовленных (микро) процессоров используется во встроенных Smart-устройствах или системах (all embedded CPUs 4…32 bit) и лишь единицы процентов – в традиционных компьютерах (PC & workstation CPUs 32 bit). Следует заметить, что ключевой программно-технической особенностью подавляющего большинства таких приложений является использование ограниченного количества постоянных команд управления объектами, квазипостоянных потоков данных в сенсорных сетях либо постоянных сигналов коммуникации для RFID-меток.

Развитие указанных технологий делает чрезвычайно актуальными вопросы, связанные с их информационной безопасностью. Экспертами в области безопасности использование уязвимостей умных домов и интернета вещей рассматривается как один из основных методов кибер атак. В силу условий функционирования Internet-things, а также жестких ценовых рамок (свойственных массовому производству), эти устройства характеризуются значительными ограничениями на используемые ресурсы памяти, вычислительную мощность, источники питания и т.д. Отсюда следуют ограничения [1] на используемые технологические решения для средств низкоресурсной криптографии (LWC - Lightweight Cryptography). Многие требования, предъявляемые к алгоритмам LWC, были закреплены международным стандартом ISO/IEC FDIS 29192 – Information technology – Security techniques – Lightweight cryptography. Эти требования затрагивают вопросы обеспечения секретности, аутентичности, идентификации, безотказности и ключевого обмена (data confidentiality, authentication, identification, non-repudiation and key exchange). Разумеется, при разработке решений для LWC необходимо выдерживать требуемый баланс между безопасностью, ценой и производительностью.

Наиболее продуктивными среди шифров LWC на практике оказались алгоритмы блочного шифрования, развитие которых шло по двум направлениям:

- модификация известных алгоритмов блочного шифрования в сторону их ресурсного «облегчения» (при условии незначительного снижения криптографических свойств);
- разработка новых блочных шифров, ориентированных на оптимальную реализацию (на микропрограммном или аппаратном уровне).

Так во вторую часть стандарта ISO/IEC 29192-2 (Block ciphers) уже в 2012 году были включены два алгоритма: блочный шифр PRESENT (размер информационного блока – 64-бит, размер ключа – 80 или 128 бит) и блочный шифр CLEFIA (размер информационного блока – 128-бит, размер ключа – 128, 192 или 256 бит). Также конкурентоспособными считаются реализации алгоритмов ГОСТ 28147-89 – 615 GE, KATAN– 802-1054 GE, KTANTAN – 462-688 GE, Piccolo – 683 GE, PRESENT – 1075 GE, PRINT – 402-967 GE (существуют версии для разных размеров информационных блоков), SIMON&SPECK – 763-1396, TWINE и XTEA. Основные сравнительные характеристики этих алгоритмов приведены в [1], где обоснован вывод, что с общих позиций криптоанализа практически все блочные шифры LWC на платформах 8-разрядных микроконтроллеров типа AVR и ПЛИС (FPGA) являются уязвимыми. Указанный недостаток дополняется тем, что специальной адаптации к условиям работы с постоянными командами управления объектами, квазипостоянными потоками данных в сенсорных сетях либо постоянными сигналами для RFID-меток не имеет ни один из существующих шифров LWC (включая варианты AES).

Такая ситуация определяет актуальность дальнейших разработок алгоритмов и шифров LWC, которые продолжаются и в настоящее время. Однако в ближайшей перспективе получение «идеального» (неуязвимого) решения на традиционных принципах для LWC остается маловероятным. Более того разработчики, прежде всего, обеспокоены увеличением криптостойкости шифроключей и алгоритмов их расширения. За рамками анализа остается основная угроза, связанная с практической возможностью прямой внешней атаки типа «Попугай» (название авторское). Дадим краткое описание такой атаки. В условиях использования постоянных команд управления объектами или постоянных потоков данных, а также применении постоянного входного ключа, в передаваемых зашифрованных последовательностях могут быть достаточно просто выявлены однотипные фрагменты. Эта

ситуация является подобной к известным методам атак с избранными текстами. Отличие для «Internet-things» состоит в том, что злоумышленнику, имеющему перехваченный фрагмент шифротекста, нет необходимости определять секретные ключи. Ему достаточно при энергетическом перехвате канала коммуникации многократно ретранслировать этот фрагмент шифротекста (то есть, «как попугай», многократно повторять этот сигнальный фрагмент, даже не понимая его истинного смысла). В итоге при аварийной остановке управляемого объекта (при условии срабатывания защиты от многократно повторяемых одинаковых команд), а также подлоге данных сенсоров или RFID-меток злоумышленник добивается потери со стороны истинного пользователя контроля над текущим состоянием системы управления. В любом случае перспективной для устранения этих рисков является технология динамически изменяющихся входных ключей. Один из возможных вариантов реализации такой технологии осуществлен в авторском проекте *HiSNeC* (High Speed Network Coder).

Целью данной статьи является представление способа построения криптографических примитивов, используемого в работе блочного кодировщика *HiSNeC,* и подтверждение его криптографической стойкости на основании статистических исследований с помощью тестов NIST STS.

ПРИНЦИП РАБОТЫ АЛГОРИТМА HISNEC

В основу алгоритма *HiSNeC* положен новый способ построение криптографических примитивов на основе операций нелинейной трансформации матричных структур, представленный в Патенте Украины на полезную модель № 129836 от 12.11.2018 года «Способ генерации ключей для симметричных блочных алгоритмов шифрования». Здесь математические принципы кодирования базируются на новой теории детерминированного хаоса в колебаниях дискретных структур матричного типа в виде сеток Судоку, предложенной проф. Пенкиным Ю.М. в [2].

Целесообразно напомнить, что большинство существующих алгоритмов блочного шифрования активно используют для формирования ключей шифрования данных операции: подстановок (замены одних элементов данных на другие при установке взаимно однозначного соответствия между множествами, содержащими эти данные) и перестановок (изменение порядка следования элементов данных). Например, способ, предложенный в патенте [3], базируется на аппаратной реализации контролируемых операций перестановок - криптографических примитивов, что позволяет существенно ускорить процессы шифрования. Однако следует иметь ввиду, что характерным для подобных скоростных шифров является использование предвычислений, в том числе осуществляющих и расширение секретного ключа. При этом требования по частоте смены ключей вступают в противоречие со скоростным применением шифров на базе предвычислений, поскольку необходимость многократного выполнения последних вносит существенные ограничения по быстродействию. В связи с этим весьма важным становится условие уменьшение сложности предвычислений (или отказ от них) при сохранении высокой криптостойкости преобразований. Таким образом, практически значимой является разработка скоростных шифров нового поколения, базирующихся на новых способах генерации динамических ключей, допускающих как эффективную аппаратную реализацию, так и сохраняющих высокую скорость шифрования при частой смене ключей.

В полной мере это относится и к разработкам систем LWC, для которых технология динамически изменяющихся ключей принципиально должна базироваться на «минимальной математике» (в связи с ограниченными ресурсными возможностями). Также нужно иметь ввиду, что в силу специфики представления информации в цифровых устройствах наибольший интерес представляют блочные шифры, позволяющие кодировать информацию без изменения ее структурированности, то есть позволяющие шифровать информацию, которая предварительно структурирована в соответствии с каким-либо методом протокольной защиты. Именно таким требованиям и отвечают принципы работы кодировщика *HiSNeC*.

Здесь подчеркнем ключевые моменты его программной реализации. Пусть имеется два конечных множества $\boldsymbol{\Omega_1}$ и $\boldsymbol{\Omega_2}$, каждое из которых содержит $\boldsymbol{n}$ различных элементов.

Подстановкой σ будем называть взаимно однозначное отображение Ω_1 на Ω_2. Например, если $\Omega_1 = (1,2,3)$, а $\Omega_2 = (7,11,9)$, то одна из 6-ти возможных подстановок будет иметь вид:

$$\sigma = \left(\frac{1,\ 2,\ 3}{7,\ 11,\ 9} \right).$$

Если количество элементов в каждом из множеств – n, то количество возможных подстановок равно факториалу n (то есть каждому элементу множества Ω_1 может соответствовать любой элемент из множества Ω_2). В частном случае множества Ω_1 и Ω_2 могут совпадать. Взаимно однозначное отображение множества на себя будем называть ***перестановкой*** (π). Например, если $\Omega = (1,2,3)$, существует 6 (или $3!$) перестановок:

$$\pi_1 = \left(\frac{1,\ 2,\ 3}{1,\ 2,\ 3} \right),\ \pi_2 = \left(\frac{1,\ 2,\ 3}{2,\ 3,\ 1} \right),\ \pi_3 = \left(\frac{1,\ 2,\ 3}{3,\ 1,\ 2} \right),$$

$$\pi_4 = \left(\frac{1,\ 2,\ 3}{1,\ 3,\ 2} \right),\ \pi_5 = \left(\frac{1,\ 2,\ 3}{2,\ 1,\ 3} \right),\ \pi_6 = \left(\frac{1,\ 2,\ 3}{3,\ 2,\ 1} \right).$$

Определим множество Ω как множество натуральных чисел $1,2,\ldots,n$. Латинский квадрат L будем представлять как квадратную таблицу $n \times n$, в каждой строке и в каждом столбце которой любой элемент множества Ω встречается точно один раз. Выберем значение n таким образом, чтобы $n = k^2$. Тогда латинский квадрат L с размерами $n \times n$ можно разбить на n смежных, непересекающихся квадратов размером $k \times k$. Потребуем дополнительно, чтобы в каждом малом квадрате $k \times k$ любой элемент множества Ω также встречается точно один раз (требования сетки Судоку [2]). Такой латинский квадрат назовем ***S***-квадратом, малые квадраты которого имеют размер $k \times k$ и содержат $n = k^2$ элементов.

Пусть построен ***S***-квадрат порядка $n = k^2$. Выберем один из малых квадратов и предположим, что он заполнен так, как изображено на рис. 1a. Такое расположение описывается перестановкой

$$\pi_1 = \left(\frac{1, 2, 3, 4, 5,\ 6,\ 7,\ 8,\ 9, 10, 11.\ 12, 13, 14, 15, 16}{1, 2, 3, 4, 5,\ 6,\ 7,\ 8,\ 9, 10, 11.\ 12, 13, 14, 15, 16} \right).$$

Если переставить элементы квадрата по алгоритму вихревого сдвига [2] так, как показано на рис. 1b, получим перестановку, изображенную на рис. 1c:

$$\pi_2 = \left(\frac{1, 2, 3, 4, 5,\ 6,\ 7,\ 8,\ 9, 10, 11.12, 13, 14, 15, 16}{2, 3, 4, 8, 1, 10, 6, 12, 5,\ 11, 7,\ 16,\ 9, 13, 14, 15} \right).$$

Отметим, что здесь применяются перестановки циклического типа, которые позволяют обеспечить существенно различные матричные трансформации.

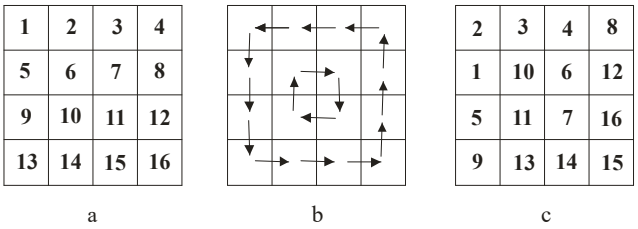

Рис. 1. *Пример перестановки элементов малого квадрата для* $k = 4$, $n = 16$.

На рис. 1b можно выделить две циклические перестановки: первая – продвижение против часовой стрелки приграничных клеток и продвижение по часовой стрелке центральных клеток таблицы. Сопоставление нижних строк перестановок π_1 и π_2 задает правила переобозначений чисел малого квадрата при выполнении перестановки π_2. Действительно 1 меняется на 2, 2 – на 3, …, 11 – на 7, …, 16 – на 15. Проведя одновременно такую же групповую замену чисел во всех малых квадратах, получим новую матричную форму большого S-квадрата.

Далее полагаем исходной нумерацию клеток малых квадратов матричной структуры в соответствии с рис. 1а. Тогда любую перестановку элементов малого квадрата можно описать вектором размерности 16. Условимся, что запись $\pi = (2,3,4,8.1,10,6,12,5,11,7,16,9,13,14,15)$, определяет следующие действия: элемент из клетки 1 переставляется в клетку 2, из клетки 2 – в клетку 3, …, из 6 – в 10, из 7 – в 6, … , из 16 – в 15. Таким образом, порядковый номер координаты вектора указывает, из какой клетки следует брать элемент для перестановки. Значение координаты вектора определяет, в какую клетку следует поместить выбранный элемент.

В анализируемом варианте работы кодировщика будем полагать, что ключи шифрования генерируются для 128-ми битной версии криптографического алгоритма AES (Advanced Encryption Standard). В связи с этим, выберем в качестве начального ***S***-квадрата некоторый квадрат S_0 (n = 16). Набор операций перестановок определим посредством задания векторов перестановки размерностью 16. Назовем эти перестановки базисными. Количество таких векторов целесообразно выбрать равным степени 2. В проводимых авторами статистических экспериментах это количество m выбиралось равным 32. Таблица операций (T_{op}) над ***S***-квадратами будет иметь размер $n \times m$ (в этом случае моделирования использовалась таблица 16×32). В общем случае, получаемые в результате этих операций структурные компоненты (строки, столбцы или малые квадраты) матриц ***S***-квадратов могут рассматриваться как генерации множеств новых ключей кодировки или их составляющих. Здесь важно подчеркнуть, что за один раунд коллективных перестановок реализуется возможность одновременной генерации не одного, а множества («связки») ключей шифрования. Причем эти генерации осуществляются без использования каких-либо дополнительных математических процедур. Более того, знание только последовательности (расписания) используемых операций T_{op} исключает необходимость хранения таблиц ***S***-квадратов после каждой выполненной операции в отличие от используемых сейчас многораундовых технологий генерации ключей. Перечисленные особенности способа генерации ключей, прежде всего, позволяют декларировать его существенные преимущества над прототипами в скорости реализации процессов кодирования и декодирования информационных потоков.

С целью «маскировки» передачи в открытом канале связи кодированных последовательностей (пусть и случайных) с постоянно используемыми натуральными числами (от 1 до 16) дополнительно формируются квадратные таблицы R_0 и R_i размером n^2 каждая. Причем матрица R_0 должна быть сформирована предварительно с помощью генератора случайных чисел (с заданной равномерностью распределения) из диапазона 0…255. Для дальнейшего получения из R_0 матрицы R_i будет использоваться та же таблица операций T_{op}, которые теперь будут определять конкретные перестановки мест ячеек этой матрицы вместе с содержащимися в них случайными числами.

Таким образом, в рассматриваемом варианте предлагаемый алгоритм генерации ключа объединяет следующую последовательность шагов:

1) вычисляются произведения M_r матриц R_0 и R_i (используются операции сложения и умножения по модулю 256);
2) вычисляются суммы s_r элементов таблицы M_r по модулю m;
3) используя s_r в качестве указателя, из таблицы T_{op} выбирается вектор перестановки;
4) выбранная операция перестановки выполняется над S_0-квадратом, в результате которой формируется новый S_i-квадрат;
5) по заданному правилу из S_i выбирается либо строка (одна из 16 строк), либо столбец (один из 16 столбцов), либо малый квадрат (один из 16). Полученные 16 чисел образуют вектор перестановки. Отметим, что в модельной реализации авторами использовались только эти 48 вариантов, хотя нет причин и для использования других вариантов выборки;
6) для формирования выборки из матрицы M_r строится вспомогательная функция $G(i,R)$, результатом которой является назначенная последовательность из 16 элементов матрицы M_r, которые и образуют текущий динамический ключ для алгоритма шифрования AES.

При оговоренных выше условиях, в соответствии с приведенным алгоритмом можно сформировать $48 \times 256 = 12288$ ключей шифрования. Далее может использоваться следующий за s_r указатель на таблицу T_{op}, и повторение шагов 3…6. Это обеспечит генерацию очередного множества из 12288 ключей. Таким образом, использование 32-х векторов перестановок, содержащихся в таблице T_{op}, обеспечивает генерацию множества из $48 \times 256 \times 32 = 393216$ существенно разных ключей шифрования, что позволяет зашифровать в рамках одного полного цикла 393216 блоков по 16 байт каждый (свыше 6 Мбайт информации). При необходимости далее следует переход к новой форме таблицы операций T_{op}. Для использования рассмотренного алгоритма следует «закрыть» в постоянной памяти контроллеров начальный S_0-квадрат и таблицы операций T_{op}. Таблица случайных чисел R_0 хранится в оперативной памяти процессоров и может при необходимости обновляться заново. Разумеется, матрицы S_0, T_{op} и R_0 должны быть одинаковыми (заданными) для всей группы устройств, которые будут обмениваться кодированной информацией.

При практической реализации защиты канала передачи информации между двумя устройствами T_t (передатчик) и T_r (приемник) осуществляется следующая последовательность процедур: 1) передатчик T_t обращается к приемнику T_r, передавая запрос на передачу; 2) при получении от T_t разрешения на передачу T_r генерирует таблицу случайных чисел R_0 и передает ее приемнику; 3) приемник и передатчик одновременно вычисляют матрицу M_r; 4) приемник и передатчик в соответствии с рассмотренным выше алгоритмом готовят синхронную генерацию одинаковых ключей: T_t для кодирования, а T_r для декодирования информационного потока. Подчеркнем, что при этом устройства различных групп (с различными S_0, T_{op} R_0) не смогут обмениваться информацией. Тоже можно утверждать и о попытке несанкционированного доступа к сеансу связи, поскольку получение действующих ключей в реальный момент времени возможно только при использовании исходных S_0, T_{op} и R_0. Эти параметры, в соответствии с вышеизложенным, являются недоступными не только гипотетическому “злоумышленнику”, но и пользователю устройства защиты канала связи.

РЕЗУЛЬТАТЫ ПРОХОЖДЕНИЯ ТЕСТОВ NIST STS

Каждый из тестов в [4], предлагаемых NIST, получает на вход конечную исследуемую двоичную последовательность из знаков 0 и 1 (рекомендуемая длина последовательности 1000000 бит). Далее вычисляется статистика, характеризующая некое свойство данной последовательности. Это может быть и единичное значение, и множество значений. После чего эта статистика сравнивается с эталонной теоретической статистикой, которая характерна для идеально случайной последовательности.

В основе такого сопоставления лежит общий алгоритм сравнений статистических гипотез. При этом за нулевую гипотезу принимается предположение, что исследуемая последовательность является действительно случайной (знаки которой появляются равновероятно и независимо от друга). В каждом тесте вычисляется так называемое P-значение: это вероятность того, что исследуемый алгоритм генерации случайной последовательности создал последовательность не хуже, чем гипотетический истинный. Если P-значение = 1, то исследуемая последовательность идеально случайна, а если оно = 0, то последовательность полностью предсказуема. В дальнейшем P-значение сравнивается с α (уровнем статистической значимости), и если оно больше α, то нулевая гипотеза принимается и последовательность признается близкой к случайной. В противном случае — отбраковывается.

В тестах NIST STS фиксируется α = 0.01. Из этого следует, что:

- если P-значение ≥ 0.01, то последовательность признается случайной с уровнем доверительной вероятности 0.99;
- если P-значение < 0.01, то последовательность отбраковывается с 99%-ным уровнем надежности.

При прохождении тестов использовались кодовые последовательности, сгенерированные кодировщиком *HiSNeC*, длиной 1000000 битов (согласно рекомендациям NIST

STS). Статистические исследования были проведены для двух режимов использования кодировщика. В первом случае обеспечивался режим динамически изменяемых 128-ми битных ключей для стандартного шифратора Rijndael (AES) с минимальным количеством раундовых процедур *Nr*=10. Во втором кодовая последовательность формировалась потоком динамически изменяемых ключей генератора *HiSNeC*, хешированных с помощью известного алгоритма MD5 (для устранения постоянных групп данных, появляющихся в представлениях используемых чисел в двоичной системе счисления).

В первом случае в качестве исходной информации для кодирования использовалась электронная версия книги [5], которая содержит множество разноформатных материалов (текст, рисунки, диаграммы, таблицы и т.д.). Тест-контролю были подвержены 300 разных кодовых последовательностей, из которых 100% последовательностей успешно прошли все 15 предлагаемых тестов. На рис. 2, как типичные, представлены результаты одного из вариантов в проведенной серии тестирования для величин P-значений. Оригиналы протокольных отчетов прохождения тестов для этого варианта приведены в Приложении к статье. Следует указать, что в этом случае параллельно был проведен эксперимент по определению времени шифрования данных с использованием предложенного способа генерации ключей в совокупности с алгоритмом AES. Эксперимент показал несущественное (0.3%) увеличение затрат процессорного времени при использовании режима динамического изменения ключа (при шифровании каждого 16-байтового блока) по сравнению с использованием алгоритма AES с постоянным ключом. При этом относительное увеличение объема кодовых последовательностей составляло не более 3%. Такие результаты тестирования позволяют полагать, что технология динамически изменяемых ключей *HiSNeC* может быть успешно совмещена и с другими типами шифраторов LWC.

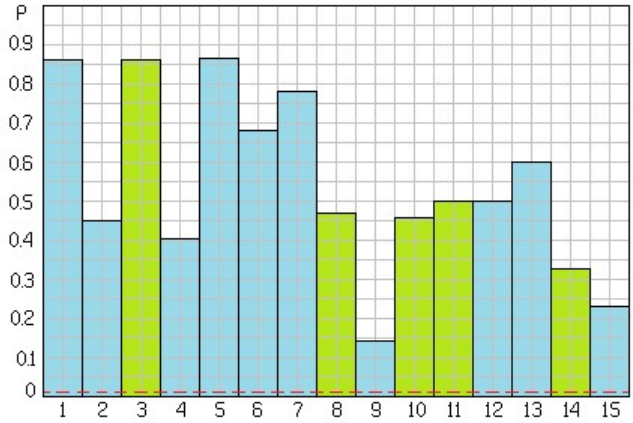

Рис. 2. *Диаграмма общих результатов тестирования*.

На рис. 2 натуральные числа, расположенные по горизонтальной оси, соответствуют номерам тестов: *1 - Approximate entropy test; 2 - Block frequency test; 3 - Cumulative sums (forward) test; 4 - FFT test; 5 - Frequency test; 6 - Linear complexity; 7 - Longest runs of ones test; 8 - Nonperiodic templates test; 9 - Overlapping template of all ones test; 10 - Random excursions test; 11 - Random excursions variant test; 12 - Rank test; 13 - Runs test; 14 - Serial test; 15 - Universal statistical test.* Голубым цветом показаны уровни единичных выходных значений вероятностей, а салатным (3,8,10,11,14) – среднее значение множества выходных P-значений.

Во втором случае из 100 исследованных кодовых последовательностей 96 успешно преодолели 11 тестов. Проблемные результаты наблюдались по тестам с номерами 10-11 и 14-15, связанных с анализом перекрывающихся шаблонов. Прежде всего, это свидетельствует, о неоптимальном порядке размещения ключей при формировании случайной последовательности. Однако, при условии дальнейшей доработки алгоритма (с целью минимизации этой уязвимости) автономное использование кодировщика на базе *HiSNeC* может быть вполне достаточным для многих практических приложений.

Таким образом статистические исследования подтвердили возможность успешного использования технологии *HiSNeC* в обеих случаях рассмотренных реализаций. Кроме того,

анализ тестированных последовательностей показал, что предложенная технология может обеспечивать режимы их генерации в соответствии с принципом шифра Г. Вернама по предварительно выбранному модулю цикличности.

ВЫВОДЫ

В статье представлено описание нового алгоритма генерации ключей шифрования, реализованного в работе блочного кодировщика *HiSNeC*. На основании тестов NIST STS исследованы статистические свойства кодовых последовательностей кодировщика, полученных в режиме динамически изменяющихся входных ключей. Результаты тестирования подтвердили требуемую криптографическую стойкость кодировщика *HiSNeC*, что позволяет рекомендовать его ПО для использования в LWC системах.

Следует также уточнить, что предлагаемый кодировщик относится к программно-аппаратным средствам защиты. Совместно с декодировщиком его целесообразно на практике производить в виде пары структурно автономных микромодулей, для которых хранилищем секретного ключа является автономная (от сети) флэш-память микроконтроллеров. Для взлома этой памяти необходим физический захват самого контроллера с программным обеспечением, а также наличие сложного специального оборудования. Кроме того, даже после предполагаемого прочтения контента этой памяти нужна серьезная работа по криптоанализу снятой информации. То есть использование в *HiSNeC* автономных модулей на базе микроконтроллеров полностью снимает риск внутренних угроз сетевого типа. Для борьбы с атаками внешнего типа здесь использован принцип хранения не самих ключей, а только цепочка процедур их генерации из исходной матрицы. Такая информация о генерации ключей состоится из двух компонентов: один из которых храниться в постоянной флэш-памяти устройств защиты, а второй является изменяемым программным способом. Последнее позволяет при необходимости производить перенастройку системы защиты без изменения прошивки флэш-памяти микромодулей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Жуков А.Е. Легковесная криптография. Часть 1 / А.Е. Жуков // Вопросы кибербезопасности, 2015. - №1(9). – С. 26 – 43.
2. Penkin Yu., Khara G. Deterministic Chaos in Vibrations of Discrete Structures of Matrix Type / Yu. Penkin, G. Khara // Proc. Inter. Scien.-Pract. Conf. on Problems of Infocommunications Science and Technology (PIC S&T 2018), October 9-12, Kharkiv (Ukr.), 2018. – Vol. 2. – P. 548 – 552.
3. Патент РФ №2309549 от 27.10.2007 г. «Способ криптографического преобразования цифровых данных».
4. Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., Leigh S., Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S. A statistical test suite for random and pseudorandom number generators for cryptographic applications // NIST Spec. Publ. 2001. 800 - 22 revision 1a: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SPS00-22b.pdf.
5. Микробиология. Руководство к лабораторным занятиям: Учеб. пособие для студентов высш. учеб. заведений / Под ред. д.м.н., проф. И.Л. Дикого. – Х.: НФаУ: Золотые страницы,2002. – 444 с.

ПРИЛОЖЕНИЕ. Оригиналы протокольных отчетов прохождения тестов

Листинги протоколов представлены в алфавитном порядке согласно названий тестов.

Тест 1.

```
          APPROXIMATE ENTROPY TEST
-------------------------------------------
COMPUTATIONAL INFORMATION:
-------------------------------------------
(a) m (block length)     = 10
(b) n (sequence length) = 1000000
(c) Chi^2                = 974.915528
(d) Phi(m)           = -6.931020
(e) Phi(m+1)         = -7.623679
(f) ApEn                 = 0.692660
(g) Log(2)               = 0.693147
```

```
                ---------------------------------------------
SUCCESS                p_value = 0.861628
```

Комментарий. Тест приблизительной энтропии проверяет близость соотношения числа вхождений перекрывающихся шаблонов длины *t* и длины *t+1* ожидаемому для случайной последовательности.

Тест 2.

```
                        BLOCK FREQUENCY TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) Chi^2           = 7827.000000
                (b) # of substrings = 7812
                (c) block length    = 128
                (d) Note: 64 bits were discarded.
                ---------------------------------------------
SUCCESS                p_value = 0.450159
```

Комментарий. Частотный блочный тест делит исследуемую последовательность на блоки одинаковой длины и проверяет равномерность распределения числа единиц в этих блоках.

Тест 3.

```
                  CUMULATIVE SUMS (FORWARD) TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) The maximum partial sum = 737
                ---------------------------------------------
SUCCESS                p_value = 0.868630

                 CUMULATIVE SUMS (REVERSE) TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) The maximum partial sum = 760
                ---------------------------------------------
SUCCESS                p_value = 0.849583
```

Комментарий. Тест кумулятивных сумм. Биты исследуемой последовательности трактуются как целые числа, причем все нули заменяются на -1. Далее вычисляются суммы первых *t* членов последовательности. Тест проверяет соответствие значения максимальной по абсолютной величине суммы ожидаемому.

Тест 4.

```
                          FFT TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) Percentile = 94.981800
                (b) N_l        = 474909.000000
                (c) N_o        = 475000.000000
                (d) d          = -0.835073
                ---------------------------------------------
SUCCESS                p_value = 0.403676
```

Комментарий. В тесте дискретного преобразования Фурье (еще имеет название Spectral Text) биты исследуемой последовательности рассматриваются как вещественные числа при условии замены каждого нуля на -1. От полученной последовательности вычисляется дискретное преобразование Фурье. Далее проверяется близость числа спектральных компонент, амплитуда которых превышает 95% к ожидаемому для случайной последовательности.

Тест 5.

```
                        FREQUENCY TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
```

```
                ---------------------------------------------
                (a) The nth partial sum = 170
                (b) S_n/n               = 0.000170
                ---------------------------------------------
SUCCESS               p_value = 0.865010
```

Комментарий. Частотный побитовый тест (Frequency Monobit Text) проверяет, что в исследуемой последовательности число единиц приблизительно равно числу нулей.

Тест 6.

```
                          LINEAR COMPLEXITY
                ---------------------------------------------
                M (substring length)     = 500
                N (number of substrings) = 2000
                ---------------------------------------------
                               FREQUENCY
                ---------------------------------------------
                  C0   C1   C2   C3   C4   C5   C6    CHI2    P-value
                ---------------------------------------------
                Note: 0 bits were discarded!
                  16   56  269  987  510  119   43  3.985116 0.678691
```

Комментарий. Тест проверяет исследуемую последовательность на линейную сложность. Линейной сложностью последовательности называется наименьшая длина регистра сдвига с линейной обратной связью, ее порождающего. Исследуемая последовательность разделяется на подпоследовательности определенной длины. Для каждой из них вычисляется линейная сложность с помощью алгоритма Берлекэмпа - Месси. Тест проверяет соответствие распределения линейной сложности этих подпоследовательностей и ожидаемого.

Тест 7.

```
                     LONGEST RUNS OF ONES TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) N (# of substrings)  = 100
                (b) M (Substring Length) = 10000
                (c) Chi^2                = 3.270145
                ---------------------------------------------
                                FREQUENCY
                ---------------------------------------------
                <=10  11  12  13  14  15 >=16 P-value  Assignment
                   6  24  22  24  12   5    7 SUCCESS   p_value = 0.774256
```

Комментарий. Тест на самую длинную последовательность единиц в блоке разбивает исследуемую последовательность на блоки фиксированной длины и проверяет соответствие распределения максимальных длин непрерывных последовательностей из единиц внутри блоков ожидаемому для случайной последовательности.

Тест 8.

```
                     NONPERIODIC TEMPLATES TEST
                ---------------------------------------------
                  COMPUTATIONAL INFORMATION
                ---------------------------------------------
       LAMBDA = 244.125000 M = 125000   N = 8  m = 9  n = 1000000
                ---------------------------------------------
                FREQUENCY
Template   W_1  W_2  W_3  W_4  W_5  W_6  W_7  W_8    Chi^2   P_value Assignment
Index
-------------------------------------------------------------------------------
--
000000001  258  238  260  241  218  261  242  252  6.463571 0.595451 SUCCESS   0
000000011  246  241  281  246  236  253  226  261  9.043703 0.338623 SUCCESS   1
000000101  240  244  240  227  239  258  273  238  6.004951 0.646677 SUCCESS   2
000000111  245  214  275  242  244  243  237  238  8.285339 0.406105 SUCCESS   3
000001001  246  263  232  246  213  281  262  250 13.527160 0.094953 SUCCESS   4
000001011  230  246  244  238  237  243  271  214  8.144470 0.419487 SUCCESS   5
```

000001101	260	241	244	267	242	238	275	226	8.934609	0.347842	SUCCESS	6
000001111	246	250	245	239	216	251	231	253	4.890707	0.769191	SUCCESS	7
000010001	259	212	224	260	267	237	249	214	14.470878	0.070288	SUCCESS	8
000010011	251	228	217	243	229	261	231	252	7.592643	0.474238	SUCCESS	9
000010101	244	252	273	233	255	251	247	252	5.318610	0.723043	SUCCESS	10
000010111	240	245	251	251	261	223	235	232	4.548596	0.804552	SUCCESS	11
000011001	232	255	265	234	230	236	260	267	7.814010	0.451846	SUCCESS	12
000011011	216	255	245	247	229	224	242	213	10.699182	0.219333	SUCCESS	13
000011101	262	231	260	237	267	246	266	235	7.978181	0.435604	SUCCESS	14
000011111	260	248	252	223	218	247	231	243	6.946551	0.542410	SUCCESS	15
000100011	245	215	208	247	259	226	228	227	13.834319	0.086187	SUCCESS	16
000100101	235	224	218	236	238	253	247	221	8.033257	0.430228	SUCCESS	17
000100111	238	241	237	241	234	260	232	241	2.623029	0.955747	SUCCESS	18
000101001	230	264	260	208	262	265	226	255	14.208205	0.076497	SUCCESS	19
000101011	250	240	248	226	230	248	263	265	5.938224	0.654152	SUCCESS	20
000101101	255	234	249	254	230	257	276	231	8.031139	0.430434	SUCCESS	21
000101111	216	251	258	272	255	239	219	223	12.836583	0.117597	SUCCESS	22
000110011	231	258	259	247	230	247	241	265	5.285776	0.726641	SUCCESS	23
000110101	251	241	224	255	270	221	257	226	9.654843	0.290092	SUCCESS	24
000110111	237	265	265	253	224	218	219	245	11.526392	0.173619	SUCCESS	25
000111001	241	251	252	248	231	248	244	240	1.433583	0.993757	SUCCESS	26
000111011	250	225	254	239	274	251	270	241	9.079715	0.335616	SUCCESS	27
000111101	263	240	254	224	256	230	263	240	6.734718	0.565506	SUCCESS	28
000111111	234	224	221	233	251	231	246	241	5.926573	0.655457	SUCCESS	29
001000011	253	258	234	236	238	234	282	221	10.799803	0.213303	SUCCESS	30
001000101	251	237	269	265	271	227	233	203	16.875189	0.031435	SUCCESS	31
001000111	264	203	236	245	234	258	220	227	14.080046	0.079703	SUCCESS	32
001001011	231	227	227	244	236	245	242	272	6.808859	0.557389	SUCCESS	33
001001101	268	226	216	259	255	243	242	226	10.012842	0.264126	SUCCESS	34
001001111	230	232	249	236	239	250	257	251	3.008566	0.933819	SUCCESS	35
001010011	260	254	254	260	267	240	228	258	7.167917	0.518629	SUCCESS	36
001010101	208	278	241	218	228	261	227	229	17.843268	0.022433	SUCCESS	37
001010111	269	257	246	237	234	257	248	246	4.768902	0.781966	SUCCESS	38
001011011	240	250	231	247	241	255	261	241	2.773431	0.947762	SUCCESS	39
001011101	253	246	215	242	264	228	245	278	11.601593	0.169884	SUCCESS	40
001011111	238	227	232	257	219	256	221	226	9.655902	0.290012	SUCCESS	41
001100101	218	252	248	268	247	237	245	248	5.949874	0.652847	SUCCESS	42
001100111	235	258	225	248	224	243	245	255	5.007215	0.756805	SUCCESS	43
001101011	251	266	213	239	262	226	266	249	11.316677	0.184393	SUCCESS	44
001101101	213	260	245	248	246	246	239	216	8.731249	0.365472	SUCCESS	45
001101111	237	256	255	250	279	237	230	243	7.678436	0.465495	SUCCESS	46
001110101	272	226	253	241	260	227	245	265	9.218466	0.324203	SUCCESS	47
001110111	258	235	231	241	270	244	232	245	5.402285	0.713840	SUCCESS	48
001111011	256	251	243	238	240	237	254	260	2.730005	0.950143	SUCCESS	49
001111101	230	243	247	246	241	280	263	236	8.183659	0.415739	SUCCESS	50
001111111	238	245	230	232	241	246	228	227	4.030663	0.854346	SUCCESS	51
010000011	258	232	264	261	242	250	238	250	4.789027	0.779869	SUCCESS	52
010000111	280	253	247	230	240	222	271	222	13.946590	0.083168	SUCCESS	53
010001011	251	246	250	258	254	223	220	223	7.837311	0.449521	SUCCESS	54
010001111	237	229	254	258	257	256	248	223	5.667077	0.684467	SUCCESS	55
010010011	241	247	222	245	251	254	261	232	4.596258	0.799727	SUCCESS	56
010010111	236	222	235	250	256	280	246	264	10.591147	0.225958	SUCCESS	57
010011011	248	228	233	256	290	254	225	237	13.380995	0.099396	SUCCESS	58
010011111	230	245	235	246	238	238	250	214	5.525148	0.700252	SUCCESS	59
010100011	240	253	239	252	276	242	243	223	6.999510	0.536686	SUCCESS	60
010100111	259	236	243	255	254	226	237	229	4.712767	0.787788	SUCCESS	61
010101011	243	243	234	222	245	253	247	220	5.356741	0.718855	SUCCESS	62
010101111	264	266	239	229	226	231	253	230	8.081979	0.425503	SUCCESS	63
010110011	241	256	248	255	255	234	250	265	4.131284	0.845089	SUCCESS	64
010110111	222	235	249	243	243	266	254	273	8.510942	0.385214	SUCCESS	65
010111011	269	260	234	239	268	226	261	286	16.677125	0.033652	SUCCESS	66
010111111	258	226	223	249	235	263	258	232	7.499437	0.483826	SUCCESS	67
011000111	249	243	232	236	226	255	260	257	4.671459	0.792044	SUCCESS	68
011001111	214	249	240	243	236	236	249	242	4.702175	0.788882	SUCCESS	69
011010111	243	261	240	244	262	243	253	256	3.574161	0.893358	SUCCESS	70
011011111	235	242	234	268	273	238	228	260	9.081833	0.335440	SUCCESS	71
011101111	243	239	230	239	279	221	235	273	12.376904	0.135163	SUCCESS	72
011111111	257	225	234	247	257	240	235	227	5.090889	0.747819	SUCCESS	73

100000000	258	238	260	241	218	261	242	252	6.463571	0.595451	SUCCESS	74
100010000	241	238	231	227	238	261	251	191	15.695276	0.046955	SUCCESS	75
100100000	246	216	281	255	225	278	275	243	20.083408	0.010025	SUCCESS	76
100101000	246	234	259	229	228	242	230	243	4.327230	0.826459	SUCCESS	77
100110000	228	232	247	249	253	232	250	239	3.074234	0.929613	SUCCESS	78
100111000	264	238	237	224	245	266	244	235	6.146880	0.630783	SUCCESS	79
101000000	270	226	258	258	239	264	229	267	10.830519	0.211488	SUCCESS	80
101000100	243	231	210	243	252	276	248	225	11.854734	0.157804	SUCCESS	81
101001000	249	236	280	252	241	219	222	236	11.165216	0.192512	SUCCESS	82
101001100	228	253	239	229	238	239	260	216	7.204988	0.514686	SUCCESS	83
101010000	249	233	242	257	257	223	217	249	7.157326	0.519758	SUCCESS	84
101010100	232	249	243	252	243	242	255	241	1.558565	0.991692	SUCCESS	85
101011000	221	254	222	234	228	241	254	235	7.095894	0.526323	SUCCESS	86
101011100	241	245	252	237	256	277	266	246	7.740927	0.459178	SUCCESS	87
101100000	239	239	245	214	216	247	238	242	7.635010	0.469910	SUCCESS	88
101100100	253	218	254	249	254	226	252	232	6.429677	0.599218	SUCCESS	89
101101000	239	262	233	259	250	243	267	227	6.537712	0.587228	SUCCESS	90
101101100	244	222	245	251	230	231	239	245	3.967113	0.860078	SUCCESS	91
101110000	235	260	256	245	242	262	291	226	14.094874	0.079326	SUCCESS	92
101110100	254	262	237	240	252	258	225	274	8.463279	0.389571	SUCCESS	93
101111000	242	263	256	254	263	222	252	253	6.718830	0.567250	SUCCESS	94
101111100	242	244	229	249	251	237	250	262	3.004329	0.934086	SUCCESS	95
110000000	246	247	251	227	218	249	265	252	6.593848	0.581018	SUCCESS	96
110000010	235	259	230	258	236	228	248	224	6.111927	0.634696	SUCCESS	97
110000100	227	229	222	224	248	235	271	234	9.912221	0.271241	SUCCESS	98
110001000	233	242	233	230	268	229	226	225	8.238736	0.410504	SUCCESS	99
110001010	240	266	230	241	212	242	239	267	9.705683	0.286294	SUCCESS	100
110010000	260	244	275	261	247	258	270	236	10.279752	0.245937	SUCCESS	101
110010010	226	249	244	242	240	231	253	277	7.226171	0.512439	SUCCESS	102
110010100	220	238	260	247	247	237	243	237	4.198011	0.838831	SUCCESS	103
110011000	227	238	258	259	223	219	234	259	9.091366	0.334647	SUCCESS	104
110011010	249	256	251	245	237	234	254	268	4.379129	0.821401	SUCCESS	105
110100000	255	250	254	243	233	281	249	256	8.049145	0.428684	SUCCESS	106
110100010	235	240	231	252	274	256	244	234	6.230554	0.621424	SUCCESS	107
110100100	249	242	246	261	249	238	247	251	1.836067	0.985632	SUCCESS	108
110101000	259	231	236	244	275	227	232	235	8.203783	0.413822	SUCCESS	109
110101010	267	214	236	256	260	234	275	232	13.102434	0.108373	SUCCESS	110
110101100	207	250	246	261	260	250	254	250	8.980153	0.343973	SUCCESS	111
110110000	248	249	252	233	229	262	230	255	4.620619	0.797248	SUCCESS	112
110110010	273	230	268	246	229	242	248	224	9.575405	0.296100	SUCCESS	113
110110100	247	282	218	238	230	254	275	231	15.190053	0.055554	SUCCESS	114
110111000	241	253	259	245	266	252	264	217	8.396552	0.395723	SUCCESS	115
110111010	265	255	239	228	232	243	238	274	8.128582	0.421012	SUCCESS	116
110111100	250	251	218	241	255	216	253	257	8.167772	0.417256	SUCCESS	117
111000000	243	254	251	246	237	244	302	233	15.563939	0.049064	SUCCESS	118
111000010	249	248	231	220	269	226	255	242	7.893447	0.443947	SUCCESS	119
111000100	232	209	256	255	245	238	214	219	13.629899	0.091937	SUCCESS	120
111000110	242	251	270	248	240	267	211	261	11.263719	0.187200	SUCCESS	121
111001000	271	243	270	265	225	263	250	224	12.669234	0.123748	SUCCESS	122
111001010	237	258	228	235	276	254	243	228	8.309700	0.403817	SUCCESS	123
111001100	217	240	212	264	226	255	257	265	13.676503	0.090597	SUCCESS	124
111010000	253	261	258	232	235	243	230	259	5.119487	0.744732	SUCCESS	125
111010010	238	240	234	264	256	255	223	245	5.331321	0.721648	SUCCESS	126
111010100	255	213	249	213	258	226	228	266	14.146773	0.078019	SUCCESS	127
111010110	207	267	248	266	233	250	231	238	11.706451	0.164790	SUCCESS	128
111011000	270	256	255	234	225	239	239	261	7.347976	0.499596	SUCCESS	129
111011010	263	248	222	229	245	253	244	243	4.958493	0.762003	SUCCESS	130
111011100	260	254	276	240	276	235	267	227	13.974129	0.082442	SUCCESS	131
111100000	226	244	219	259	254	223	268	223	11.613244	0.169312	SUCCESS	132
111100010	248	234	246	259	237	241	236	243	1.991765	0.981263	SUCCESS	133
111100100	257	265	247	244	229	258	251	222	6.642570	0.575641	SUCCESS	134
111100110	242	241	232	279	223	235	239	261	9.397465	0.309883	SUCCESS	135
111101000	232	262	248	249	273	262	233	247	7.586288	0.474888	SUCCESS	136
111101010	239	210	252	246	240	234	242	241	5.889502	0.659608	SUCCESS	137
111101100	264	231	275	226	234	254	236	236	9.240708	0.322399	SUCCESS	138
111101110	254	274	261	212	255	237	226	258	12.696773	0.122717	SUCCESS	139
111110000	237	217	214	235	252	237	252	233	8.794799	0.359901	SUCCESS	140
111110010	247	247	242	221	250	235	255	237	3.569925	0.893696	SUCCESS	141

```
111110100  208  234  243  272  253  266  249  227 12.964742 0.113069 SUCCESS 142
111110110  256  229  230  249  232  278  246  231  8.741840 0.364540 SUCCESS 143
111111000  240  229  233  239  250  248  244  259  2.824271 0.944897 SUCCESS 144
111111010  222  221  240  279  253  246  253  230 11.092133 0.196533 SUCCESS 145
111111100  256  248  219  243  267  255  237  258  7.089539 0.527004 SUCCESS 146
111111110  257  225  234  247  257  240  235  227  5.090889 0.747819 SUCCESS 147
```

Комментарий. Тест на совпадение неперекрывающихся шаблонов. Проверяется соответствие числа повторений некоторой фиксированной подпоследовательности (ее называют шаблоном) в исследуемой последовательности ожидаемому. При этом для поиска шаблона длины t применяется скользящее окно: если шаблон не обнаружен, то окно сдвигается на одну позицию, а если обнаружен - на t позиций.

Тест 9.

```
     OVERLAPPING TEMPLATE OF ALL ONES TEST
---------------------------------------------
COMPUTATIONAL INFORMATION:
---------------------------------------------
(a) n (sequence_length)      = 1000000
(b) m (block length of 1s)   = 9
(c) M (length of substring)  = 1032
(d) N (number of substrings) = 968
(e) lambda [(M-m+1)/2^m]     = 2.000000
(f) eta                      = 1.000000
---------------------------------------------
   FREQUENCY
  0   1   2   3   4 >=5   Chi^2   P-value  Assignment
---------------------------------------------
363 205 126  82  58 134  8.224142 0.144308 SUCCESS
```

Комментарий. Тест на совпадение перекрывающихся шаблонов. Аналогичен тесту на совпадение неперекрывающихся шаблонов, за исключением того, что окно всегда сдвигается на одну позицию.

Тест 10.

```
          RANDOM EXCURSIONS TEST
-------------------------------------------
COMPUTATIONAL INFORMATION:
-------------------------------------------
(a) Number Of Cycles (J) = 1113
(b) Sequence Length (n)  = 1000000
(c) Rejection Constraint = 500.000000
-------------------------------------------
SUCCESS          x = -4 chi^2 =  7.503211 p_value = 0.185824
SUCCESS          x = -3 chi^2 =  3.402113 p_value = 0.638248
SUCCESS          x = -2 chi^2 =  6.049749 p_value = 0.301408
SUCCESS          x = -1 chi^2 =  5.398922 p_value = 0.369157
SUCCESS          x =  1 chi^2 =  2.525606 p_value = 0.772634
SUCCESS          x =  2 chi^2 =  2.581156 p_value = 0.764226
SUCCESS          x =  3 chi^2 =  8.989443 p_value = 0.109486
SUCCESS          x =  4 chi^2 =  1.801423 p_value = 0.875883
```

Комментарий. В тесте на произвольные отклонения биты исследуемой последовательности трактуются как целые числа, причем все нули заменяются на -1. Далее полагается S_t – сумма первых t членов последовательности для $t=1,2,\ldots n$. Множество всех различных значений S_t рассматривается как множество вершин графа, а последовательность значений S_t как блуждание по этому графу. Тестом оценивается соответствие распределения количества циклов в этой последовательности (под циклом в тесте понимается последовательность вершин, начинающаяся и заканчивающаяся нулем), в которых присутствуют определенные вершины определенное число раз к ожидаемому.

Тест 11.

```
     RANDOM EXCURSIONS VARIANT TEST
-------------------------------------------
COMPUTATIONAL INFORMATION:
```

```
                ------------------------------------------
                (a) Number Of Cycles (J) = 1113
                (b) Sequence Length (n)  = 1000000
                ------------------------------------------
SUCCESS               (x = -9) Total visits = 1326; p-value = 0.273540
SUCCESS               (x = -8) Total visits = 1323; p-value = 0.250457
SUCCESS               (x = -7) Total visits = 1262; p-value = 0.381087
SUCCESS               (x = -6) Total visits = 1224; p-value = 0.478104
SUCCESS               (x = -5) Total visits = 1147; p-value = 0.810166
SUCCESS               (x = -4) Total visits = 1082; p-value = 0.803870
SUCCESS               (x = -3) Total visits = 1098; p-value = 0.886936
SUCCESS               (x = -2) Total visits = 1121; p-value = 0.922015
SUCCESS               (x = -1) Total visits = 1103; p-value = 0.832145
SUCCESS               (x =  1) Total visits = 1100; p-value = 0.782903
SUCCESS               (x =  2) Total visits = 1035; p-value = 0.339836
SUCCESS               (x =  3) Total visits =  989; p-value = 0.239847
SUCCESS               (x =  4) Total visits = 1008; p-value = 0.400259
SUCCESS               (x =  5) Total visits =  995; p-value = 0.404463
SUCCESS               (x =  6) Total visits =  864; p-value = 0.111552
SUCCESS               (x =  7) Total visits =  784; p-value = 0.053110
SUCCESS               (x =  8) Total visits =  763; p-value = 0.055441
SUCCESS               (x =  9) Total visits =  866; p-value = 0.204182
```

Комментарий. Вариант теста на произвольные отклонения, где рассматривается тот же граф, что и в предыдущем тесте, однако проверяется соответствие распределения числа проходов блуждания через каждую вершину и ожидаемого.

Тест 12.

```
                              RANK TEST
                ---------------------------------------------
                COMPUTATIONAL INFORMATION:
                ---------------------------------------------
                (a) Probability P_32 = 0.288788
                (b)             P_31 = 0.577576
                (c)             P_30 = 0.133636
                (d) Frequency   F_32 = 284
                (e)             F_31 = 574
                (f)             F_30 = 118
                (g) # of matrices    = 976
                (h) Chi^2            = 1.388266
                (i) NOTE: 576 BITS WERE DISCARDED.
                ---------------------------------------------
SUCCESS               p_value = 0.499507
```

Комментарий. Тест рангов бинарных матриц (Binary Matrix Rank Test). Исследуемая последовательность разбивается на подпоследовательности некоторой длины, из которых составляется двоичная матрица. Тест проверяет соответствие количеств матриц максимального ранга и матриц ранга на единицу меньше максимального ожидаемым для случайной последовательности.

Тест 13.

```
                              RUNS TEST
                ----------------------------------------
                COMPUTATIONAL INFORMATION:
                ----------------------------------------
                (a) Pi                        = 0.500085
                (b) V_n_obs (Total # of runs) = 499739
                (c) V_n_obs - 2 n pi (1-pi)
                    -----------------------   = 0.369089
                      2 sqrt(2n) pi (1-pi)
                ----------------------------------------
SUCCESS               p_value = 0.601690
```

Комментарий. Тест на последовательность одинаковых битов. Анализирует отклонения числа непрерывных серий от ожидаемого для случайной последовательности (под непрерывной

серией длины t понимается последовательность из t единиц, ограниченная нулями, либо последовательность из t нулей, ограниченная единицами).

Тест 14.

```
                    SERIAL TEST
          ---------------------------------------------
           COMPUTATIONAL INFORMATION:
          ---------------------------------------------
          (a) Block length    (m) = 16
          (b) Sequence length (n) = 1000000
          (c) Psi_m               = 65807.249408
          (d) Psi_m-1             = 32810.201088
          (e) Psi_m-2             = 16211.177472
          (f) Del_1               = 32997.048320
          (g) Del_2               = 16398.024704
          ---------------------------------------------
SUCCESS         p_value1 = 0.185328
SUCCESS         p_value2 = 0.467667
```

Комментарий. Тест на подпоследовательности определяет, соответствует ли ожидаемому количество вхождений каждого из 2^t шаблонов длины t (в случайной последовательности такие шаблоны равновероятны).

Тест 15.

```
                UNIVERSAL STATISTICAL TEST
          ---------------------------------------------
          COMPUTATIONAL INFORMATION:
          ---------------------------------------------
          (a) L         = 7
          (b) Q         = 1280
          (c) K         = 141577
          (d) sum       = 876774.744346
          (e) sigma     = 0.002768
          (f) variance  = 3.125000
          (g) exp_value = 6.196251
          (h) phi       = 6.192918
          (i) WARNING:  1 bits were discarded.
          ----------------------------------------
SUCCESS         p_value = 0.228651
```

Комментарий. Универсальный статистический тест Маурера вычисляет сумму логарифмов расстояний между одинаковыми шаблонами в исследуемой последовательности и проверяет ее близость к ожидаемой для случайной последовательности. Фактически тест проверяет невозможность сжатия последовательности. Тест позволяет выявлять широкий класс статистических дефектов, поэтому называется универсальным.