ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**
**( Н И У « Б е л Г У » )**

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ
СИСТЕМ И ТЕХНОЛОГИЙ

# РАЗРАБОТКА МЕТОДА ФОРМИРОВАНИЯ ПСП ДЛЯ ПОТОЧНОГО ШИФРОВАНИЯ ДАННЫХ

Выпускная квалификационная работа
обучающегося по направлению подготовки
11.04.02 Инфокоммуникационные технологии и системы связи,
магистерская программа «Системы и устройства радиотехники и связи»
очной формы обучения, группы 07001636
Картамышева Александра Вячеславовича

Научный руководитель
кандидат технических наук, доцент,
доцент кафедры ИТСиТ
НИУ «БелГУ» Буханцов А.Д.

Рецензент
кандидат технических наук, доцент,
доцент кафедры Организации и
технологии защиты информации
Белгородского университета
кооперации, экономики и права
Земляченко В.В.

БЕЛГОРОД 2018

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

В последнее время вырос интерес к вопросам защиты информации в связи с развитием компьютерных систем для передачи, хранения, сбора информации. Развитие компьютерных сетей вызывает появление больших возможностей для несанкционированного доступа к передаваемой информации и хранимой информации. Что в свою очередь, вызывает необходимость обеспечения целостности, конфиденциальности и доступности информации. Существуют различные способы защиты информации: физические (препятствие); законодательные; управление доступом; криптографическое закрытие. Наибольший интерес представляют криптографические способы защиты информации.

Криптография в переводе с древнегреческого означает "тайнопись". Это последовательность символов (открытый текст) подвергается некоторому преобразованию (в котором используется ключ), и в результате получается закрытый текст, непонятный тому, кто не знает алгоритма шифрования и ключа.

Для преобразования (шифрования) обычно используется некоторый алгоритм или устройство, реализующее заданный алгоритм, которые могут быть известны широкому кругу лиц. Управление процессом шифрования осуществляется с помощью периодически меняющегося кода ключа, обеспечивающего каждый раз оригинальное представление информации при использовании одного и того же алгоритма или устройства. Знание ключа позволяет просто и надёжно расшифровать текст. Но без знания ключа эта процедура может быть практически невыполнима даже при известном алгоритме шифрования. Даже простое преобразование информации является весьма эффективным средством, дающим возможность скрыть её смысл от большинства неквалифицированных нарушителей.

Неотъемлемым элементом практически любой системы шифрования, независимо от ее сложности и назначения, являются программные и

программно-аппаратные средства генерации псевдослучайных последовательностей (ПСП).

В задачах шифрования генераторы ПСП используются либо непосредственно, либо в составе генераторов случайных последовательностей, либо на их основе строятся алгоритмы хеширования информации. В последних двух случаях качество операций генерации случайных последовательностей и хеширования определяется в первую очередь качеством используемых генераторов ПСП.

Таким образом, именно от свойств генераторов ПСП, особенно в тех случаях, когда необходимо обеспечить надежную работу систем шифрования при наличии случайных и умышленных деструктивных воздействий, в значительной степени зависит целостность и сохранность информации. К ним предъявляются жесткие требования, в первую очередь по таким параметрам, как непредсказуемость, статистические и периодические свойства.

В настоящий момент существует трудно разрешимое противоречие между непредсказуемостью генераторов ПСП, с одной стороны, и их производительностью и эффективностью реализации. Поэтому актуальной научной задачей является создание новых алгоритмов генерации ПСП, сочетающих в себе непредсказуемость, высокое быстродействие и эффективную реализацию на различных платформах.

Целью данной выпускной квалификационной работы является: разработка и исследование метода генерации ПСП для применения в задачах поточного шифрования данных.

Для достижения поставленной цели необходимо сформируем следующие задачи, которые будут решены в ходе выполнения данной ВКР:

1) Сформулировать требования, предъявляемые к системе потокового шифрования;

2) Провести анализ существующих алгоритмов формирования ПСП;

3) Провести оценку существующих методов анализа “случайности” методов формирования ПСП;

4) Разработать и исследовать метод формирования ПСП для задач шифрования данных;

5) Провести сравнительные исследования разработанного метода генерации ПСП и стандартных методов генерации ПСП;

6) Разработать рекомендации по использованию генераторов ПСП в задачах шифрования данных.

# ГЛАВА 1 АНАЛИЗ ПОДХОДОВ К ФОРМИРОВАНИЮ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

## 1.1 Основы поточного шифрования

Потоковый шифр – симметричный шифр, преобразующий каждый символ открытого текста в символ шифрованного, зависящий от ключа и расположения символа в тексте. Отметим, что специалисты используют термин «потоковый шифр» только в том случае, когда шифруемые символы открытого текста малы (одна буква, бит или байт). [2]

Авторитетный швейцарский криптограф Райнер Рюппель главное отличие потоковых и блоковых шифров сформулировал так: блоковые шифры применяют одно и то же криптографическое преобразование к большим блокам данных открытого текста; потоковые шифры реализуют сменное по времени преобразование отдельных битов (байтов) открытого текста.

Так как посимвольное шифрование при потоковом шифровании данных не приводит к задержкам в криптосистеме, то важнейшее достоинство потоковых шифров – высокая скорость шифрования, соизмеримая со скоростью поступления входной информации. Это позволяет обеспечить шифрование практически в реальном масштабе времени вне зависимости от объема информации и разрядности потока данных. [29]

Классические примеры потоковых шифров – шифр Вернама, или одноразовый блокнот. Уже отмечалось, что если для гаммы последовательность битов выбирается случайно, и длина гаммы равна длине сообщения, то взломать шифр невозможно. Но у данного режима шифрования есть и отрицательная особенность – проблемы с передачей и хранением ключей, ведь ключи, сравнимые по длине с передаваемыми сообщениями, трудно использовать на практике.

Поэтому основная идея современных потоковых шифров – реализовать концепцию одноразового блокнота, используя секретный ключ меньшей

длины, из которого для гаммы генерируется псевдослучайная числовая последовательность, похожая на случайную. Самые популярные потоковые шифры можно назвать двоичными аддитивными: в них секретный ключ $K$ используется только для управления генератором псевдослучайной числовой последовательности, порождающим битовую последовательность $k_1, k_2, ..., k_n$ называемую ключевым потоком или гаммой. Поток битов $k_1, k_2, ..., k_n$ далее суммируется по модулю два с потом битов открытого текста $x_1, x_2, ..., x_n$ (операция XOR аппаратно реализована в арифметико-логическом устройстве процессора).[10] В итоге появляется поток битов шифрованного текста $y_1, y_2, ..., y_n$ где $y_i = x_i \oplus \gamma_i$. Гамма является псевдослучайной последовательностью, которую требуется сформировать на стороне отправителя и получателя. Для дешифрования получатель над битами шифротекста и той же самой гаммы повторно выполняет операцию XOR: $x_i = y_i \oplus \gamma_i$. Схема потокового шифрования в режиме гаммирования представлена на рисунке 1.1. Фактически секретный ключ небольшой размерности использует роль ядра, порождающего значительно более длительную ключевую последовательность.

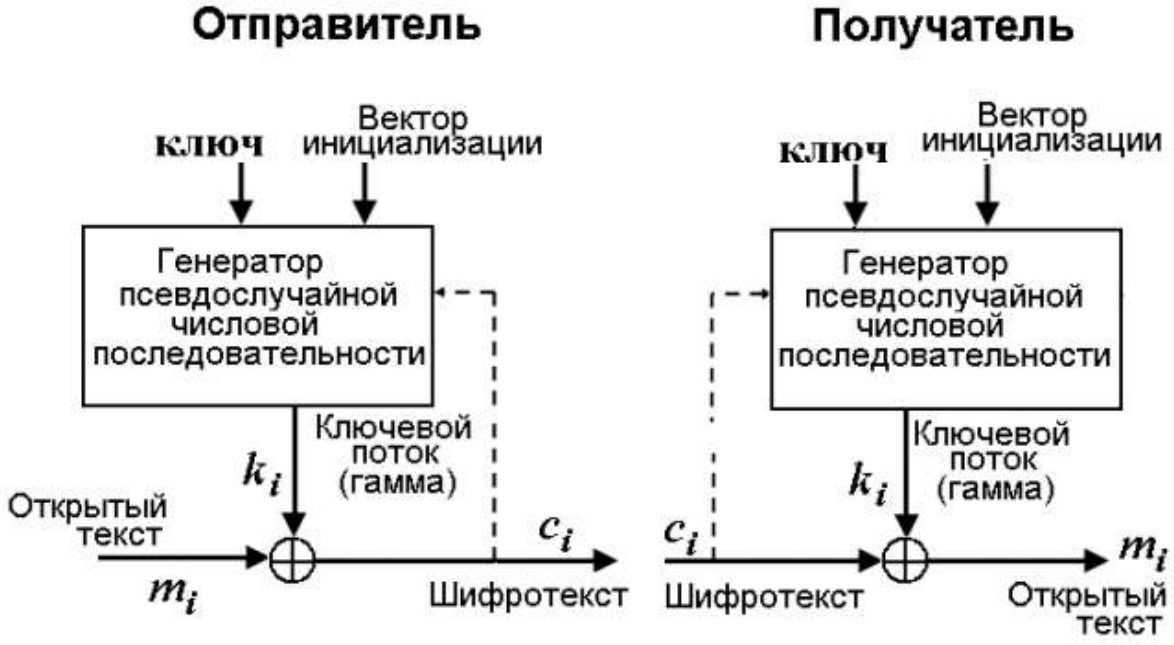


Рисунок 1.1 – Блок схема процесса потокового шифрования [14]

Повторное использование одной и той же гаммы при потоковом шифровании, как отмечалось, может вызвать катастрофическую потерю стойкости шифрования. К сожалению, сменить ключи перед началом шифрования новых сообщений не всегда технично возможно. Чтобы примирить эти обстоятельства, в потоковых шифрах начали использовать «начальное состояние» шифратора. Действительно, двойной запуск шифратора без смены ключа, но с разными начальными состояниями, даст две абсолютно разные гаммы. Комбинацию знаков, передаваемую по каналу связи и предназначенную для вхождения в связь шифрующей аппаратуры на стороне отправителя и получателя, называют вектором инициализации (синхропосылкой). [2] Его передают перед началом сообщения и сохраняют вместе с шифротекстом. Открытая передача вектора инициализации в том виде, как он используется в алгоритме шифрования, может снизить стойкость системы. Поэтому вектор инициализации закрывают с помощью дополнительного ключа (ключ инициализации) или применяют в алгоритме модификации векторов, зависящие от такого ключа. Таким образом, получатель и отправитель корреспонденции могут вычислять гамму по известному алгоритму на основе ключа и вектора инициализации (вектор часто является меткой времени). Гаммирование чувствительно к синхронизации гаммы и шифротекста. Если в ходе передачи будет поврежден один символ шифротекста, то это не повлияет на правильность дешифрования остальных символов. Если же при передаче потерять хотя бы один знак криптограммы, то весь дальнейший текст дешифруется неправильно. Поэтому необходимо обеспечить синхронизацию зашифрования и расшифрования текстов. [12]

Основным недостатком потоковых шифров является необходимость передачи информации для начальной инициализации (синхропосылка) перед заголовком сообщения, которая должна быть принята до расшифровывания любого сообщения.

## 1.2 Псевдослучайные последовательности в потоковом шифровании и теоретические основы их формирования

Поскольку потоковый шифр максимально должен имитировать одноразовый блокнот, то шифрующая гамма должна по своим свойствам походить на истинно случайную числовую последовательность. Устройство, с помощью которого реализуют истинно случайную последовательность, называют ее генератором, а члены последовательности – случайными числами. Известно, что при реализации криптографических преобразований, используют различные случайные первичные состояния либо целые последовательности. Отсюда следует, что стойкость криптопреобразования напрямую зависит от алгоритма формирования случайных чисел и последовательностей, точнее от их степени случайности. Современные компиляторы обладают собственной реализацией генератора псевдослучайных последовательностей, однако с криптографической точки зрения они являются непригодными [14].

Основная сложность генерации последовательности псевдослучайных чисел на компьютере в том, что компьютеры детерминистичны по своей сути. Компьютер может находиться только в конечном количестве состояний (количество состояний огромно, но все-таки конечно). Следовательно, любой датчик случайных чисел по определению периодичен. Все периодическое – предсказуемо, т.е. не случайно. Лучшее, что может произвести компьютер – это псевдослучайная последовательность. Период такой последовательности должен быть таким, чтобы конечная последовательность разумной длины не была периодической.

Относительно короткие непериодические подпоследовательности должны быть как можно более неотличимы от случайных последовательностей, в частности, соответствовать различным критериям случайности. Генератор последовательности псевдослучаен, если он выглядит случайным, т.е. проходит все статистические тесты случайности. Для

криптографических приложений статистической случайности недостаточно, хотя это и необходимое свойство. Последовательность называется криптографически надежной псевдослучайной последовательностью (КНПСП), если она непредсказуема, т.е. вычислительно неосуществимо предсказать следующий бит, имея полное знание алгоритма (или аппаратуры) и всех предшествующих битов потока. КНПСП тоже подвержены криптоанализу – как любой алгоритм шифрования [21].

Генератор последовательности называется случайным, если он не может быть достоверно воспроизведен, т.е. дважды запуская генератор с абсолютно одинаковыми исходными данными (по крайней мере, на пределе человеческих возможностей), мы получим случайные различные последовательности. Вопрос существования случайных последовательностей – философский вопрос. Мы знаем, что на микроуровне случайность существует (квантовая механика), но сохранит ли эта случайность при переходе на макроуровень. Дополнительное свойство случайной последовательности: случайная последовательность не может быть сжата.

Основа любого генератора ПСП — нелинейное преобразование, используемое либо в качестве функции обратной связи генератора, либо в качестве функции выхода. В настоящее время существуют несколько подходов к построению этих преобразований. Перечислим основные из них.

Использование многократного повторения одной и той же раундовой операции, в состав которой входят преобразования замены и перестановки. Смысл конструкции - обеспечить свойства рассеивания и перемешивания информации, которые, согласно К. Шеннону, необходимы любому непредсказуемому генератору ПСП [31]. Подход применяется при построении блочных генераторов.

Все существующие государственные стандарты: российский (ГОСТ 28147-89) и американский (AES) используют этот принцип. Недостатком генераторов данного класса является низкое быстродействие. Это цена, которую приходится платить, чтобы обеспечить непредсказуемость

формируемых последовательностей. Однако утверждение о наличии у генератора этого свойства основывается на недоказуемом предположении о том, что у противника не хватит ресурсов (вычислительных, временных или материальных) чтобы инвертировать используемую нелинейную функцию обратной связи или выхода [27].

Другим подходом является использование односторонних функций. Понятие односторонней функции введено Диффи и Хеллманом в 1978 году, однако до сих пор ни одной односторонней функции не найдено. Существуют и реально используются лишь функции-кандидаты на звание односторонних, иначе говоря, функции, которые вероятно являются односторонними [37]. Задача инвертирования этих функций эквивалентна решению трудной математической задачи, например, задачи факторизации больших чисел, на основе которой построен генератор RSA, долгое время являвшийся де-факто неофициальным мировым стандартом для генераторов ПСП этого класса. Недостатки генераторов этого класса те же, что и в предыдущем случае, причем их быстродействие во много раз ниже, чем у блочных генераторов, которые сами считаются медленными.

Все остальные подходы к построению генераторов ПСП позволяют в той или иной степени решить проблему быстродействия, но обязательно в ущерб свойству непредсказуемости. Подавляющее большинство фактов компрометации генераторов ПСП ответственного назначения относятся именно к алгоритмам, отличным от двух, описанных выше.

Все остальные подходы к построению генераторов ПСП позволяют в той или иной степени решить проблему быстродействия, но обязательно в ущерб свойству непредсказуемости. Подавляющее большинство фактов компрометации генераторов ПСП ответственного назначения относятся именно к алгоритмам, отличным от двух, описанных выше.

Существует проблема оценки степени случайности псевдослучайной последовательности. Только по исходным данным трудно сказать, каким является наблюдаемый процесс — случайным или хаотическим, потому что

практически не существует явного чистого 'сигнала' отличия. Всегда будут некоторые помехи, даже если их округлять или не учитывать. Это значит, что любая система, даже если она детерминированная, будет содержать немного случайностей.

Чтобы отличить детерминированный процесс от стохастического, нужно знать, что детерминированная система всегда развивается по одному и тому же пути от данной отправной точки [2]. Таким образом, чтобы проверить процесс на детерминизм необходимо:

- Выбрать тестируемое состояние.
- Найти несколько подобных или почти подобных состояний.
- Сравнить их развитие во времени.

Погрешность определяется как различие между изменениями в тестируемом и подобном состояниях. Детерминированная система будет иметь очень маленькую погрешность (устойчивый, постоянный результат) или она будет увеличиваться по экспоненте со временем (хаос). Стохастическая система будет иметь беспорядочно распределенную погрешность.

По существу, все методы определения детерминизма основываются на обнаружении состояний, самых близких к данному тестируемому (то есть, измерению корреляции, экспоненты Ляпунова, и т. д.). Чтобы определить состояние системы обычно полагаются на пространственные методы определения стадии развития. Исследователь выбирает диапазон измерения и исследует развитие погрешности между двумя близлежащими состояниями. Если она выглядит случайной, тогда нужно увеличить диапазон, чтобы получить детерминированную погрешность. Кажется, что это сделать просто, но на деле это не так. Во-первых, сложность состоит в том, что, при увеличении диапазона измерения, поиск близлежащего состояния требует намного большего количества времени для вычислений чтобы найти подходящего претендента [7]. Если диапазон измерения выбран слишком маленьким, то детерминированные данные могут выглядеть случайными, но

если диапазон слишком большой, то этого не случится — метод будет работать.

Когда в нелинейную детерминированную систему вмешиваются внешние помехи, её траектория постоянно искажается. Более того, действия помех усиливаются из-за нелинейности, и система показывает полностью новые динамические свойства. Статистические испытания, пытающиеся отделить помехи от детерминированной основы или изолировать их, потерпели неудачу [13]. При наличии взаимодействия между нелинейными детерминированными компонентами и помехами, в результате появляется динамика, которую традиционные испытания на нелинейность иногда не способны фиксировать [21].

Итак, существует трудно разрешимое противоречие между непредсказуемостью генераторов ПСП, с одной стороны, и их производительностью, и эффективностью реализации. Поэтому актуальной научной задачей является создание новых алгоритмов генерации ПСП, сочетающих в себе непредсказуемость, высокое быстродействие и эффективную реализацию на различных платформах. Одним из направлений решения данной задачи является исследование стохастических алгоритмов формирования цифровых последовательностей и разработка на их основе непредсказуемых генераторов ПСП. Указанные алгоритмы генерации ПСП основаны на использовании стохастических сумматоров, т. е. сумматоров с непредсказуемым результатом работы (R-блоков), впервые предложенных С.А. Осмоловским для решения задач помехоустойчивого кодирования. Стохастические генераторы ПСП сочетают в себе высокое качество формируемых последовательностей и эффективную программную и аппаратную реализацию [19]. Вторая проблема, с которой приходится сталкиваться при разработке программных средств генерации ПСП, - отсутствие инструментальных средств для статистического исследования формируемых последовательностей. Более того, этим исследованиям уделялось мало внимания.

## 1.3 Источники формирования псевдослучайных последовательностей

Источники настоящих случайных чисел найти крайне трудно. Физические шумы, такие, как детекторы событий ионизирующей радиации, дробовой шум в резисторе или космическое излучение, могут быть такими источниками. Однако применяются такие устройства в приложениях сетевой безопасности редко. Сложности также вызывают грубые атаки на подобные устройства. У физических источников случайных чисел существует ряд недостатков:

- Время- и трудозатраты при установке и настройке по сравнению с программными методами формирования псевдослучайных последовательностей (ПМФПП);
- Дороговизна;
- Генерация последовательностей случайных чисел происходит медленнее, чем при реализации ПМФПП.
- Невозможность воспроизведения ранее сгенерированной последовательности случайных чисел.

В то же время случайные числа, получаемые из физического источника могут использоваться в качестве порождающего элемента (seed) для ПМФПП. Такие комбинированные генераторы применяются в криптографии, лотереях, игровых автоматах [18].

Никакой детерминированный алгоритм не может генерировать полностью случайные числа, он может только аппроксимировать некоторые их свойства. Как сказал Джон фон Нейман, «всякий, кто питает слабость к арифметическим методам получения случайных чисел, грешен вне всяких сомнений».

Любой ПМФПП с ограниченными ресурсами рано или поздно зацикливается — начинает повторять одну и ту же последовательность чисел. Если порождаемая последовательность ПМФПП сходится к слишком

коротким циклам, то такой ПМФПП становится предсказуемым и непригодным для практических приложений. Классификация генераторов ПСП представлена на рисунке 1.2

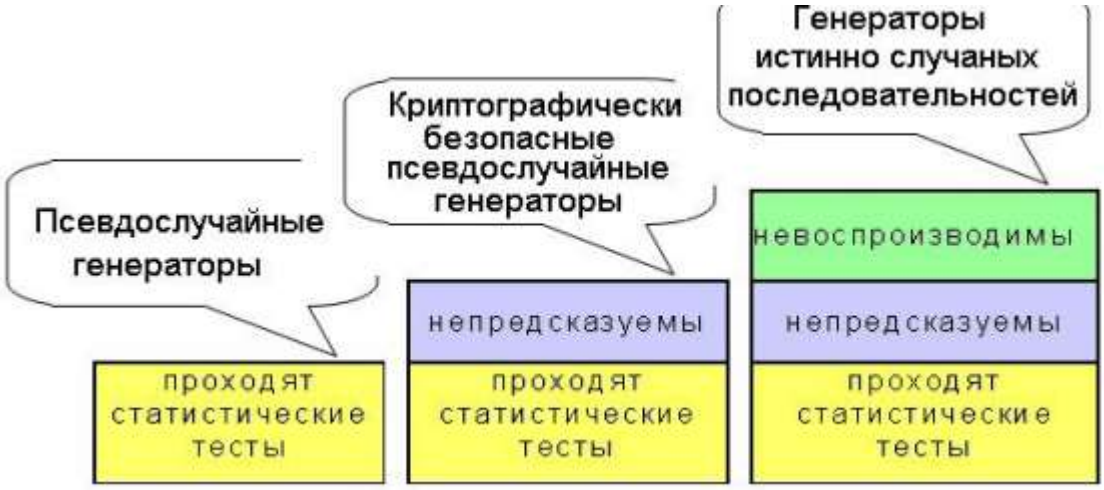


Рисунок 1.2 – Классификация генераторов ПСП

Большинство простых арифметических генераторов хотя и обладают большой скоростью, но страдают от многих серьёзных недостатков:

- Слишком короткий период/периоды.
- Последовательные значения не являются независимыми.
- Некоторые биты «менее случайны», чем другие.
- Неравномерное одномерное распределение.
- Обратимость.

Ниже приведем список генераторов ПСП активно применяемых на данный момент:

1. Алгоритм Блюм — Блюма — Шуба
2. Алгоритм Блюма-Микали
3. Метод Фибоначчи с запаздываниями
4. Линейный конгруэнтный метод
5. Регистр сдвига с линейной обратной связью
6. Метод числа Софи Жермен
7. Вихрь Мерсена

8. Среднеквадратический метод
9. MIXMAX generator
10. Multiply-with-carry
11. Правило 30
12. Алгоритм Вихмана-Хилла
13. Алгоритм сдвига исключающего ИЛИ
14. Xoroshiro128+
15. Алгоритм Ярроу

Каждый алгоритм обладает своими достоинствами и недостатками, в работе предлагается, проанализировать все имеющиеся подходы генерации и выработать свой метод на основе одного из существующих.

## 1.4 Криптостойкие методы формирования псевдослучайных последовательностей

Последовательность называется криптографически надежной псевдослучайной последовательностью (КНПСП), если она непредсказуема, т.е. вычислительно неосуществимо предсказать следующий бит, имея полное знание алгоритма (или аппаратуры) и всех предшествующих битов потока.

Методы формирования криптографически надежных псевдослучайных последовательностей должно быть невоспроизводимыми по выходной последовательности. Таким образом Методы формирования КНПСП должны проходить дополнительные, помимо статистических, тесты. Однако на данный момент, не существует теории которая бы позволила с высокой вероятностью определять степень криптографической надежности метода формирования. Данная задача находится на одном уровне сложности с задачей факторизации целых чисел. Таким образом, чтобы подтвердить сертификация криптографически надежных методов формирования ПСП занимает годы [10].

Некоторые классы криптографически надежных методов формирования ПСП включают следующие:

- Потоковые шифры.
- Блочные шифры, зацикленные в счетчике или в режиме обратной связи.
- МФПСП которые были специально разработаны криптостойкими: CryptGenRandom от Microsoft, алгоритм Ярроу – применяемый в Mac OS X и FreeBSD системах, а также алгоритм Фортуна.
- Комбинации нескольких примитивных МФПСП с целью избавления от обнаруживаемых эффектов детерминированности последовательности.
- Методы формирования, основанные на математически сложных задачах: генератор Микали-Шнора, функция Наора-Рейнгольда и алгоритм Блюма-Шуба, которые обеспечивают надежную защиту. Такие алгоритмы довольно медленны по сравнению с традиционными конструкциями и непрактичны для многих приложений.
- Общие МФПСП на основе односторонних функций. Однако эти методы на практике чрезвычайно медленны и в основном представляют собой теоретический интерес.

Большинство алгоритмов формирования ПСП создают последовательности, которые равномерно распределены по любому из нескольких тестов. Это открытый вопрос и один из центральных в теории и практике криптографии, есть ли способ отличить вывод метода формирования ПСПС от действительно случайной последовательности.

Безопасность большинства криптографических алгоритмов и протоколов с использованием МФПСП основана на предположении, что невозможно определить результаты работы генератора ПСП от по-настоящему случайной последовательности [12]. Простейшими примерами этой зависимости являются потоковые шифры, которые (чаще всего) работают путем эксклюзивного или открытого текста сообщения с выходом МФПСП, создающего зашифрованный текст. Разработка криптографически адекватных МФПСП чрезвычайно сложна, поскольку они должны отвечать

дополнительным критериям. Размер периода является важным фактором криптографической пригодности МФПСП, но не единственным.

Немецкое бюро по Безопасности (Bundesamt für Sicherheit in der Informationstechnik, BSI [58]) разработало четыре критерия для определения качества МФПСП:

K1 – Должна обеспечиваться высокая вероятность того, что числа в последовательности будут отличаться друг от друга.

K2 – Последовательность чисел, генерируемых МФПСП должна быть неотличима от «истинно случайных» чисел в соответствии с заданными статистическими тестами. Тесты - это тест на монобит (равное количество единиц и нулей в последовательности), тест на покере (специальный экземпляр теста хи-квадрат), тест на выполнение (подсчет частоты прогонов различной длины), тест на длинные последовательности (проверяет, существует ли последовательность одного символа длиной 34 или выше в 20000 бит последовательности) и теста на автокорреляцию. По сути, эти требования являются проверкой того, насколько хорошо последовательность случайна: имеет нули и единицы одинаково часто; после последовательности из n нулей (или единиц) следующий бит один (или ноль) появляется с вероятностью 50 %; и любая выбранная подпоследовательность не содержит информации о следующем элементе (элементах) в последовательности.

K3 – Для любого злоумышленника не должно быть возможности вычислить или каким-либо другим образом угадать любую подпоследовательность, любые предыдущие или будущие значения в последовательности или любое внутреннее состояние генератора.

K4 – Для любых практических целей должно быть невозможно, чтобы злоумышленник мог вычислить или угадать из внутреннего состояния метода формирования ПСП любые предыдущие числа в последовательности или любых предыдущих внутренних состояниях МФПСП.

## 1.5 Статистические тесты для проверки генераторов псевдослучайных последовательностей

В криптографии под понятием "случайного числа" понимают число, которое нельзя предсказать до момента его генерации. Однако существует и такое требование – такое число нельзя угадать, зная последующие (так называемая "атака из будущего").

В криптографии оперируют не одиночными числами, а последовательностью, серией. Генератор случайных чисел непрерывно работает, выдавая постоянную последовательность битов, а они уже программно или аппаратно далее интерпретируются в зависимости от контекста. Допустим, у нас есть последовательность случайных чисел $2,5,x$ (для простоты – это десятичные числа). Криптоаналитик хочет узнать случайное число *x*. Говоря о стойкости серии случайных чисел, подразумевают, что:

- нет аналитической зависимости между последовательно сгенерированными числами;
- зная предыдущие числа (в нашем случае – 2 и 5), криптоаналитик не может найти следующие число *x* (атака из прошлого);
- зная последующие числа (в нашем случае – 7), нельзя установить предшествующие (атака из будущего);
- вероятность появления любого числа в последовательности одинаковая.

Поскольку на практике используют генерации двоичной последовательности, то вероятность появления каждого бита - 1/2 в степени *n*, где *n* - разрядность чисел. Пример: имеется последовательность 32-разрядных чисел m, вероятность того, что число $m+1$ будет таким, как предсказал криптоаналитик, равняется $1/2^{32}$ . Если перебирать в секунду 1000 чисел, то это равнозначно одному совпадению за 268 дней круглосуточной работы. Но не всякую последовательность можно назвать случайной. Для исследования

алгоритмов реализации генераторов есть несколько тестов, которые определяют, случайна или нет данная последовательность [61].

Поскольку мы имеем дело с вероятностными процессами, то суждение о случайности или нет такой последовательности также будет верным (неверным) с некоторой вероятностью. На практике для каждого теста есть свое распределение вероятности (своя статистика) и берется какое-то значение, обычно на краях диапазона, например 0,01% (так называемое критическое значение). Далее делают тест и рассчитывают вероятность – если она превышает критическое значение, тестовая последовательность признается неслучайной. Пример: в результате теста вероятность того, что числа случайные, равна 0,40%, значит, числа неслучайные, вероятность превышает критическое значение. При вероятности 0,01 проверяемая последовательность случайна в 99% случаев (если достоверность теста a=0,01, то одна из ста последовательностей будет неслучайной).

Для двоичных последовательностей мы делаем еще некоторые предположения:

Монотонность. Вероятность появления 1 или 0 каждый раз одинаковая – 1/2.

Масштабируемость. Если вся последовательность случайна (проходит тест на случайность), то должна быть случайной и любая случайно выбранная подстрока. Если серия из 1000 чисел случайна, то серия чисел от 500-го до 531- го числа также должна быть случайной и проходить тест.

Национальным институтом стандартов и технологий (NIST) разработаны 16 специальных тестов для определения случайных чисел. Имеется программная реализация в виде NIST Statistical Test Suite для платформы Unix. Она распространяется в виде исходного кода и содержит как инструменты командной строки, так и графические утилиты.

Также есть набор дополнительных данных и утилит, в частности генератор псевдослучайных чисел Блюма-Блюма-Шуба. Образцы случайных

последовательностей плохо или вообще не сжимаются (один из тестов так и называется – тест на сжимаемость).

Последовательность случайных чисел вообще нельзя сжать, так как при попытке построения словаря на таких данных его размер совпадает с размером самих данных – ведь там нет повторяющихся чисел.

Рассмотрим тесты подробнее. Заметим, что не прохождение хоть одного автоматически отменяет все последующие тесты – числа неслучайны, и продолжать проверку нет смысла.

• Частотный тест (монобитный тест на частоту, Frequency Monobits) Test). В этом тесте исследуется доля 0 и 1 в последовательности и насколько она близка к идеальному варианту – равновероятной последовательности. Для теста надо иметь не менее 100 бит данных.

• Блочный тест на частоту (Test for Frequency within a Block). Последовательность разбивается на блоки длиной M бит, и для каждого рассчитывается частота появления единиц и насколько она близка к эталонному значению – M/2. Когда M=1, длина блока 1 бит и тест равнозначен предыдущему. Длина тестовой последовательности не менее 100 бит, длина блока больше 20 бит.

• Тест на серийность (Runs Test). В тесте находятся все серии битов – непрерывные последовательности одинаковых битов – и их распределение сравнивается с ожидаемым распределением таких серий для случайной последовательности. Длина последовательности 100 и более бит.

• Тест на максимальный размер серии единиц. Исследуется длина наибольшей непрерывной последовательности единиц и сравнивается с длиной такой цепочки для случайной последовательности.

• Матрично-ранговый тест (Random Binary Matrix Rank Test). Цель теста – проверка линейной зависимости между подстроками фиксированного размера – матрицами 32x32 бита. Длина последовательности – не менее 38 912 бит, или 38 матриц.

• Спектральный тест (тест дискретным преобразованием Фурье). Цель теста – обнаружить повторяющиеся блоки или последовательности.

• Тест с неперекрывающимися непериодическими шаблонами (Nonoverlapping (Aperiodic) Template Matching Test). Показывает число заранее заданных битовых строк (шаблонов) в последовательности.

• Тест на перекрывающиеся периодические шаблоны (Overlapping (Periodic) Template Matching Test). Показывает количество заранее определенных шаблонов (периодичных битовых последовательностей) в тестовой последовательности.

• Универсальный статистический тест (Maurer's Universal Statistical Test). Показывает число бит между двумя шаблонами и служит для определения сжимаемости последовательности.

• Комплексный тест Lempel-Ziv (Lempel-Ziv Complexity Test). Цель теста – определить число четных слов в последовательности и таким образом определить сжимаемость последовательности. Кроме этих тестов есть еще несколько, но перечислим их без описания:

• Линейный тест (Linear Complexity Test);

• Серийный тест (Serial Test);

• Приближенный тест на энтропию (Approximate Entropy Test);

• Суммирующий тест (Cumulative Sum (Cusum) Test);

• Тест на случайные отклонения (Random Excursions Test и Random Excursions Variant Test).

Рассмотрим еще один набор тестов для анализа свойств псевдослучайных последовательностей DIEHARD.

Тесты DIEHARD — это набор статистических тестов для измерения качества набора случайных чисел. Они были разработаны Джорджем Марсальей в течение нескольких лет и впервые опубликованы на CD-ROM, посвящённом случайным числам. Вместе они рассматриваются как один из наиболее строгих известных наборов тестов. Прохождение всех тестов из

данного блока может считаться достаточным для использования генератора в поточном шифровании.

• Дни рождения (Birthday Spacings) — выбираются случайные точки на большом интервале. Расстояния между точками должны быть асимптотически распределены по Пуассону. Название этот тест получил на основе парадокса дней рождения.

• Пересекающиеся перестановки (Overlapping Permutations) — анализируются последовательности пяти последовательных случайных чисел. 120 возможных перестановок должны получаться со статистически эквивалентной вероятностью.

• Ранги матриц (Ranks of matrices) — выбираются некоторое количество бит из некоторого количества случайных чисел для формирования матрицы над {0,1}, затем определяется ранг матрицы. Считаются ранги.

• Обезьяньи тесты (Monkey Tests) — последовательности некоторого количества бит интерпретируются как слова. Считаются пересекающиеся слова в потоке. Количество «слов», которые не появляются, должны удовлетворять известному распределению. Название этот тест получил на основе теоремы о бесконечном количестве обезьян.

• Подсчёт единичек (Count the 1’s) — считаются единичные биты в каждом из последующих или выбранных байт. Эти счётчики преобразуется в «буквы», и считаются случаи пятибуквенных «слов».

• Тест на парковку (Parking Lot Test) — случайно размещаются единичные окружности в квадрате 100×100. Если окружность пересекает уже существующую, попытаться ещё. После 12 000 попыток, количество успешно «припаркованных» окружностей должно быть нормально распределено.

• Тест на минимальное расстояние (Minimum Distance Test) — 8000 точек случайно размещаются в квадрате 10 000×10 000, затем находится минимальное расстояние между любыми парами. Квадрат этого расстояния должен быть экспоненциально распределён с некоторой медианой.

• Тест случайных сфер (Random Spheres Test) — случайно выбираются 4000 точек в кубе с ребром 1000. В каждой точке помещается сфера, чей радиус является минимальным расстоянием до другой точки. Минимальный объём сферы должен быть экспоненциально распределён с некоторой медианой.

• Тест сжатия (The Squeeze Test) — 231 умножается на случайные вещественные числа в диапазоне [0,1) до тех пор, пока не получится 1. Повторяется 100 000 раз. Количество вещественных чисел необходимых для достижения 1 должно быть распределено определённым образом.

• Тест пересекающихся сумм (Overlapping Sums Test) — генерируется длинная последовательность на [0,1). Добавляются последовательности из 100 последовательных вещественных чисел. Суммы должны быть нормально распределены с характерной медианой и сигмой.

• Тест последовательностей (Runs Test) — генерируется длинная последовательность на [0,1). Подсчитываются восходящие и нисходящие последовательности. Числа должны удовлетворять некоторому распределению.

• Тест игры в кости (The Craps Test) — играется 200 000 игр в кости, подсчитываются победы и количество бросков в каждой игре. Каждое число должно удовлетворять некоторому распределению.

Стоит отметить, что большинство тестов в DIEHARD возвращают значение p, которое должно принадлежать интервалу [0,1], если входной файл содержит действительно независимые случайные биты. Эти p-значения получаются с помощью выражения:

$$p = F(X), \tag{1.1}$$

где $F$ – предполагаемое распределение выборочной случайной величины X, зачастую нормальное.

## 1.6 Задачи исследования

На основе проведенного анализа состояния вопроса относительно методов формирования ПСП были сформулированы следующие задачи:

1) Сформулировать требования, предъявляемые к системе потокового шифрования;

2) Провести анализ существующих алгоритмов формирования ПСП;

3) Провести оценку существующих методов анализа “случайности” методов формирования ПСП;

4) Разработать и исследовать метод формирования ПСП для задач шифрования данных;

5) Провести сравнительные исследования разработанного метода генерации ПСП и стандартных методов генерации ПСП;

6) Разработать рекомендации по использованию генераторов ПСП в задачах шифрования данных.

# ГЛАВА 2 РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДА ФОРМИРОВАНИЯ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ ПОТОЧНОГО ШИФРОВАНИЯ ДАННЫХ

## 2.1 Основные параметры алгоритмов генерации ПСП

Каждый генератор ПСП для применения в задачах шифрования данных должен обладать способностью воспроизвести последовательность имея некое начальное состояние – порождающий элемент (seed). Данное состояние определяет последовательность чисел, воспроизводимую генератором при этом не изменяя алгоритм работы генератора.

Кроме того, генераторы ПСП, имеют определенный период повтора последовательности, который обычно принято указать в соотношение степени двойки. Для примера, в таблице 2.1 приведем параметры некоторых популярных генераторов ПСП, алгоритмы которых реализованы в программном пакете MATLAB.

Таблица 2.1 – Параметры генераторов ПСП в MATLAB

| **Алгоритм** | **Тип генератора** | **Мультипо точность** | **Период повтора последовательности** |
|---|---|---|---|
| mt19937ar | Вихрь Мерсенна | Нет | $2^{19937}{-}1$ |
| dsfmt19937 | Вихрь Мерсенна типа SIMD | Нет | $2^{19937}{-}1$ |
| mcg16807 | Конгруэнтный мультипликативный | Нет | $2^{31}{-}2$ |
| mlfg6331_64 | Мультипликативный метод Фибоначчи с запаздыванием | Да | $2^{124}$ |
| mrg32k3a | Комбинированный многорекурсивный генератор | Да | $2^{127}$ |
| shr3cong | Генератор сдвиговых регистров, модифицированный линейным конгруэнтным генератором | Нет | $2^{64}$ |
| swb2712 | Модифицированный вычитанием с генератором заимствований | Нет | $2^{1492}$ |

Также генераторы обладают мультипоточносьтью, что позволяет использовать для вычисления числовой последовательности мультипоточные вычислительные алгоритмы. Данный параметр является важным при создании криптостойких методов формирования ПСП для шифрования. Кроме того, многие современные системы обладают аппаратной поддержкой мультипоточной обработки, таким образом мультипоточность является важным и актуальным параметром для современного генератора ПСП.

Другим немаловажным параметром является производительность алгоритма, которая определяет количество требуемых ресурсов для расчета последовательности длинной 10 000 000 единиц.

Каждый генератор обладает некоторым набором коэффициентов при расчете последовательности. Свойства генератора можно существенно изменить путем подбора этих коэффициентов.

## 2.2 Примеры реализации алгоритмов ПСП

### 2.2.1 Вихрь Мерсенна

Вихрь Мерсе́нна (Mersenne twister, MT) — генератор псевдослучайных чисел, разработанный в 1997 году японскими учёными Макото Мацумото и Такудзи Нисимура. Вихрь Мерсенна основывается на свойствах простых чисел Мерсенна и обеспечивает быструю генерацию высококачественных по критерию случайности псевдослучайных чисел.

Вихрь Мерсенна лишён многих недостатков, присущих другим ГПСЧ, таких как малый период, предсказуемость, легко выявляемые статистические закономерности. Тем не менее, этот генератор не является криптостойким, что ограничивает его использование в криптографии. Существуют по меньшей мере два общих варианта алгоритма, различающихся только величиной используемого простого числа Мерсенна, наиболее распространённым из которых является алгоритм MT19937.

Вихрь Мерсенна является витковым регистром сдвига с обобщённой отдачей. «Вихрь» — это преобразование, которое обеспечивает равномерное распределение генерируемых псевдослучайных чисел в 623 измерениях (для линейных конгруэнтных генераторов оно ограничено 5 измерениями). Поэтому функция корреляции между двумя последовательностями выборок в выходной последовательности вихря Мерсенна пренебрежимо мала.

Псевдослучайная последовательность, порождаемая вихрем Мерсенна, имеет очень большой период, равный числу Мерсенна, что более чем достаточно для многих практических приложений.

Существуют эффективные реализации Вихря Мерсенна, превосходящие по скорости многие стандартные ГПСЧ (в частности, в 2—3 раза быстрее линейных конгруэнтных генераторов). Алгоритм вихря Мерсенна реализован в программной библиотеке Boost, Glib и стандартных библиотеках для C++, Python, Ruby, R, PHP, MATLAB, Autoit.

Выдаваемые вихрем Мерсенна последовательности псевдослучайных чисел успешно проходят статистические тесты DIEHARD, что подтверждает их хорошие статистические свойства.

Генератор не предназначен для получения криптографически стойких случайных последовательностей чисел. Для обеспечения криптостойкости выходную последовательность генератора необходимо подвергнуть обработке одним из криптографических алгоритмов хеширования.

Вихрь Мерсенна алгоритмически реализуется двумя основными частями: рекурсивной и закалки. Рекурсивная часть представляет собой регистр сдвига с линейной обратной связью, в котором все биты в его слове определяются рекурсивно; поток выходных битов определяются также рекурсивно функцией битов состояния.

Регистр сдвига состоит из 624 элементов, и, в общей сложности, из 19937 клеток. Каждый элемент имеет длину 32 бита за исключением первого элемента, который имеет только 1 бит за счет отбрасывания бита.

Процесс генерации начинается с логического умножения на битовую маску, отбрасывающей 31 бита (кроме наиболее значащих). Следующим шагом выполняется инициализация ($x_0$, $x_1$,…, $x_{623}$) любыми беззнаковыми 32-разрядными целыми числами. Следующие шаги включают в себя объединение и переходные состояния.

Пространство состояний имеет 19937 бит (624·32 — 31). Следующее состояние генерируется сдвигом одного слова вертикально вверх и вставкой нового слова в конец. Новое слово вычисляется гаммированием средней части с исключённой. Выходная последовательность начинается с x624, x625,…

Алгоритм производится в шесть этапов:

**Шаг 0.** Предварительно инициализируется значение констант $u$, $h$, $a$
$u \leftarrow 10\ldots0$ битовая маска старших $w$-$r$ бит,
$h \leftarrow 01\ldots1$ битовая маска младших $r$ бит,
$a \leftarrow a_{w-1}a_{w-2}\ldots a_0$ последняя строка матрицы $A$.

**Шаг 1.** $\boldsymbol{x}[0], \boldsymbol{x}[1],\ldots,\boldsymbol{x}[n-1] \leftarrow$ начальное заполнение

**Шаг 2.** Вычисление $(\boldsymbol{x}_i^u \mid \boldsymbol{x}_{i+1}^l)$
$\boldsymbol{y} \leftarrow (\boldsymbol{x}[i]$ AND $u1)$ OR $(\boldsymbol{x}[(i + 1) \bmod n]$ AND $h1)$

**Шаг 3.** Вычисляется значение следующего элемента последовательности по рекуррентному выражению (1)
$\boldsymbol{x}[i] \leftarrow \boldsymbol{x}[(i + m) \bmod n]$ XOR $(\boldsymbol{y}>>1)$ XOR $a$ если младший бит $\boldsymbol{y} = 1$

Или
$\boldsymbol{x}[i] \leftarrow \boldsymbol{x}[(i + m) \bmod n]$ XOR $(\boldsymbol{y}>>1)$ XOR 0 если младший бит $\boldsymbol{y} = 0$

**Шаг 4.** Вычисление $\boldsymbol{x}[i]T$
$\boldsymbol{y} \leftarrow \boldsymbol{x}[i]$
$\boldsymbol{y} \leftarrow \boldsymbol{y}$ XOR $(\boldsymbol{y}>>u)$
$\boldsymbol{y} \leftarrow \boldsymbol{y}$ XOR $((\boldsymbol{y}<<s)$ AND $\mathbf{b})$
$\boldsymbol{y} \leftarrow \boldsymbol{y}$ XOR $((\boldsymbol{y}<<t)$ AND $\mathbf{c})$
$\boldsymbol{z} \leftarrow \boldsymbol{y}$ XOR $(\boldsymbol{y}>>l)$
вывод $\boldsymbol{z}$

**Шаг 5.** $i \leftarrow (i + 1) \bmod n$

**Шаг 6.** Перейти к шагу 2.

Параметры MT были тщательно подобраны для достижения упомянутых выше свойств. Параметры $n$ и $r$ выбраны так, что

характеристический многочлен примитивный или *nw — r* равна числу Мерсенна 19937. Обратите внимание, что значение w эквивалентно размеру слова компьютера. В этом случае это 32 бита. В то время как значения n, m, r и w фиксируются, значение последней строки матрицы A выбирается случайным образом. Тогда тест на простоту целых намного проще. Так, известно много простых чисел Мерсенна (то есть простых вида *2p−1*), до *p=43112609*. Параметры закалки (tempering) подобраны так, что мы можем получить хорошее равномерное распределение. Все параметры MT приведены в таблице 2.2.

Таблица 2.1 – Коэффициенты генератора Вихря Мерсенна

| n | w | r | m | a | u | s | t | l | b | c |
|---|---|---|---|---|---|---|---|---|---|---|
| 624 | 32 | 31 | 397 | $9908B0DF_{16}$ | 11 | 7 | 15 | 18 | $9D2C5680_{16}$ | $EFC60000_{16}$ |

### 2.2.2 Линейный конгруэнтный

Линейный конгруэнтный метод — один из методов генерации псевдослучайных чисел. Применяется в простых случаях и не обладает криптографической стойкостью. Входит в стандартные библиотеки различных компиляторов.

Линейный конгруэнтный метод был предложен Д. Г. Лемером в 1949 году. Суть метода заключается в вычислении последовательности случайных чисел $X_n$, полагая:

$$X_{n+1} = (aX_{n+1} + c) \bmod m, \qquad (2.1)$$

где $m$ — модуль (натуральное число, относительно которого вычисляет остаток от деления; $m \geq 2$), a — множитель ($0 \leq a < m$), c — приращение ($0 \leq c < m$), $X_0$ — начальное значение ($0 \leq X_0 < m$).

Эта последовательность называется линейной конгруэнтной последовательностью. Например, для $m = 10, X_0 = a = c = 7$ получим последовательность 7,6,9,0,7,6,9,0.

Линейная конгруэнтная последовательность, определенная числами m, *a, c* и $X_0$ периодична с периодом, не превышающим *m*. При этом длина периода равна m тогда и только тогда, когда

1. Числа c и m взаимно простые;
2. $b = a - 1$ кратно *p* для каждого простого *p*, являющегося делителем *m*;
3. *b* кратно 4, если *m* кратно 4.

Метод генерации линейной конгруэнтной последовательности при $c = 0$ называют мультипликативным конгруэнтным методом, а при $c \neq 0$ - смешанным конгруэнтным методом.

При выборе числа m необходимо учитывать следующие условия:

1) число *m* должно быть довольно большим, так как период не может иметь больше m элементов;

2) значение числа m должно быть таким, чтобы $(aX_n + c) \bmod m$ вычислялось быстро.

На практике при реализации метода исходя из указанных условий чаще всего выбирают $m = 2^e$, где e — число битов в машинном слове. При этом стоит учитывать, что младшие двоичные разряды сгенерированных таким образом случайных чисел демонстрируют поведение, далёкое от случайного, поэтому рекомендуется использовать только старшие разряды. Подобная ситуация не возникает, когда $m = \omega \pm 1$, где $\omega$— длина машинного слова. В таком случае младшие разряды $X_n$ ведут себя так же случайно, как и старшие. Выбор множителя *a* и приращения *c* в основном обусловлен необходимостью выполнения условия достижения периода максимальной длины.

Хотя линейный конгруэнтный метод порождает статистически хорошую псевдослучайную последовательность чисел, он не является криптографически стойким. Генераторы на основе линейного конгруэнтного метода являются предсказуемыми, поэтому их нельзя использовать в криптографии. Впервые генераторы на основе линейного конгруэнтного метода были взломаны Джимом Ридсом, а затем Джоан Бояр. Ей удалось

также вскрыть квадратические и кубические генераторы. Другие исследователи расширили идеи Бояр, разработав способы вскрытия любого полиномиального генератора. Таким образом, была доказана бесполезность генераторов на основе конгруэнтных методов для криптографии. Однако генераторы на основе линейного конгруэнтного метода сохраняют свою полезность для некриптографических приложений, например, для моделирования. Они эффективны и в большинстве используемых эмпирических тестах демонстрируют хорошие статистические характеристики.

### 2.2.3 Мультипликативный метод Фибоначчи с запаздыванием

Запаздывающие генераторы Фибоначчи (lagged Fibonacci generator) — генераторы псевдослучайных чисел, также называемые аддитивными генераторами.

В отличие от генераторов, использующих линейный конгруэнтный алгоритм, фибоначчиевы генераторы можно использовать в статистических алгоритмах, требующих высокого разрешения.

В связи с этим линейный конгруэнтный алгоритм постепенно потерял свою популярность и его место заняло семейство фибоначчиевых алгоритмов, которые могут быть рекомендованы для использования в алгоритмах, критичных к качеству случайных чисел.

Аддитивные генераторы генерируют вместо случайных битов случайные слова.

Для работы аддитивному генератору требуется некое начальное состояние, которое является ключом — набор n-битовых слов: 16-битовых, 32-битовых, 64-битовых слов и т. д. $X_1, X_2, .., X_n$.

Зная начальное состояние, можно вычислить *i-е* слово генератора по формуле:

$$X_i = (X_{i-a} + X_{i-b} + ... + X_{i-k}) \bmod (2^m) \qquad (2.2)$$

Один из широко распространённых фибоначчиевых генераторов основан на следующей итерактивной формуле:

$$X_k = \begin{cases} X_{k-a} - X_{k-b} \text{ если } X_{k-a} \geq X_{k-b} \\ X_{k-a} - X_{k-b} + 1, \text{ если } X_{k-a} < X_{k-b} \end{cases} \quad (2.3)$$

где $X_k$ - вещественные числа из диапазона [0,1), a,b – целые положительные числа, являющиеся идентификаторами задержки. При реализации через целые числа достаточно использовать формулу $X_k = X_{k-a} - X_{k-b}$ (при этом будут происходить арифметические переполнения). Для работы фибоначчиеву генератору требуется знать $\max\{a,b\}$ предыдущих сгенированных случайных чисел, которые могут быть сгенирированы обычным конгруэнтным методом.

Получаемые случайные числа обладают хорошими статистическими свойствами, причем все биты случайного числа равнозначны по статистическим свойствам. Период фибоначчиева генератора может быть оценен по следующей формуле:

$$T = (2^{\max\{a,b\}} - 1) \cdot 2^l \quad (2.4)$$

где $l$ – число битов в мантиссе вещественного числа.

## 2.3 Разработка метода формирования ПСП

В основе разрабатываемого метода формирования ПСП лежит последовательность, полученная с помощью мультиплиактивного конгруэнтного генератора. Последовательность разбивается на $k$ векторов, длины которых являются простыми числами. Причем требуется брать неповторяющееся просты числа, т.е. уникальные. Таким образом, найдя наименьшее общее кратное их длин $N_0$, что будет равно их произведению. Мы получаем период повторения последовательности. Каждый из векторов циклически продлевается до наименьшего общего кратного $N_0$, таким образом полученные $k$ векторов длиной $N_0$ складываем по модулю 2, получая искомую

последовательность. Чтобы улучшить эффективность метода можно увеличить мерность векторов *k.* В двумерном случае это будет означать, что мы разбиваем последовательность на квадратные матрицы длины сторон которых являются простыми числами. Затем каждую из матриц циклически расширяем до матрицы размерностью $N_0$x$N_0$. Итоговую последовательность получаем сложением получившихся матриц по модулю два. Кроме того, можно вывести формулу для расчёта *i-го* элемента генерируемой последовательности длительностью *N* элементов:

$$Z_i = B1\left(x_1, y_1\right) \oplus B2\left(x_1, y_1\right) \oplus ... \oplus B_k\left(x_j, y_j\right), \quad (2.5)$$

где $i = 1...N$, $j = 1...k$, *B1* *u* *B2* – сформированные последовательности мультипликативным конгруэнтным методом, *l* – вектор размерностей матриц.

$$x_j = \left(\left(i-1\right) \bmod l_j\right) + 1 \quad (2.6)$$

$$y_j = \left(\left(\left\lceil \frac{i}{No} \right\rceil - 1\right) \bmod l_j\right) + 1 \quad (2.7)$$

Таким образом, эта последовательность модифицируется до криптостойкого уровня путем применения принципа жизненного цикла цикад.

Жизненный цикл цикады:

В зависимости от вида, каждые 7, 11, 13 или 17 лет периодические цикады одновременно массово вылезают на свет и превращаются в шумных летающих тварей, спариваются и вскоре умирают.

Использование простых чисел позволяет увеличить период повтора генерируемой последовательности. В одномерном случае период повтора последовательности будет равняться наименьшему общему кратному. Для простых чисел наименьшее общее кратное будет равняться их произведению.

Исследования показали, что численность животных, которые питаются цикадами — обычно птицы, пауки, осы, рыбы и змеи — часто демонстрируют более короткий цикл 2–6 лет между пиком и спадом популяции. Таким образом, если бы цикады появлялись, например, каждые 12 лет, то каждый

хищник с жизненным циклом 2, 3, 4 или 6 лет мог бы синхронизировать циклы подъёма своей численности с регулярным появлением цикад.

С другой стороны, если выводок цикад был настолько неудачлив, чтобы появиться во время трёхлетнего пика численности ос, то следующий раз это случится только через 51 год. В промежуточные поколения цикады могут спокойно восстановить своё население и намного превысить число хищников.

Блок-схема алгоритма предлагаемого метода генерации ПСП представлена на рисунке 2.1

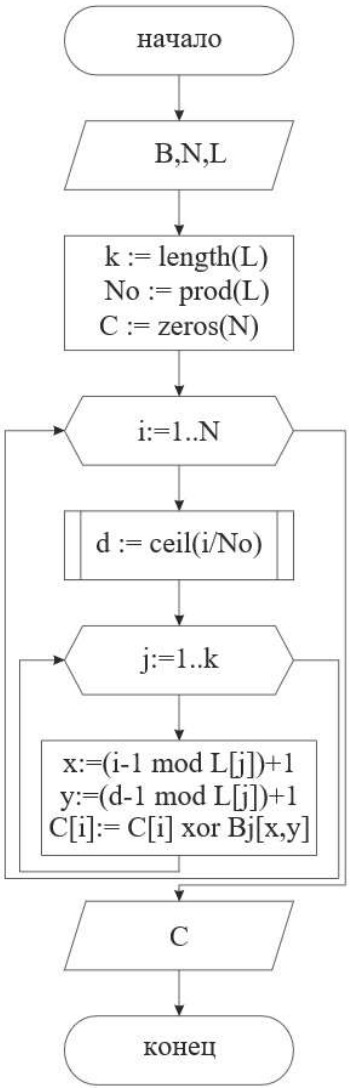


Рисунок 2.1 – Блок схема алгоритма формирования ПСП

Данный алгоритм реализован в программном пакете MATLAB и его программный код представлен в листинге 2.1.

Листинг 2.1

```
Ns = [17 21 53]; %Базовые последовательности
No = prod(Ns); %Расчёт наименьшего общего кратного
Nn = length(Ns); %Определение длины
%% генерируем слои 1 вариант
s = struct('x',[]);% Начальные состояния генератора ПСП
myStream=RandStream('mcg16807','Seed',1); %Выбор формирования базового алгоритма
for k=1:Nn % Формирование слоев
   s(k).x = randi(myStream,2,Ns(k),Ns(k))>1; % Случайный бинарный квадрат
end
%% поштучно
 N = No^2;
 c = boolean(zeros(1,N));
 d = 1;
 for i=1:N
    d = ceil(i/No); % строка
    b = 0;
    for j=1:length(Ns)
      xi = mod(i-1,Ns(j))+1;
      yi = mod(d-1,Ns(j))+1;
      b = xor(b,s(j).x(xi,yi));
    end
    c(i) = b;
 end
n = ceil(length(c)/32); % Перевод чисел в формат double
if length(c)<n*32
   c((end+1):(n*32))=false;
end
sig=zeros(1,n);
for i=1:n
   sig(i)=sum(bitset(sig(i),find(c((1:32)+(i-1)*32))));
end
%Выгрузка результатов работы алгоритма формирования ПСП
          fid=fopen('cida.txt','w');
          n=10000000;
          fprintf(fid,'# standart matlab \n');
          fprintf(fid,'type: d\n');
          fprintf(fid,'count: %d\n',n);
          fprintf(fid,'numbit: %d\n',32);
          for k=1:n
             fprintf(fid,'%d \n',sig(k));
          end
fclose(fid);
```

Следует отметить, что для тестирования последовательность количество бит в выгрузке должно превышать 10 миллионов единиц.

На рисунке 2.2 представлены иллюстрации формирования псевдослучайной последовательности.

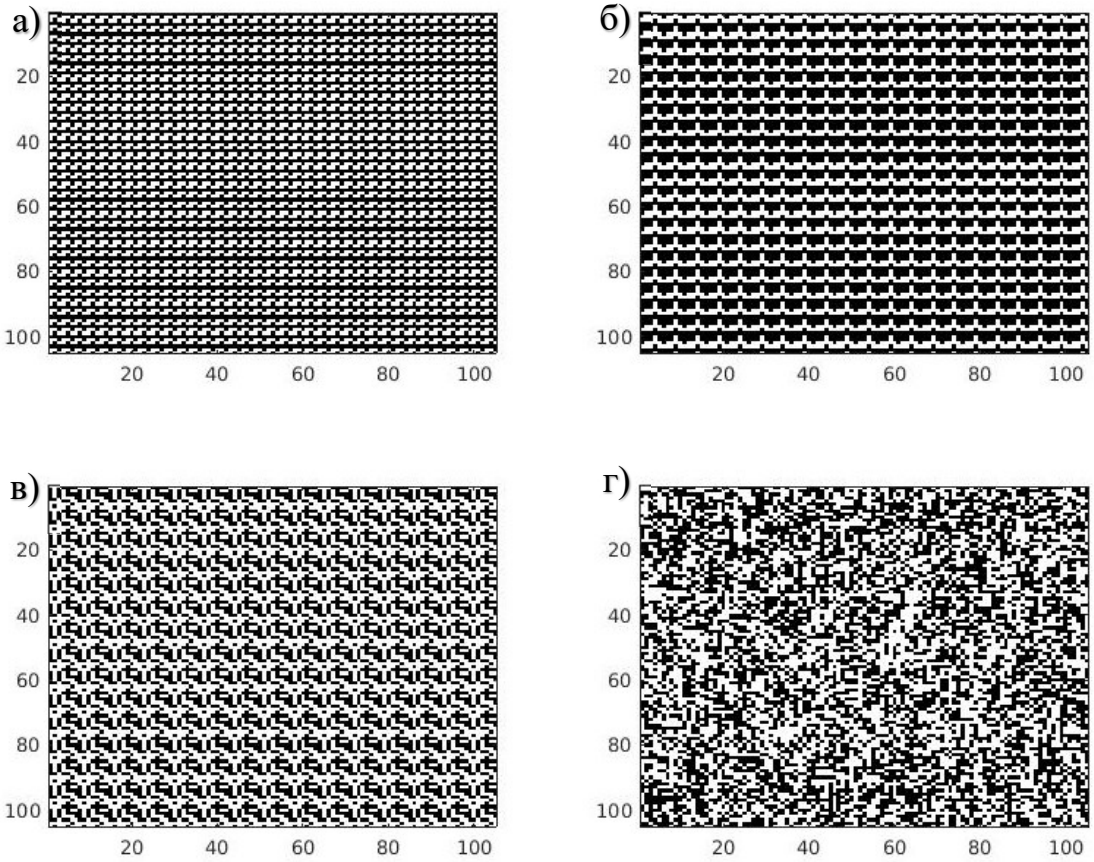


Рисунок 2.2 – Слои данных используемые для формирования ПСП: а) первый слой – последовательность 3, б) второй слой – последовательность – 5, в) третий слой – последовательность 7, г) сформированная псевдослучайная последовательность длительностью $N=105^2$ *бит*

Исходя из представленных на рисунке 2.2 изображений, можно отметить, что в изображениях: а), б), в) отчетливо заметен повторяющийся паттерн. Паттерн представляет собой восприятие некоторой закономерности. При этом на рисунке 2.2 г) паттерн визуально не наблюдается, что говорит о качественно сформированной последовательности имеющей распределение плотности вероятности, приближенное к нормальному.

Бинарные векторы псевдослучайных последовательностей длительностью $N=105^2$ *бит,* можно сгруппировать в 32 битные числа, последовательность которых представлена на рисунке 2.3.

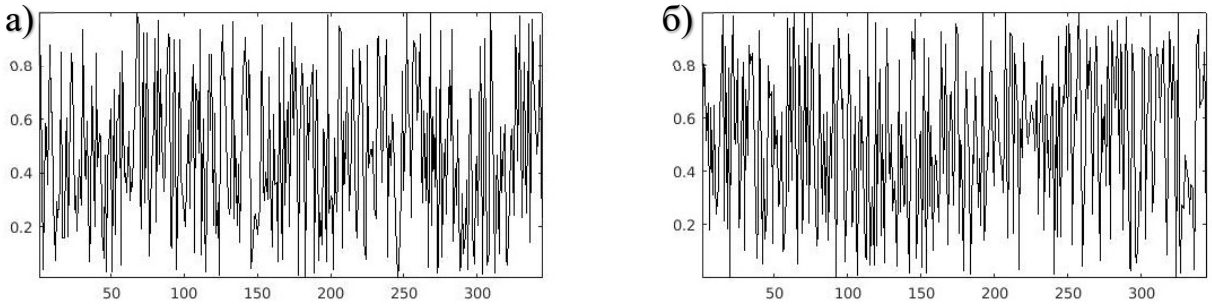


Рисунок 2.3 – Последовательность 32 битных чисел сформированных: а) мультипликативным конгруэнтным методом, б) разработанным методом формирования ПСП

Чем более распределение случайной величины в сформированной последовательности ближе к равномерному распределению, тем качественнее, с точки зрения равновероятности появления значений, рассматриваемый генератор. Можно предположить, что генераторы ПСП формирующие последовательность, распределение которой ближе к равномерному, тем лучше он подходит для задач шифрования. На рисунке 2.4 представлены гистограммы распределения случайных величин сформированных последовательностей – полученной с помощью мультипликативного конгруэнтного метода и разработанного метода формирования ПСП.

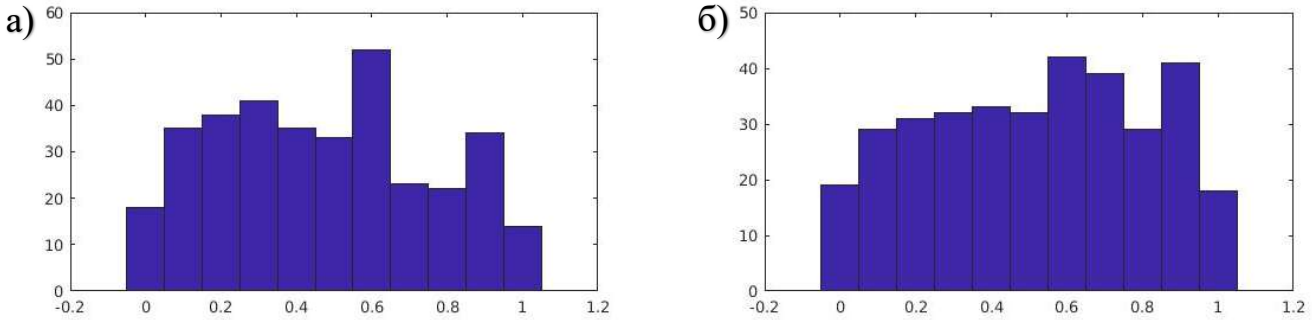


Рисунок 2.4 – Гистограммы распределения случайных величин:
а) гистограмма для мультипликативного конгруэнтного метода,
б) гистограмма для разработанного метода формирования ПСП

Следует отметить, что гистограмма распределения случайной величины для разработанного метода является в большей степени равномерной, нежели

гистограмма стандартного метода, что говорит в пользу разработанного в работе метода формирования ПСП.

Для повышения периода повтора псевдослучайной последовательности, используя разработанный генератор можно дать следующие рекомендации:

1) Использовать в качестве порождающей последовательности естественно порожденные элементы – начальные состояния, т.е. состояния, полученные с помощью физических генераторов случайных чисел: шорох листвы, льющаяся вода, белый шум и т.п.

2) Использование больших простых чисел – позволяет улучшить параметры случайности при прохождении тестов;

3) Использование многомерного сложения слоев случайных последовательностей – позволяет заметно улучшить качество псевдослучайной последовательности.

Основные параметры разработанного алгоритма генерации псевдослучайной последовательности представим в таблице 2.2.

Таблица 2.2 – Параметры разработанного алгоритма генерации ПСП

| **Базовый алгоритм формирования порождающей последовательности** | **Тип разработанного алгоритма генерации ПСП** | **Мультипо точность** | **Период повтора последовательности** |
|---|---|---|---|
| Конгруэнтный мультипликативный | Одномерное формирование ПСП | Да | $2^{152}-1$ |
| Вихрь Мерсенна | Одномерное формирование ПСП | Да | Не установлен* |
| Физический генератор | Одномерное формирование ПСП | Да | Не установлен* |
| Конгруэнтный мультипликативный | Двумерное формирование ПСП | Да | $2^{202}-1$ |
| Вихрь Мерсенна | Двумерное формирование ПСП | Да | Не установлен* |
| Физический генератор | Двумерное формирование ПСП | Да | Не установлен* |

* - **не установлен**, означает что не удалось рассчитать последовательность достаточной длины из-за ограничений вычислительной системы.

## 2.4 Исследование разработанного метода формирования ПСП с помощью статистических тестов

Для тестирования разработанного метода формирования ПСП на пригодность его использования в задачах поточного шифрования воспользуемся пакетом тестов DIEHARD. Данный пакет тестов доступен на операционной системе Linux. В качестве входной последовательность для работы данных тестов требуется подать на вход бинарную последовательность длительностью не менее 10 миллионов единиц.

В качестве аппаратной платформы для тестирования был выбран IBM PC с набором инструкций x86-64 Intel i7 Generation 7700, DDR IV 16 GB. Тестирование последовательностей осуществлялось в мультипоточном режиме, на нескольких физических ядрах, что увеличило быстродействие. Время обработки одной последовательности набором тестов составило примерно 17 минут.

В таблице 2.3 представлены результаты тестов DIEHARD для разработанного метода формирования ПСП. В качестве порождающей последовательности использовалась последовательность, сформированная конгруэнтным мультипликативным генератором.

Таблица 2.3 – Результаты DIEHARD тестов для разработанного метода формирования ПСП, в качестве порождающей последовательности использовались ПСП полученные конгруэнтным мультипликативным генератором

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.57259041 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.00090290 | WEAK |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.00019471 | WEAK |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.17572828 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.49083486 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.29539990 | PASSED |

Окончание таблицы 2.3

| **Имя теста** | **ntup** | **tsamples** | **psamples** | **p-value** | **Решение** |
|---|---|---|---|---|---|
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.38740803 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.27195005 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.89786475 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.74919577 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.00097822 | WEAK |
| diehard_runs | 0 | 100000 | 100 | 0.91782606 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.43260978 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.00000025 | FAILED |
| diehard_craps | 0 | 200000 | 100 | 0.02744465 | PASSED |
| sts_monobit | 1 | 100000 | 100 | 0.93264614 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.93264614 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.53469320 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.92652932 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.41619006 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.66443179 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.75127942 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.94513080 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.97650606 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.05887055 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65817225 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.01647396 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.29230000 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.08114023 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.82040100 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.76855501 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.10920245 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.96181498 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.98605667 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.44788938 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.47533385 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.81458602 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.19465569 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.99093570 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.68157223 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.79877959 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.43100417 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.52389604 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.48644643 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.06711507 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.42780953 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.44788938 | PASSED |

Исходя из данных представленных в данной таблице, можно констатировать, что разработанный метод формирования ПСП является не

пригодным для использования в задачах поточного шифрования. Из 47 тестов 3 теста имеют слабые результаты, 1 тест (Тест на игру в кости) был провален.

Для сравнения представим результаты ПСП сформированной конгруэнтный мультипликативным методом – таблица 2.4.

Таблица 2.4 – Результаты DIEHARD тестов ПСП сформированной конгруэнтным мультипликативным генератором

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.52287411 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.00000866 | WEAK |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.00819849 | PASSED |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.63700469 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.38931676 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.56302152 | PASSED |
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.13201501 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.55915630 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.24882615 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.18703863 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.02987542 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.62390713 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.27473482 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.00000015 | FAILED |
| diehard_craps | 0 | 200000 | 100 | 0.26096229 | PASSED |
| sts_monobit | 1 | 100000 | 100 | 0.79243265 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.79243265 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.00016990 | WEAK |
| sts_serial | 3 | 100000 | 100 | 0.26341393 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.28716090 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.07913765 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.00279340 | WEAK |
| sts_serial | 5 | 100000 | 100 | 0.38312188 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.92693578 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.00069249 | WEAK |
| sts_serial | 6 | 100000 | 100 | 0.00000313 | WEAK |
| sts_serial | 7 | 100000 | 100 | 0.00004782 | WEAK |
| sts_serial | 7 | 100000 | 100 | 0.07046148 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.68472511 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.01438339 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.16990005 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.02968913 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.03419011 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.00384100 | WEAK |

Окончание таблицы 2.4

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| sts_serial | 11 | 100000 | 100 | 0.71160614 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.59510804 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.00065348 | WEAK |
| sts_serial | 12 | 100000 | 100 | 0.00095155 | WEAK |
| sts_serial | 13 | 100000 | 100 | 0.00639349 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.08060782 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.11054062 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.40220245 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.92115917 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.13975407 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.67645585 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.36042203 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.00000000 | FAILED |

Используя в качестве порождающей последовательности для разработанного метода формирования ПСП последовательность, полученную с помощью Вихря Мерсенна сформируем ПСП и выполним ее тестирование – результаты в таблице 2.5

Таблица 2.5 – Результаты DIEHARD тестов для разработанного метода формирования ПСП, в качестве порождающей последовательности использовались ПСП полученные с помощью Вихря Мерсенна

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.57259041 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.00090290 | WEAK |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.00019471 | WEAK |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.17572828 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.49083486 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.29539990 | PASSED |
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.38740803 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.27195005 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.89786475 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.74919577 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.74397822 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.91782606 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.43260978 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.65817225 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.02744465 | PASSED |

Окончание таблицы 2.5

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| sts_monobit | 1 | 100000 | 100 | 0.93264614 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.93264614 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.53469320 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.92652932 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.41619006 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.66443179 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.75127942 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.94513080 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.97650606 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.05887055 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65817225 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.01647396 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.29230000 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.08114023 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.82040100 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.76855501 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.10920245 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.96181498 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.98605667 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.44788938 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.47533385 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.81458602 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.19465569 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.99093570 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.68157223 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.79877959 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.43100417 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.52389604 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.48644643 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.06711507 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.42780953 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.44788938 | PASSED |

Исходя из данных представленных в данной таблице, можно констатировать, что разработанный метод формирования ПСП является пригодным для использования в задачах поточного шифрования, в случае использования Вихря Мерсенна для получения порождающей последовательности. Из 47 тестов 2 теста имеют слабые результаты. Для сравнения представим результаты ПСП сформированной с помощь Вихря Мерсенна – таблица 2.6.

Таблица 2.6 – Результаты DIEHARD тестов ПСП сформированной с помощью Вихря Мерсенна

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.52287411 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.00000866 | WEAK |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.00819849 | PASSED |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.53700469 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.48931676 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.46302152 | PASSED |
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.23201501 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.45915630 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.34882615 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.28703863 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.02987542 | WEAK |
| diehard_runs | 0 | 100000 | 100 | 0.62390713 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.27473482 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.00006229 | FAILED |
| diehard_craps | 0 | 200000 | 100 | 0.26096229 | PASSED |
| sts_monobit | 1 | 100000 | 100 | 0.69243265 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.83264614 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.53469320 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.72652932 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.41619006 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.00000079 | WEAK |
| sts_serial | 4 | 100000 | 100 | 0.65127942 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.84513080 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.87650606 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.05887055 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65817225 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.01647396 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.29230000 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.08114023 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.92040100 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.76855501 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.10920245 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.86181498 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.78605667 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.44788938 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.47533385 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.81458602 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.19465569 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.89093570 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.68157223 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.79877959 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.43100417 | PASSED |

Окончание таблицы 2.6

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| sts_serial | 15 | 100000 | 100 | 0.7389604 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.68644643 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.56711507 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.42780953 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.54788938 | PASSED |

Далее приведем результаты DIEHARD тестов, полученные с помощью разработанного метода формирования ПСП на базе последовательности, сформированной с помощью физического источника – таблица 2.7. В качестве такого источника использовались данные полученные измерения атмосферного шума. Атмосферное шум - это радиошум, вызванный природными атмосферными процессами, прежде всего ударами молний во время гроз. Во всем мире происходит около 40 ударов молний в секунду - ≈3,5 миллиона в день. Таким образом, данный метод является достаточно доступным для получения данных.

Таблица 2.7 – Результаты DIEHARD тестов для разработанного метода формирования ПСП, в качестве порождающей последовательности использовались случайные числа, полученные от физического источника

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.57259041 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.91782606 | PASSED |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.91782606 | PASSED |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.17572828 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.49083486 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.29539990 | PASSED |
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.38740803 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.27195005 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.89786475 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.74919577 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.74397822 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.91782606 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.43260978 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.65817225 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.02744465 | PASSED |

Окончание таблицы 2.7

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| sts_monobit | 1 | 100000 | 100 | 0.69243265 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.83264614 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.53469320 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.72652932 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.41619006 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.72652932 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.65127942 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.84513080 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.87650606 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.05887055 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65817225 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.01647396 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.29230000 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.08114023 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.92040100 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.76855501 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.10920245 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.86181498 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.78605667 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.44788938 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.47533385 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.81458602 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.19465569 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.89093570 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.68157223 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.79877959 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.43100417 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.7389604 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.68644643 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.56711507 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.42780953 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.00788938 | WEAK |

Исходя из данных представленных в таблице 2.7, можно констатировать, что разработанный метод формирования ПСП является пригодным для использования в задачах поточного шифрования, при использовании физического генератора для получения порождающей последовательности Из 47 тестов 1 тест имеет слабые результаты.

Для сравнения, в таблице 2.8 приведем результаты тестов случайных чисел полученных с помощью измерения атмосферного шума

(последовательность была взята из открытых источников – Карты замера частоты молний https://sos.noaa.gov/datasets/lightning-flash-rate/)

Таблица 2.8 – Результаты DIEHARD тестов для случайной последовательности, полученной от физического источника

| **Имя теста** | **ntup** | **tsamples** | **psamples** | **p-value** | **Решение** |
|---|---|---|---|---|---|
| diehard_birthdays | 0 | 100 | 100 | 0.57259041 | PASSED |
| diehard_operm5 | 0 | 1000000 | 100 | 0.91782606 | PASSED |
| diehard_rank_32x32 | 0 | 40000 | 100 | 0.91782606 | PASSED |
| diehard_rank_6x8 | 0 | 100000 | 100 | 0.87572828 | PASSED |
| diehard_bitstream | 0 | 2097152 | 100 | 0.69083486 | PASSED |
| diehard_count_1s_str | 0 | 256000 | 100 | 0.29532990 | PASSED |
| diehard_count_1s_byt | 0 | 256000 | 100 | 0.38740803 | PASSED |
| diehard_parking_lot | 0 | 12000 | 100 | 0.47195005 | PASSED |
| diehard_2dsphere | 2 | 8000 | 100 | 0.89786475 | PASSED |
| diehard_3dsphere | 3 | 4000 | 100 | 0.54919577 | PASSED |
| diehard_squeeze | 0 | 100000 | 100 | 0.64397822 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.81782606 | PASSED |
| diehard_runs | 0 | 100000 | 100 | 0.93260978 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.55817225 | PASSED |
| diehard_craps | 0 | 200000 | 100 | 0.32744465 | PASSED |
| sts_monobit | 1 | 100000 | 100 | 0.29243265 | PASSED |
| sts_serial | 1 | 100000 | 100 | 0.33264614 | PASSED |
| sts_serial | 2 | 100000 | 100 | 0.23469320 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.32652932 | PASSED |
| sts_serial | 3 | 100000 | 100 | 0.21619006 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.32652932 | PASSED |
| sts_serial | 4 | 100000 | 100 | 0.45127942 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.54513080 | PASSED |
| sts_serial | 5 | 100000 | 100 | 0.87650606 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65887055 | PASSED |
| sts_serial | 6 | 100000 | 100 | 0.65817225 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.71647396 | PASSED |
| sts_serial | 7 | 100000 | 100 | 0.69230000 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.58114023 | PASSED |
| sts_serial | 8 | 100000 | 100 | 0.62040100 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.56855501 | PASSED |
| sts_serial | 9 | 100000 | 100 | 0.00920245 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.436181498 | PASSED |
| sts_serial | 10 | 100000 | 100 | 0.43605667 | PASSED |

Окончание таблицы 2.8

| Имя теста | ntup | tsamples | psamples | p-value | Решение |
|---|---|---|---|---|---|
| sts_serial | 11 | 100000 | 100 | 0.44432938 | PASSED |
| sts_serial | 11 | 100000 | 100 | 0.43353385 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.81448602 | PASSED |
| sts_serial | 12 | 100000 | 100 | 0.29435569 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.4093570 | PASSED |
| sts_serial | 13 | 100000 | 100 | 0.48157223 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.39527959 | PASSED |
| sts_serial | 14 | 100000 | 100 | 0.43140417 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.73823604 | PASSED |
| sts_serial | 15 | 100000 | 100 | 0.63643643 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.52411507 | PASSED |
| sts_serial | 16 | 100000 | 100 | 0.44780953 | PASSED |
| dab_bytedistrib | 0 | 51200000 | 1 | 0.00030338 | WEAK |

Результаты данных экспериментов сведем в таблицу 2.9.

Таблица 2.9 – Результаты статистической оценки различных методов формирования ПСП

| **Базовый алгоритм формирования порождающей последовательности** | **Тип разработанного алгоритма генерации ПСП** | **Тестов пройдено** | **Тестов провалено** | **Тесты пройдены частично** |
|---|---|---|---|---|
| Нет | Конгруэнтный мультипликативный | 45 | 2 | 10 |
| Нет | Вихрь Мерсенна | 46 | 1 | 3 |
| Нет | Физический генератор | 47 | 0 | 1 |
| Конгруэнтный мультипликативный | Разработанный метод формирования | 46 | 1 | 3 |
| Вихрь Мерсенна | Разработанный метод формирования | 47 | 0 | 2 |
| Физический генератор | Разработанный метод формирования | 47 | 0 | 1 |

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы магистра были исследованы особенности существующих подходов к формированию псевдослучайных последовательностей для поточного шифрования данных. На основе данного исследования были сделаны выводы, которые позволили модифицировать существующий мультипликативный конгруэнтный метод формирования для использования его в задачах поточного шифрования данных. Сформированы требования, предъявляемые к псевдослучайным последовательностям, применяемым в потоковых системах шифрования. Проведен анализ существующих алгоритмов формирования ПСП. Даны примеры методик оценки случайности псевдослучайных последовательностей и проведен анализ некоторых известных методов формирования ПСП наряду с разработанным методом.

Разработан и исследован метод формирования псевдослучайной последовательности, который обладает свойствами, которые позволяют использовать генерируемые последовательности в поточном шифровании данных. Проведены сравнительные исследования ПСП сформированных следующими методами: Вихрь Мерсенна, Мультипликативный конгруэнтный метод, Разработанный метод формирования ПСП.

На основании данных исследований можно сделать вывод, что для задач поточного шифрования пригодны: вихрь Мерсенна (ограничено, т.к. не отвечает всем параметрам крипто-стойкости) и разработанный метод формирования ПСП с использованием в качестве порождающей последовательность случайные числа от физического генератора, либо полученные с помощью Вихря Мерсенна.

Разработаны рекомендации по выбору начальных параметров для разработанного метода формирования ПСП которые позволяют увеличить период повтора последовательности и улучшить случайность ее элементов за счет устранения паттернов.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ


1. ГОСТ 28147-89. Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
2. Анин Б.Ю. Защита компьютерной информации. - СПб.: БХВ — Санкт-Петербург, 2000. - 384 с.
3. Брассар Ж. Современная криптология: Пер. с англ. — М.: ПОЛИМЕД, 1999.176 е., с илл.
4. Бурдаев О.В., Иванов М.А., Тетерин И.И. Ассемблер в задачах защиты информации / Под ред. И.Ю. Жукова. М.: Кудиц-образ, 2002. – 320с.
5. Бухштаб А.А. Теория чисел. — М.: Учпедгиз, 1960. 376 с.
6. Варфоломеев А.А., Жуков А.Е., Пудовкина М.А. Поточные криптосистемы. Основные свойства и методы анализа стойкости. Учебное пособие. М.: ПАИМС, 2000.-272 с.
7. Введение в криптографию / Под общ. ред. В.В. Ященко. — М.: МЦНМО: «ЧеРо», 1999.-272 с.
8. Гантмахер Ф. Р. Теория матриц. М.: Наука, 1988. - 552 с.
9. Генератор цифровых последовательностей: А.С. 1513449 СССР / М.А. Иванов И Открытия, изобретения. 1989. № 37.
10. Генераторы псевдослучайных кодов в задачах криптографической защиты информации / М.А. Иванов, Т.В. Левчук, И.В. Чугунков и др. // Безопасность информационных технологий. 1998. - № 2. - С. 91-93.
11. Герасименко В.А., Малюк А.А. Основы защиты информации. М.: МИФИ, 1997.-538 с.
12. Гилл А. Линейные последовательностные машины: Пер. с англ. — М.: Наука, 1974. 288 с.
13. Гмурман В.Е. Теория вероятностей и математическая статистика: Учебное пособие для вузов. М.: Высшая школа, 1972. — 368 с. с илл.
14. Гутер Р.С., Овчинский Б.В. Элементы численного анализа и математической обработки результатов опыта. М.: Наука, 1970. - 432 с.

15. Дисман А.М., Иванов А.А., Иванов М.А. Принципы проектирования и свойства генераторов L-ричных последовательностей максимальной длины // Автоматика и вычислительная техника. 1990. № 4. - С. 55-73.

16. Доценко В.И., Фараджев Р.Г. Анализ и свойства последовательностей максимальной длины // Автоматика и телемеханика. 1969. № 11. — С. 119-127.

17. Доценко В.И., Фараджев Р.Г., Чхартищвили Г.С. Свойства последовательностей максимальной длины с Р-уровнями // Автоматика и телемеханика. — 1971. №8. -С. 189-194.

18. Зензин О.С. Иванов М.А. Стандарт криптографической защиты — AES. Конечные полей / под ред. М.А. Иванова. М.: КУДИЦ-ОБРАЗ, 2002. - 176 с.

19. 26. Жуков И.Ю., Иванов М.А., Осмоловский С.А. Принципы построения криптостойких генераторов псевдослучайных кодов // Проблемы информационной безопасности. Компьютерные системы. 2001. № 1. — С. 55-65.

20. 27. Жуков И.Ю., Иванов М.А., Чугунков И.В. Генераторы псевдослучайных кодов для решения задач поточного шифрования // Научная сессия МИФИ-2001. Сборник научных трудов. В 14 т. М.: МИФИ, 2001. - Т. 12. - С. 6667.

21. Иванов М.А. Алгоритмы нелинейного преобразования данных для криптографических примитивов хеширования, генерации ПСП и поточного шифрования. Научная сессия МИФИ. Сборник научных трудов. -М.: 2003.

22. Иванов М.А. Контроль хода программ и микропрограмм с использованием сигнатурного анализа // Автоматика и вычислительная техника. — 1990. № 4. -С. 90-94.

23. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях. М.: КУДИЦ-ОБРАЗ, 2001. - 368 с.

24. Иванов М.А. Многоканальные анализаторы сигнатур // Автоматика и вычислительная техника. 1990. № 2. - С. 84-92.

25. Иванов М.А. Принципы построения и свойства недвоичных генераторов псевдослучайных кодов // Безопасность информационных технологий. — 1998. №2.-С. 94-96.

26. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. М.: КУДИЦ-ОБРАЗ, 2003. 312 с. с илл.

27. Кнут Д. Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд.: Пер. с англ.: Уч. пос. — М.: Издательский дом «Вильяме», 2000. — 832 с. с ил.

28. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров: Пер. с англ. / под ред. И.Г. Арамановича. М.: Наука, 1973. — 832 с. с илл.

29. Криптография в банковском деле / М.И. Анохин, Н.П. Варновский, В.М. Сидельников, В.В. Ященко. М.: МИФИ, 1997. - 274 с.

30. Мак-Уильяме Ф.Дж., Слоан Н.Дж.А. Псевдослучайные последовательности и таблицы // ТИИЭР. 1976. №12. - С. 80-95.

31. Мельников В.В. Защита информации в компьютерных системах. М.: Финансы и статистика, 1997. — 368 с. с илл.

32. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. — М.: Радио и связь, 1999.-328 с.

33. Чугунков И.В. Оценка качества криптоалгоритмов и алгоритмов генерации псевдослучайных последовательностей // Компьютерные системы и технологии: Лабораторный практикум / под ред. Л. Д. Забродина. М.: Диалог-Мифи, 2001.-336 с.

34. Чугунков И.В. Система оценки качества генераторов псевдослучайных кодов // Научная сессия МИФИ-2000. Сборник научных трудов. В 13 т. — М.: МИФИ, 2000. Т. 11, С. 45-46.

35. Liddell, Henry George; Scott, Robert; Jones, Henry Stuart; McKenzie, Roderick (1984). A Greek-English Lexicon. Oxford University Press.

36. Rivest, Ronald L. (1990). "Cryptography". In J. Van Leeuwen. Handbook of Theoretical Computer Science. 1. Elsevier.

37. Bellare, Mihir; Rogaway, Phillip (21 September 2005). "Introduction". Introduction to Modern Cryptography. p. 10.

38. Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A. Handbook of Applied Cryptography. ISBN 0-8493-8523-7. Archived from the original on 7 March 2005.

39. Biggs, Norman (2008). Codes: An introduction to Information Communication and Cryptography. Springer. p. 171.

40. "Overview per country". Crypto Law Survey. February 2013. Retrieved 26 March 2015.

41. "UK Data Encryption Disclosure Law Takes Effect". PC World. 1 October 2007. Retrieved 26 March 2015.

42. Ranger, Steve (24 March 2015). "The undercover war on your internet secrets: How online surveillance cracked our trust in the web". TechRepublic. Archived from the original on 2016-06-12. Retrieved 2016-06-12.

43. Doctorow, Cory (2 May 2007). "Digg users revolt over AACS key". Boing Boing. Retrieved 26 March 2015.

44. Kahn, David (1967). The Codebreakers. ISBN 0-684-83130-9.

45. Sharbaf, M.S. (2011-11-01). "Quantum cryptography: An emerging technology in network security". 2011 IEEE International Conference on Technologies for Homeland Security (HST): 13–19. doi:10.1109/THS.2011.6107841.

46. Oded Goldreich, Foundations of Cryptography, Volume 1: Basic Tools, Cambridge University Press, 2001, ISBN 0-521-79172-3

47. "Cryptology (definition)". Merriam-Webster's Collegiate Dictionary (11th ed.). Merriam-Webster. Retrieved 26 March 2015.

48. "RFC 2828 – Internet Security Glossary". Internet Engineering Task Force. May 2000. Retrieved 26 March 2015.

49. Singh, Simon (2000). The Code Book. New York: Anchor Books. pp. 14–20. ISBN 9780385495325.

50. Al-Kadi, Ibrahim A. (April 1992). "The origins of cryptology: The Arab contributions". Cryptologia. 16 (2): 97–126. doi:10.1080/0161-119291866801.

51. Schrödel, Tobias (October 2008). "Breaking Short Vigenère Ciphers". Cryptologia. 32 (4): 334–337. doi:10.1080/01611190802336097.

52. Hakim, Joy (1995). A History of US: War, Peace and all that Jazz. New York: Oxford University Press. ISBN 0-19-509514-6.

53. Gannon, James (2001). Stealing Secrets, Telling Lies: How Spies and Codebreakers Helped Shape the Twentieth Century. Washington, D.C.: Brassey's. ISBN 1-57488-367-4.

54. Diffie, Whitfield; Hellman, Martin (November 1976). "New Directions in Cryptography" (PDF). IEEE Transactions on Information Theory. IT-22: 644–654. doi:10.1109/tit.1976.1055638.

55. Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications), 2005, by Douglas R. Stinson, Chapman and Hall/CRC

56. Blaze, Matt; Diffie, Whitefield; Rivest, Ronald L.; Schneier, Bruce; Shimomura, Tsutomu; Thompson, Eric; Wiener, Michael (January 1996). "Minimal key lengths for symmetric ciphers to provide adequate commercial security". Fortify. Retrieved 26 March 2015.

57. "FIPS PUB 197: The official Advanced Encryption Standard" (PDF). Computer Security Resource Center. National Institute of Standards and Technology. Retrieved 26 March 2015.

58. "RFC 2440 - Open PGP Message Format". Internet Engineering Task Force. November 1998. Retrieved 26 March 2015.

59. Schneier, Bruce (1996). Applied Cryptography (2nd ed.). Wiley. ISBN 0-471-11709-9.

60. "NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition". Tech Beat. National Institute of Standards and Technology. October 2, 2012. Retrieved 26 March 2015.

61. Diffie, Whitfield; Hellman, Martin (8 June 1976). "Multi-user cryptographic techniques". AFIPS Proceedings. 45: 109–112.

62. Kahn, David (Fall 1979). "Cryptology Goes Public". Foreign Affairs. 58 (1): 153. doi:10.2307/20040343.

63. Rivest, Ronald L.; Shamir, A.; Adleman, L. (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM. Association for Computing Machinery. 21 (2): 120–126. doi:10.1145/359340.359342.