Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет (Институт) Центр заочного и дистанционного образования

Кафедра Радиотехнические устройства и системы диагностики

Направление подготовки 11.03.01 «Радиотехника»
(код, наименование)

ДОПУСКАЕТСЯ К ЗАЩИТЕ:
Заместитель заведующий кафедрой

_____ Пузырёв П.И.
(подпись)

«_____ » _____ 2023г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Устройства закрытия речевых сигналов
(тема работы)

Шифр БР-02068999-ЗРТ-181-30-460819 ПЗ

Обучающийся ЗРТ-181 Шестопалова Ангелина Владимировна _____ _____
(группа) (фамилия, имя, отчество) (подпись) (дата)

Руководитель Одинец А.И. _____ _____
(ФИО) (подпись) (дата)

Нормоконтролер Хоменко И.В. _____ _____
(ФИО) (подпись) (дата)

Омск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный технический университет»

УТВЕРЖДАЮ:
Заместитель заведующего кафедрой

_____ Пузырёв П.И.
(подпись)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Обучающийся Шестопалова Ангелина Владимировна Группа ЗРТ-181
(фамилия, имя, отчество)

Факультет (институт) Центр заочного и дистанционного образования
Кафедра Радиотехнические устройства и системы диагностики
Направление подготовки (специальность) 11.03.01 «Радиотехника»
(код, наименование)

Направленность (специализация) Радиотехника

Квалификация бакалавр

Наименование темы ВКР Устройства закрытия речевых сигналов

Утверждена приказом от « _____ » _____ 20 _____ г. № _____

Срок сдачи обучающимся законченной ВКР « _____ » _____ 20 _____ г.

Исходные данные к работе

Содержание ВКР (перечень подлежащих разработке разделов) Введение; анализ литературы, связанной с вопросами построения скремблеров; обзор методов сокрытия речевого сигнала, применяемых на практике; анализ известных программных алгоритмов псевдослучайных чисел и перестановок; обзор методов проверки псевдослучайных числовых последовательностей; программная реализация скремблера речевого сигнала; заключение; список используемой; литературы

Перечень графического материала (с указанием обязательных чертежей и (или) иллюстративного материала) 1. Общий порядок работы скремблера (1 лист); Блок - смеха комбинированного скремблера (1 лист); РСЛОС с заданной функцией обратной связи (1 лист); Перестановка фрагментов сигнала во временной и частотной области (1 лист); Программное обеспечение для устройств закрытия речевых сигналов (6 листов).

Дата выдачи задания « _____ » _____ 20 _____ г.

Руководитель _____ _____ _____

(подпись) (ученая степень, ученое звание) (ФИО)

Задание принято к исполнению _____ _____

(подпись обучающегося) (ФИО)

Приложение к заданию

на разработку устройства закрытия речевых сигналов

1 Область применения

Устройства закрытия речевых сигналов применяются в телефонных сетях общего пользования, спутниковой и радиорелейной связи, цифровом телевидении, а также для защиты лазерных дисков от копирования.

2 Назначение

Устройства предназначены для безопасной связи при передаче речевых сообщений.

3 Технические требования

3.1 Программное обеспечение

3.1.1 Требования к операционной системе, то есть к платформе, на которой будут функционировать информационная система;

3.1.2 требования к системе управления базой данных, которая будет осуществлять функционирование базы данных и связь базы с другими программными приложениями.

3.2 Выбор операционной системы. По требованию заказчика информационная система должна функционировать под управлением операционной системы Windows, установленной на имеющемся персональном компьютере заказчика и зарекомендовавшей себя как вполне надежная и стабильная система.

3.3 Техническое описание работы системы распознавания голосовых команд.

Библиотека распознавания голосовых команд составляет ядро системы голосового управления. С ее помощью разработчики могут добавить голосовое управление в создаваемые ими приложения.

Аннотация

Тема выпускной квалифицированной работы: «Устройства закрытия речевых сигналов»

Выпускная квалифицированная работа содержит 69 страницы текста, 28 рисунков, 8 таблиц и 17 использованных источников.

Предметом разработки является программа Qt5.

Целью работы является разработка программного скремблера речевого сигнала, выполненного в виде программы для компьютера.

В работе описана программная реализация скремблера речевого сигнала.

В работу входит введение, 5 глав, заключение, приложения.

Список сокращений

ГПСП – генератор псевдослучайных последовательностей;

ГПСЧ – генератор псевдослучайных чисел;

КИХ – конечная импульсная характеристика;

ЛРС – линейный регистр сдвига;

ПСЧ – псевдослучайные числа;

РСЛОС – регистр сдвига с линейной обратной связью;

РСОСП – регистр сдвига с обратной связью с переносом;

ТЧ – тональная частота.

Содержани

e

Введение.....9

1 Построение аналоговых и цифровых скремблеров речи, анализ литературы.....11

2 Обзор методов скремблирования речевых сигналов, используемых на практике.....14

2.1 Структурное скрытие в каналах связи речевой информации.....15

2.2 Классификация способов скремблирования.....16

2.3 Скремблирование во временной области.....19

2.4 Скремблирование в частотной области.....22

2.5 Комбинированное скремблирование.....25

2.6 Критерии оценки систем сокрытия речи.....27

3 Генерация псевдослучайных чисел и перестановок.....30

3.1 Анализ программных алгоритмов генерации псевдослучайных чисел.30

3.1.1 Метод срединных квадратов.....30

3.1.2 Полиномиальный конгруэнтный и линейный метод.....30

3.1.3 Аддитивный генератор псевдослучайных чисел.....32

3.1.4 Регистр сдвига с обратной линейной связью.....35

3.1.5 Регистр сдвига с обратной связью по переносу.....38

3.1.6 Регистр сдвига с нелинейной обратной связью.....39

3.1.7 Комбинирующие генераторы с фиксированным тактированием.....40

3.1.8 Генераторы с переменным тактированием.....41

3.2 Анализ известных способов генерации псевдослучайных перестановок 44

4 Обзор методов проверки статистических свойств генерируемых псевдослучайных числовых последовательностей.....48
5 Программная реализация скремблера оцифрованного речевого сигнала 52
5.1 Общее описание программы и интерфейса пользователя.....52
5.2 Ввод и вывод сигнала.....54
5.3 Генератор псевдослучайных чисел.....58
5.4 Генератор псевдослучайных перестановок.....62
5.5 Проверка статистических свойств псевдослучайных последовательностей числа.....64
5.6 Цифровая обработка сигнала.....69
Заключение.....77
Список используемых источников.....78
Приложения.....80

Введение

В течение достаточно длительного периода его развития человек накопил огромное количество опыта и огромное количество знаний относительно способов и методов разведывательного ведения. Естественно, первый опыт имел в основном военное значение, но потом нашел благодатный грунт для мирной реализации на промышленном шпионаже. Одним из главных способов разведывательного действия является доступ к каналам, используемым конкурирующей стороной. Прежде всего, нападение подвергается каналам телефонного сообщения, где, кроме речи, передается факсимильное, модемное сообщение. Также линия связи с телефонами, подключенными к ней, часто представляет собой какой-то черный ход в помещении, так как большинство телефонов из-за неправильной конструкции допускает утечку акустической информации из помещения.

Большинство контактов, в том числе деловых, и личных, все еще осуществляются с использованием общего телефонного сети ТСОП. В то же время ТСОП является самым уязвимым каналом коммуникации, который имеется для противоправной деятельности, связанной с бесконтрольным съемом акустических данных. Таким образом, проблема защиты линий телефона от незаконного прослушивания не теряет актуальности.

Одним из путей закрытия речевого сигнала, передаваемого по каналам провода и сети беспроводной связи, являются аналоговые скремблирования или дискретизации сообщения. Скремблирование осуществляется изменением параметров речи во временном или частотном направлении. Иногда сигналы разбиваются на отдельные части определенной длины, фрагменты которые после этого подвергаются перестановке или перемещению.

Аналоговый скремблер характеризуется наличием в связном канале фрагмента открытого сообщения, превращающегося в временную зону. Скремблер делает ставку на высокое надежное защитное обеспечение.

Камера позволяет защититься от любых видов прослушивания телефонов, включая специальное операторское оборудование. Трудность скремблирования можно увеличить в соответствии с потребностями, увеличив количество фрагментов речевой сигнализации.

В этой работе разработаны программные скремблеры речевых сигналов, которые реализуются в виде компьютерных программ. Программа написана на языке C++, для разработки графических интерфейсов использовался пакет библиотек Qt5. Эта программа может быть использована в качестве самостоятельного инструмента для предоставления конфиденциальной информации голоса или в качестве шаблона для разработки и разработки программных и аппаратных скремблеров.

1 Построение аналоговых и цифровых скремблеров речи, анализ литературы

Разработка аналоговых или программных скремблеров речи вызывает разрешения ряда проблем, сопряженных со сложностью их реализацией. В первую очередность необходимо постигнуть принципы работы скремблера, как они функционируют и их недостатки или преимущества. Затем рассмотрим вопросы выполнения операции скремблер.

Так что, чтобы реализовать скремблер, нужно проверять литературу о следующих вопросах:

– методы скремблирования речевого сигнала;

– спектр гласных и голосовых частот, спектр речевой частоты, спектр ширины спектра, необходимой для проведения речи в сетях телекоммуникаций;

– создание ГПСЧ;

– теория математической статистически и вероятностной статистики, закон распределения случайных величин, преобразования значений в требуемый распределение, статистическое свойство последовательностей случайных величин;

– средства проверки статистических параметров псевдослучайных последовательностей;

– цифровая обработка сигналов;

– средства разработки программных глушителей.

Был проведен анализ соответствующей литературы вышеуказанных вопросов.

Информация о методах шифрования и типах шифраторов представлена в книге «Основы практической защиты информации» Петракова А.В. и другие [1]. В книге рассматриваются основные проблемы защиты информации в системах связи и радиотехнических системах, а также системах всеобщего использования.

Доскональные материалы о свойствах звука и человеческой речи представлена в книжке «Объективное распознавание звуков речи» Е.Н. Мясников. [2]. В книге смотрятся вопросы механического определения речи и применение вербальных указаний для управления механизмами.

Обзор генераторов последовательностей псевдослучайных чисел представлен в учебном пособии «Генераторы псевдослучайных чисел», наложенном Слеповичевым И.И. [3]. Руководство включает представление ключевых способов генерации псевдослучайных чисел. Осмотрены вопросы криптостойкости, особенности и производительности генераторов псевдослучайных чисел. Кроме того, отдельные данные о известных способах генерирования псевдослучайного числа представлены в книге Шнайера «Практические криптографические методы» [4]. В книге изложена простая и доступная история исследований в области криптографии, начиная от всеобщего представления целей программиста, шифрующего передачу информации; примеры подобных программ.

Материалы о беспорядочных величинах и их распределениях хранится в учебнике «Теория вероятностей и математическая статистика. Базовый курс с примерами и заданиями» авторы Кибзун А.И., Горяинова Е.Р. и др. [5]. Книжка нужна для первого знакомства с базами теории вероятностей и математической статистики и для формирования навыков ответа фактических задач.

Вопросы о методах испытаний статистических параметров псевдослучайных последовательностей озарены в книге Дональда Кнута «Принципы программирования» [6] и М.А. Ивановой, И.В. Чугунковой «Теоретические, применяемые и оценочные качества псевдослучайных генераторов» [7] . Эта книга охватывает широкий круг тем, связанных с программными алгоритмами, включая описание генераторов псевдослучайных последовательностей и испытание их свойств. Вторая книжка напрямую сопряжена с генераторами псевдослучайных последовательностей.

Курс по цифровой обработке сигналов см. в книге Ричарда Лайонса Digital Signal Processing [8]. Это учебник по цифровому обрабатыванию сигналов с обширными картинками и явными примерами. Книга содержит короткое вступление в необходимый математический аппарат, принципы автоматического изображения сигналов (двоичные форматы), осматривает трудности периодической выборки. В разделах цифровой фильтрации представлены дискретные и быстрые преобразования Фурье. Цифровой фильтр с конечным, бесконечным импульсом; интерполированный фильтр КИХ.

Что относится к разработке программного скремблера, детальную информацию о языке программирования C++ и его типовой библиотеке возможно встретить на сайте cplusplus.com. Для разработки графичного интерфейса можно использовать комплект библиотек Qt 5, информация о которой презентована на официальном веб-сайте qt.io.

2 Обзор методов скремблирования речевых сигналов, используемых на практике

Методы против подслушивания, прежде всего, направлены на предотвращение утечки информации с простых гидроакустических и сейсмических каналов. Для повышения скорости перехвата используются сложные каналов утечки информации, в том числе электронные с использованием интегрированных устройств, РЧ и оптической системы. При этом защита информации от передачи содержит методики и средства для блокировки любых каналов, в которых происходит утечка звукового или аудио данного.

Защита разговоров с телефоном осуществляется активными и пассивными методами.

Основные активные методы:

а) подача в телефонной линии синфазного низкочастотного сигнала, метод синфазного низкочастотного сигнала, метод синфазного низкочастотного сигнала;

б) подача в телефонной линии сигнала с высокочастотным звуковым диапазоном, метод с высокочастотным звуковым диапазоном;

в) подача в связи с телефонной линией маскирующего сигнала высокочастотной ультразвуковой помехи, метод маскирующего сигнала ультразвуковой помехи;

г) повышение напряжения в связи с телефонной линией;

д) подача во время беседы в напряжение, компенсирующее постоянное содержание сигнала, методом «обнуления»;

е) подача на линию при положении телефонной трубки, маскирующего низкого частотного сигнала метода низкого частотного маскирующего сигнала метода низкого частотного маскирующего сигнала;

ж) подача на линию при приёме сообщений с маскирующим низкочастотным диапазоном с известным компенсационным методом;

з) подача высоковольтного импульса в линию телефона методом «выжигания».

2.1 Структурное скрытие в каналах связи речевой информации

Передача речевых данных является фундаментом коммуникаций в обществе человека, защита ее является основным вопросом защиты. Речевая информация передается через канал связи, при увеличении длины ключей увеличивается время его выявления.

Чтобы структурно сократить речевые данные в связи, используется шифрование и техническое блокирование.

В криптографии аналоговые речевые сигналы с микрофона приводятся в реорганизацию с помощью аналогового цифрового преобразователя к цифровому сигналу. При аналогичном и цифровом преобразовании, амплитуды сигнала измеряются одинаковым интервалом времени, называемым фазой дискреции.

Закрытие звукового сигнала узкополосного телефона осуществляется техническим или аналоговым методом маски. Эти методы получили название скремблеры перемешиваний. Скремблинге отличается от криптографической скрембляция тем, что в шифровании сообщение скрыто в символе, а в техническом зашифровании скрыто речевой сигнал, не превращаясь в цифру. При технических замыканиях характеристики начальных речевых сигналов модифицируются так, что они делаются аналогичным шуму, но занимают одну и ту же частотную полосу. Это дает возможность передавать скремблированные сигналы в обычных типовых телефонных каналах.

Главное преимущество методов технического замыкания заключается в несложности технического реализации скриптов, их минимальной стоимости и право применения скриптов абсолютно на любой канал связи, который предназначен для проведения речевого сообщения. Ключевым недостатком технических способов закрытия информации является меньшая степень

надежности закрытия данных. Также смесители, кроме простых с частотной инверсией, записывают колебания в реконструируемом речевом сигнале. Ее появление возникает из-за обратного преобразования сигналов на приёмной стороне, где частотные диапазоны и временные отрезки искажаются, что порождает некоторое искажение спектра восстановленных речевых сигналов. Комплексная задержка частей речевых сигналов также имеет вредные последствия. Изменения технических средств искажения приводит к уменьшению избыточного показателя восстановленной речевой сигнализации на 3-5 %.

Но, несмотря на все свои недостатки, методики временной и частотной скрембляции позволяют гарантировать защищенность информации на стратегических уровнях.

2.2 Классификация способов скремблирования

Подавители могут быть реализованы как программное или аппаратное устройство, которое обрабатывает аналоговые или цифровые голосовые сигналы в течение времени. В результате сигналы теряют свою семантическую ценность и становятся неразличимыми для посторонних лиц, либо на выходе скремблера появляется случайная битовая последовательность.

Побитовое цифровое скремблирование получается несложным наложением битов речевого сигнала на биты гаммы, см. рис. 1. Важнейшим моментом в схеме является синхронизация гаммовых генераторов на стороне приема и переноса инфекций. Этот ход решается путем передачи в последовательности информации параметров генератора, либо путем специальных синхронических кадров или отдельного канала соединения.



Рисунок 1− Побитовое цифровое скремблирование

Глушители активно используются для защиты телефонных переговоров и радиосвязи. В случае глушения может произойти изменение сигнала речи по трем показателям: времени и частоте, а также амплитуды. В маневренных радиосвязных системах основным образом отыскались фактические использования частотных и временных преображений сигнала и их комбинаций.

При наличии в радиоканале помех, значительно затрудняющих восстановление амплитуды речевого сигнала, при глушении применяются исключительно амплитудные преобразования.

Основные технические характеристики скремблеров:

– качество восстановления сигнала;

– остаточная разборчивость;

– уровень закрытости информации;

Побитовое цифровое скремблирование аспекты закрытия информации могут, определены из статистических параметров длины ключа, гамма-излучения, выдаваемого генератором, и других параметров.

Остаток разборчивости обусловлен вероятностью того, что отдельные параметры речи паузы, длительность слов, голос диктора, т.д. восстановляются из скремблированного сигнала при помощи приемников без дескриптора.

Качество воспроизведения сигнала зависит от того, как точно сигнал перед скремблером приемной части соответствует исходному в момент

скремблирования. Приемлемое или коммерческое качество реконструированной речи — это качество, когда слушатель не особо усиливает значение голоса и значения произносимой речи.

Качество восстановления сигналов зависит от самого метода установки помех, а также от других параметров. Рис 2. Помехи в радиосвязи могут возникать от пересеченной местности или движения приемника либо передатчика;



Рисунок 2 − Классификация скремблирования речевых сигналов

Скремблеры еще возможно разбить на две категории:

– статические скремблеры;

– динамические скремблеры.

Статический скремблер не меняет правила шифрования, которые они устанавливают в зависимости от времени. Это позволяет упростить конструкцию скремблера, но значительно повышает возможность нарушения конфиденциальности языка. Такие скремблеры можно использовать только для краткосрочных передач или в случаях, когда конфиденциальная информация не передается. После этого необходимо завершить показ связи и изменить характеристики скремблера.

Динамические скремблеры, отличающиеся от статических скриптов, изменяют правило, по которому они скрипты, внедряя в систему псевдослучайные последовательности. Это повышает надежность синтеза, но эти синтезы часто вызывают более сложную синхронизацию системы.

2.3 Скремблирование во временной области

Имеется два ключевых способа скремблирования голосового сигнала:

− временные перестановки сегментов речевого сигнала;

− во временной области (временная инверсия речевых сегментов).

Скремблирование во временной области содержится в том, что участок речевого сигнала инвертируется во времени, то есть передается в обратном порядке. Этот способ не обеспечивает бесконечной конфиденциальности передаваемой речи, поскольку можно легко отменять инверсию и извлечь из исходного сигнала простыми аппаратными средствами. Однако без таких средств, преступнику будет мудрено понять, о чем идет речь.

Метод временного преобразования сигнала − отрезок речи определенной длительности делится на части − фрагментов или отрезков и затем смешивается по определенному правилу. Качество шифрования повышается по мере уменьшения длительности перетасовываемых сегментов. Обратите внимание, что по мере увеличения продолжительности сегмента сигнала, зашифрованного в аппаратном или программном устройстве, увеличивается задержка перед передачей речи.

Визуальное представление операции обращения сигнала во времени показано на рисунке 3. Здесь сегмент сигнала (слева), начиная с момента времени 0, инвертируется для получения нового сигнала (справа).

Ход временной перестановки показан на рисунке 4. Сегмент сигнала делится на 4 фрагмента, в непринужденных порядках 1-2-3-4 и потом эти фрагменты переходят в измененный порядок 2-4-1- 3.

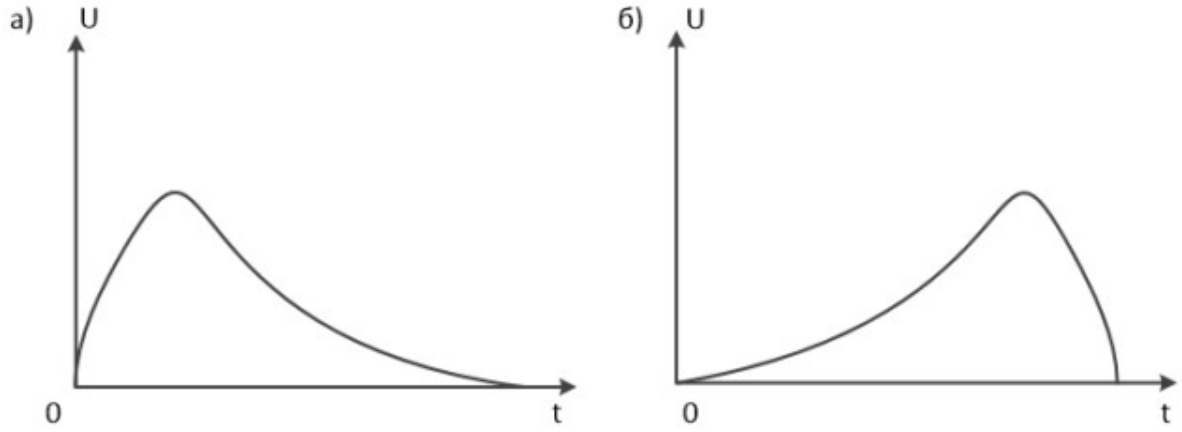

Рисунок 3 − Временная инверсия сигнала: а − исходный сигнал; б − инвертированный сигнал

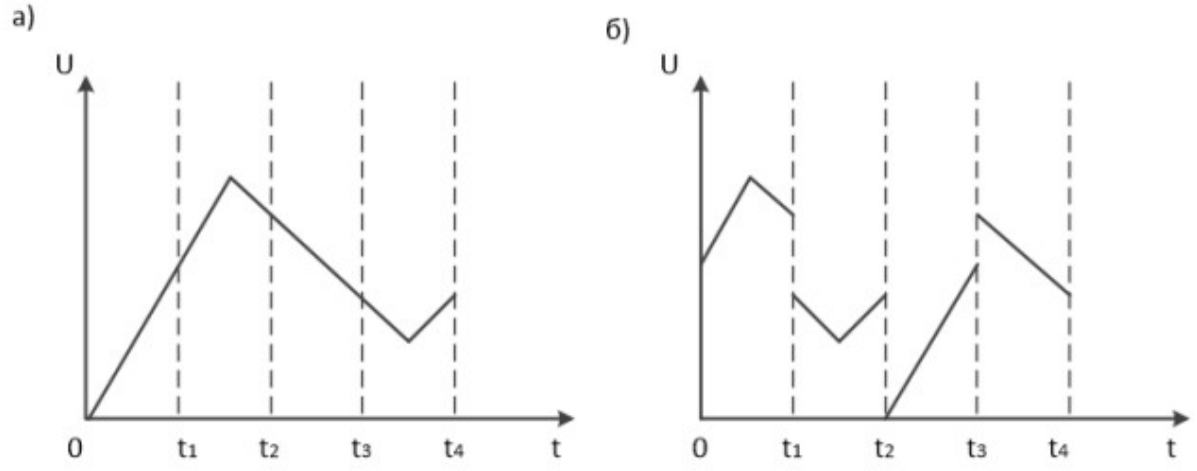

Рисунок 4 − Перестановка фрагментов сигнала: а − исходный сигнал; б − преобразованный сигнал

На рисунке 5 и 6 более подробно показаны процессы перестановки во временной области.

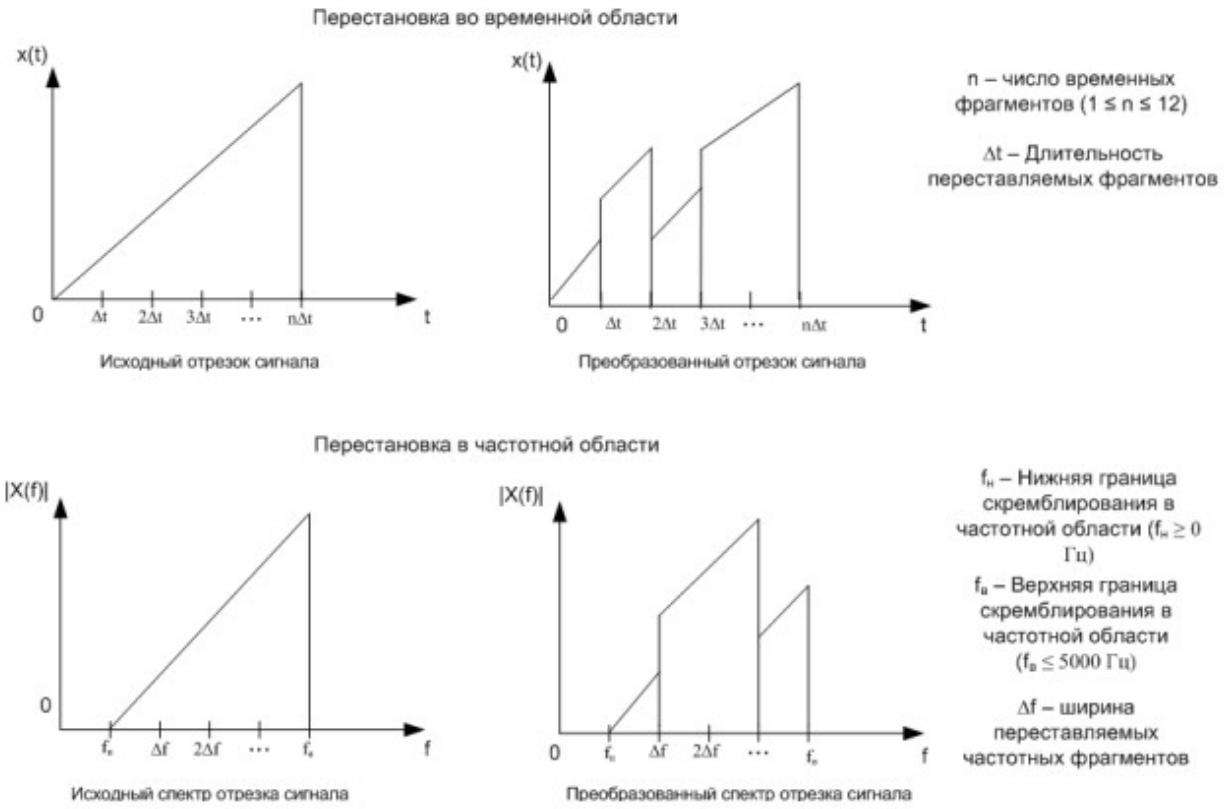

Рисунок 5 − Перестановка фрагментов сигнала во временной и частотной области

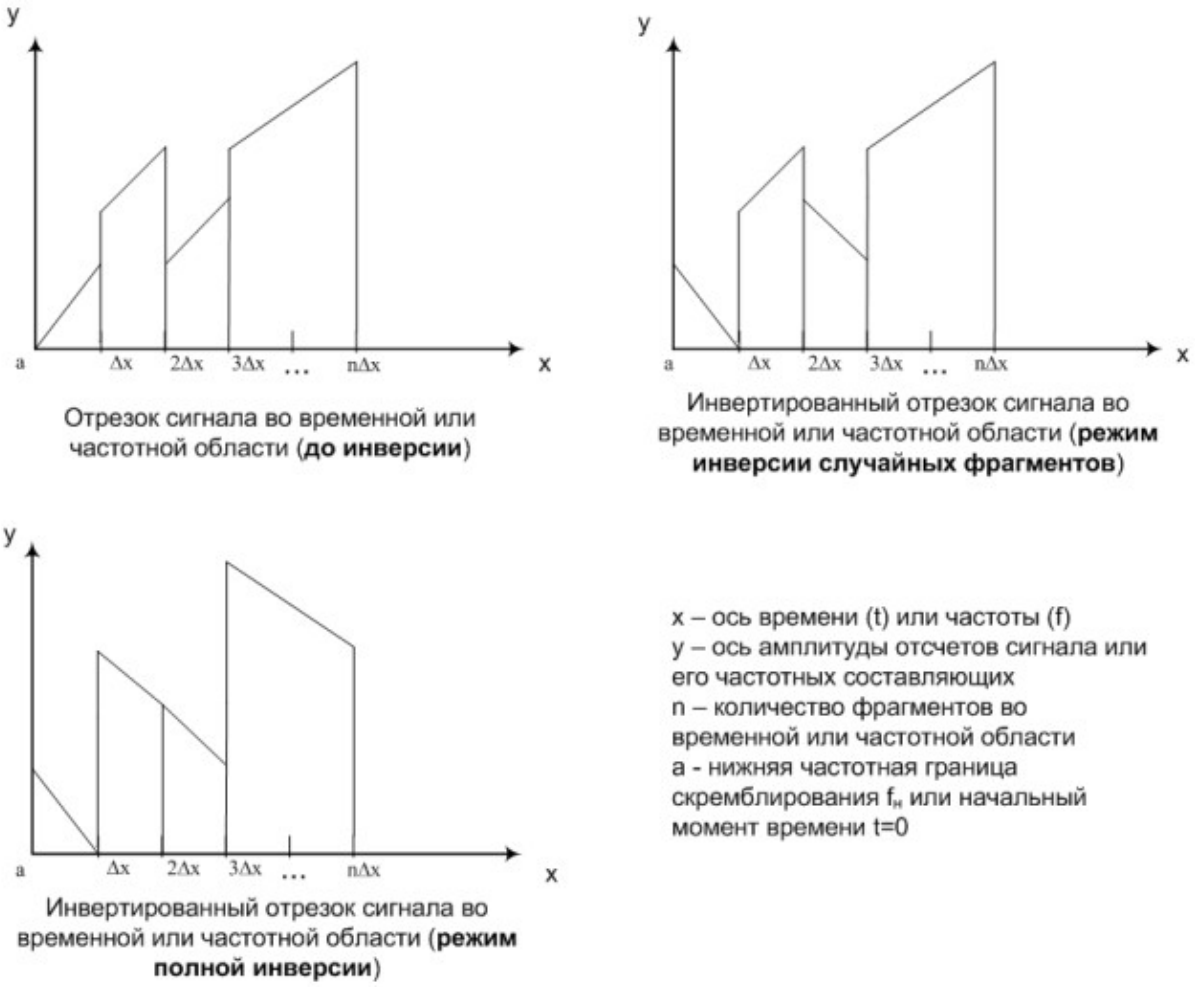

Рисунок 6 − Инверсия сигнала во временной и частотной области

2.4 Скремблирование в частотной области

Скремблирование в области частоты осуществляется на спектре частоты, а именно на самом спектре сигнала, который выражается в виде функции зависимости времени. На рис. 7 представлен частотный спектр русских звуков.

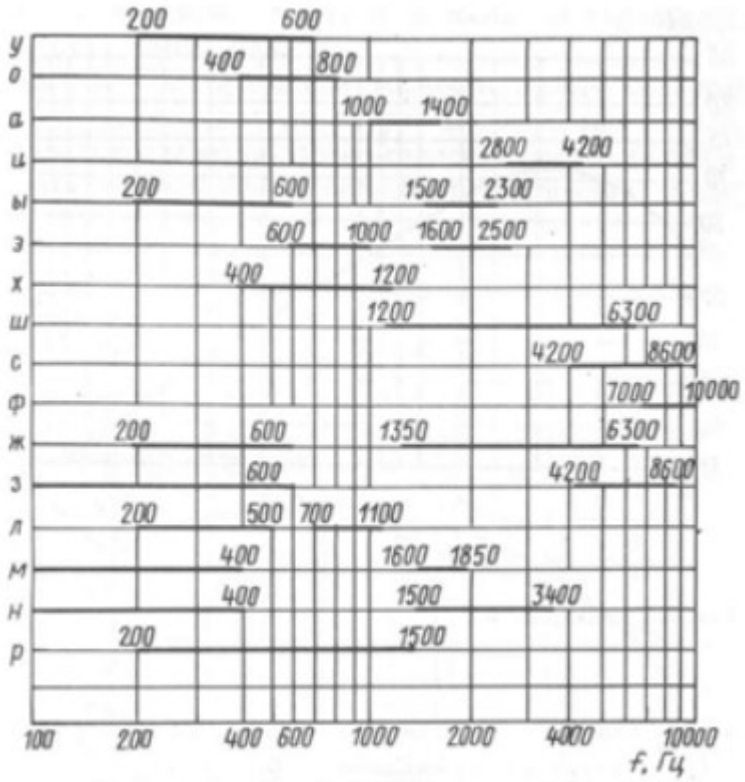

Рисунок 7 − Частотный спектр звуков русского языка

Человеческая речь имеет обширную полосу частот приблизительно 50 − 10 000 Гц, человеческое ухо способно распознать звуки в диапазоне 20 Гц − 25 000 дБ. В вокальном спектре выделяют частоты, в которых сконцентрирована ключевая деятельность сигнала и эти частоты именуют формантами. В русской речи распознают 5 формантов, но для ее удовлетворительного распознавания слушателем полно присутствие в сигнале лишь первых двух-трех формант. Это облегчает передачу речи по беспроводным или проводным сетям связи.

Типовой канал ТЧ, используемый как в телефонной, так и в радиосвязи. Обладает действенной шириной спектра 3100 Гц (от 300 Гц до 3400 Гц). Данной ширины полно для передачи человеческой речи с возможностью идентификации личности говорящего.

Первым преобразованием, которое можно использовать для кодирования речевого сигнала в частотной области, является инверсия частоты. При таком рассмотрении в спектре частот отрезков сигнала

определённой длины отличаются основные частоты $F_ц$ частота инверсии. Амплитуда всех составляющих частоты, расположенных по одной стороне от этого частоты, с более высокой или низкой частотой отражается на диапазоне составляющих частоты, расположенных по другой стороне от центральной. Это преобразование соответствует вращению полосы частот сигнала окрест частоты $F_ц$. На рисунке показано это преобразование на рис. 8.

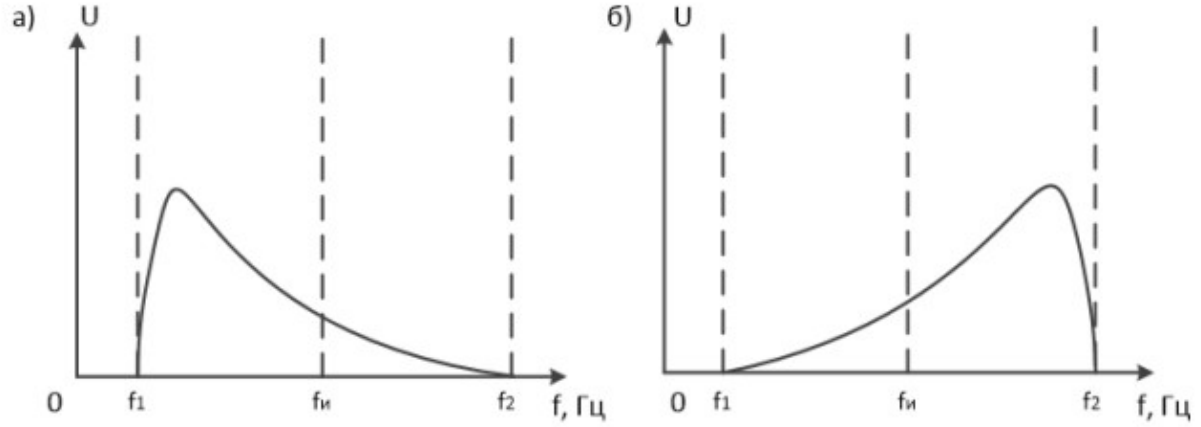

Рисунок 8 − Частотная инверсия сигнала: а − спектр исходного сигнала; б − инвертированный спектр сигнала

Сей способ дозволяет уменьшить уровень замыкания, так как при захвате свободно определить значение частоты, подходящей типичной точке инверсии в спектральном диапазоне речевого сигнала.

На рисунке 6 инверсия исполняется в полосе частот от f1 Гц до f2 Гц, частотой инверсии представляется частота посередине между ними. Также используются частотные инверсии более двух диапазонов речи. Вначале участки пропускания всех речевых сигналов разбиваются на частотных поддиапазонах, в большинстве своем на два, которые потом инвертируют. Источником к данной налаженности представляется частотное деление спектра сигнала; подобные скремблеры, отличаются от моделей, использующих одинарную инверсию частоты, порядком увеличивают уровень предохранения информации.

Спектр сигнала можно разделить на несколько полос одинаковой или переменной ширины, каждая из которых инвертирована по частоте. Чем

больше подобных поддиапазонов в спектре, тем менее требовательной будет зашифрованная речь и тем сложнее будет ее восстановить, не зная ключевого слова. Вторым способом реализации частотного глушения является разделение частотного диапазона на несколько диапазонов, а затем их перестановка. Частотный спектр, полученный в результате этого действия, снова преобразуется в функцию времени, которая представляет собой скремблированный речевой сигнал, с использованием обратного преобразования Фурье. Модель подобного преобразования представлен на рисунке 9.

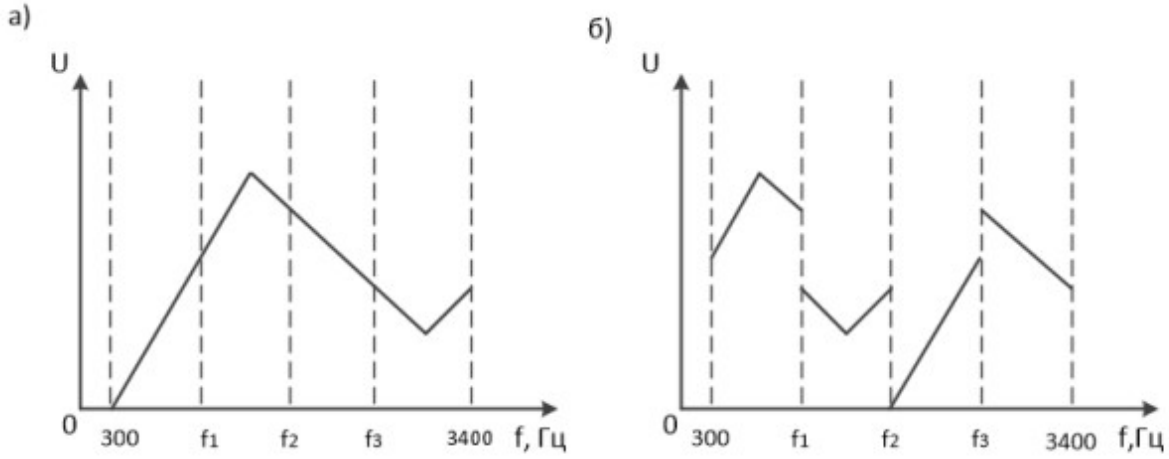

Рисунок 9 − Перестановка фрагментов частотного спектра сигнала: а − спектр исходного сигнала; б − спектр преобразованного сигнала

На рисунках 5 и 6 более подробно показаны процессы инверсии в частотной области и перестановки.

Кодирование частоты позволяет блокировать речь с тем же количеством поддиапазонов, что и исходный сигнал. Хотя операции в частотной области вызывают подготовительного преобразования сигнала во времени в частотное представление — прямого преобразования Фурье (через обратное преобразование Фурье). К этому фактору добавляется ряд дополнительных требований к производительности скремблеров.

2.5 Комбинированное скремблирование

Сочетая временной и частотной скремблер, можно значительно улучшить закрытие речи. Комбинированный скремблер значительно сложнее обыкновенных, и приводят к компромиссу при выборе уровня закрытия и остаточной разборчивой сигнала. Часть разных систем, которые работают на этом принципе, была урезано измышлениями разработчика.

В качестве примера такой системы мы посмотрим скремблер, схему которого можно увидеть на рис. 10. В данном скремблере операции частотной и временной перестановки дискретного сегмента речевой сигнала осуществляются через четыре процессора цифрового сигнала, одна из которых может выполнять функцию ПСП-генератора.

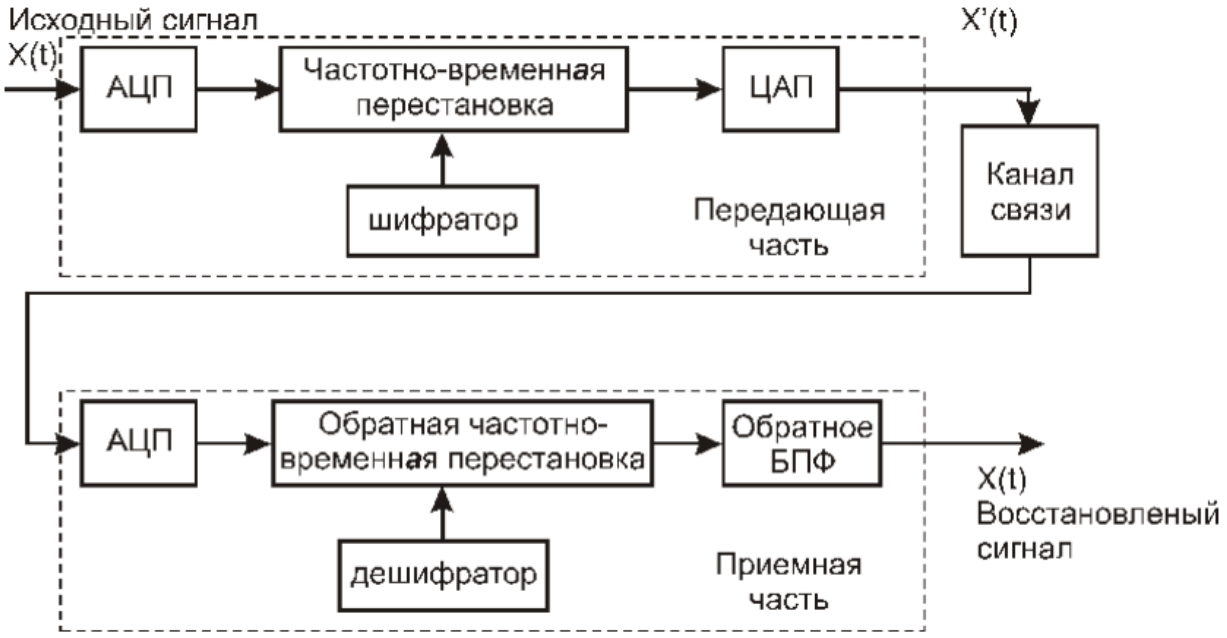

Рисунок 10 − Блок-схема комбинированного скремблера

Степень закрытия речи, может значительно повысить с помощью сочетания быстро-частотной шифровки. Комбинированные скремблеры не столь сложны, как обычные скремблеры, и вызывают компромиссы в плане уровня запуска, периода задержки, остатков разборчивости и количества искажений в ранее созданных сигналах, сложности систем.

Самый безусловный прием комбинированного скремблирования является просто применение поочередных преобразований в временную

область, а потом в частотную область речи. Для этих целей можно использовать сложные схемы, где сигнал передается в виде определенного временного пространства с перемещением фрагментов частоты. Как видно на рис. 11.

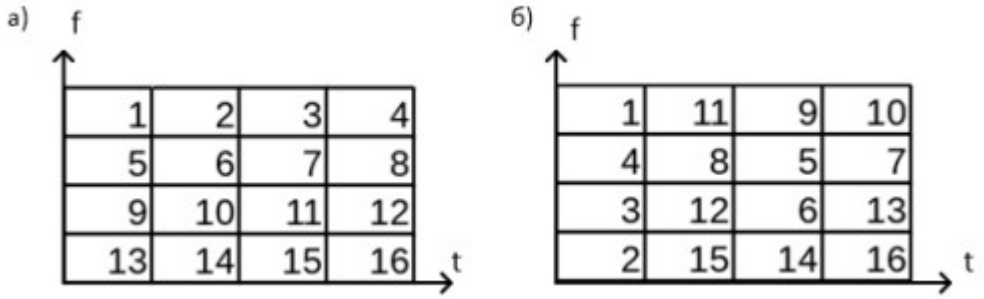

Рисунок 11 − Комбинированное скремблирование в частотно-временной плоскости: а − вид частнотно-временной плоскости в исходном сигнале; б − вид частнотно-временной плоскости после преобразования

2.6 Критерии оценки систем сокрытия речи

Есть четыре ключевых критерия, по которым оцениваются устройства для окклюзии речи:

− основные технические характеристики системы;

− разборчивость речи;

− степень закрытия;

− распознавание говорящего.

Параметры узкополосных систем передачи речи, воздействующие на качество восстановленного узкополосного речевого сигнала, определяются методами кодирования, шумовым воздействием, способами модуляции, критериями распространения и приборными погрешностями. Шум, искажение всячески влияют на эффективность какой-либо части системы и ухудшение качества звука, который встречается у пользователя, представляет собой комбинированный эффект снижения производительности компонентов. Имеются беспристрастные методы оценки специфики речи, а также систем, не имеющие никакого отношения к сопоставлению этих

данных с дискретными узкополосными системами связи, где речевой сигнал первым образом преобразуется в среду передачи в канал связи на приемнике.

Имеются индивидуальные методы измерений разборчивости и естественности различаются огромной сложностью, поскольку многие зависят от использования словаря или канала коммуникации. При этом проводить измерение, чтобы получить статистически правдоподобные и достоверные оценки системы по изменяющимся критериям, дорого стоит.

Проблемы, связанные с использованием радиоканалов, возникают из-за неопределенности критериев распространения, а также невозможности создания итогов без использования радио−канальных моделей.

В системе дуплекса особо влияет на качество времени задержки сигнала, который вносит речевой скремблер.

Аналоговый микшер используется в тех случаях, когда затруднены цифровые системы запуска речи. Из-за погрешностей в передаче плохих дальних или стационарных радиолиний они обеспечивают тактическую устойчивость и хорошую защиту от разговоров извне, как соседей, так и коллег. Для подобных приложений используются системы постоянной блокировки, т.е. производящие криптографию в фиксированном ключе.

При необходимости защищать конфиденциальность информации конкурентов, имеющих технические и специальные техники или имеющих прогрессивное оборудование, можно использовать аналоговые стационарные микшеры. Но такие системы тормозят злоумышленников в разработке метода шифрования и сильно тормозят выполнение действий, получая информацию из перехваченного сообщения, время, затрачиваемое на это, сократит количество приобретенной информации в процессе утечки сообщения.

В случае бытовых закрытых устройств последовательность с дополнительной информацией об ключе, передаваемой в момент передачи сообщения, передаётся одновременно до того, как начало сообщения начинается. Злоумышленник имеет только один способ его открыть: просмотреть множество первостепенных установок. При ежедневной

модификации ключей злоумышленник должен был перебрать много вариантов и использовать популярный метод перевода языка для обнаружения истинного порядка ключей.

Когда предполагается, что для извлечения значительной информации злоумышленники могут прибегнуть к услугам профессионалов и их технического арсенала. Для обеспечения отсутствия утечек данных необходимо использовать системы закрытия речей с стратегической защитой. Это может гарантировать дискретизацию голоса и дальнейшее шифрование, а также аналоговые скремблеры других типов. После перехода аналогового сигнала к цифровой форме применяются криптографические блокировки, которые аналогично блокируют данные. Потом, приобретая засекреченные сообщения, они вновь превращаются в аналоговые сигналы и подаются на связную линию. Для того, чтобы обнаружить принятый сигнал на конце приема, производится обратное преобразование. Данные гибриды свободно подключаются к имеющимся сетям связи, гарантируют высокую защиту голоса от аналоговых скремблеров.

3 Генерация псевдослучайных чисел и перестановок

3.1 Анализ программных алгоритмов генерации псевдослучайных чисел

3.1.1 Метод срединных квадратов

Наиболее важным алгоритмом извлечения одинаково распределенных псевдослучайных чисел представляется методом средних квадратов (или метод центрального квадрата). Он состоит из следующих частей:

а) избирается отдельное изначальное беспорядочное число X_0, обладающее n-разрядное представление;

б) изменение числа X_i в квадрат и получение нового числа Y_i, имеющее разрядность равную 2n;

в) из полученного числа Y_i выбирается n разряд, взятый из середины числа. Это число X_i+1 применяется в следующем алгоритме.

Числа X_i представляют выходные значения в алгоритме. Например, при взятии исходного значения X_0, равного 6433, следует следующая серия псевдослучайных значений: 3834, 6995, 9300; и т.д.

Этот алгоритм прост в программной реализации, но имеет малый период генерации и наличие неподходящих начальных (промежуточных) значений.

Метод не годится для криптографического предохранения информации и скремблирования речевого сигнала.

3.1.2 Полиномиальный конгруэнтный и линейный метод

Более современным методом, нежели метод средних квадратов (но очень простым), является метод линейной конгруэнтности.

Параметры метода представляют собой ряд целочисленных значений:

– начальное значение X_0;

– множитель a;

– модуль m;

– приращение (инкремент) c.

Для этих значений следует выполнять следующее неравенство:

$$m > 0$$

$$0 \le a \le m$$

$$0 \le c \le m$$

$$0 \le X_0 \le m$$

Алгоритм функционирует по следующей формуле:

$$X_{k+1} = \left(a X_k + c\right) \cdot mod \;\; m$$

Значения X_k – выходные значения генераторов. Их вид определяется по выбранному начальному значению X_0 и числам m, a, c.

Период сгенерированной последовательности не выше значения модуля m. Если надо извлечь точное значение периода, равное m, то при подборе параметров надо осуществить ряд условий:

а) значение $a \;\; минус \;\; 1$ кратно p, для любых простых p - разделителей m;

б) число c и m являются взаимно простыми, наименьший общим делителем чисел c и m должна быть равна 1;

в) значение $a \;\; минус \;\; 1$ кратно 4. При реализации выгодно выбирать m, равным 2e, в котором e является числом битов в машинном слове. Потому что это позволяет избежать сравнительно медлительных операций.

Статистические характеристики полученных последовательностей чисел полностью определяется выбором a и c констант при заданном разряде e. Для указанных констант установлены условия, которые гарантируют удовлетворение качества полученного случайного числа. Отсюда выбирают следующие пункты. Значение: a равна 1664525, c равна 1013904223 и m равна 232. При выборе непригодных параметров в последовательности чисел можно увидеть определенную закономерность, которая выражается в появлении яркие удлиненные диагональные полосы, см. рисунок 12. На основе данного метода можно построить полиномиальный когерентный метод. В таких генераторах предшествующее значение X_i входит для

следующего выражениеX_i+1 в виде своего квадрата. Общее выражение полиномиального генератора степени n:

$$X_n = \sum_{i=0}^{r} a_i X_{n-1}^{i} \ mod \ m, \ n \geq 1$$

Но хотя рассматриваемый генератор ПСП не используется для защиты криптографических данных, его можно использовать в тех случаях, когда требуется обеспечить гарантированную стойкость шифрования. Перед использованием этого метода требуется предварительная проверка полученной последовательности, так как ее качество зависит от параметров генератора. Также необходимо, чтобы значения сгенерированных чисел не были доступны потенциальному злоумышленнику в явном виде, а подвергались некоторому преобразованию, ведь зная точную форму строки, злоумышленник может легко восстановить параметры генератора.

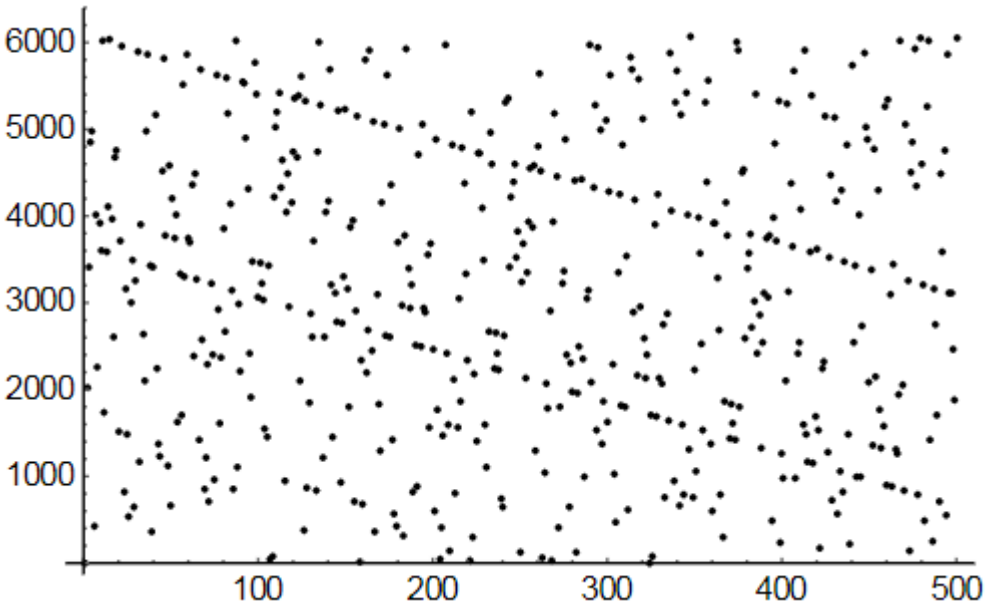

Рисунок 12 − Закономерность в распределении псевдослучайных чисел при использовании некоторых параметров линейного конгруэнтного метода

3.1.3 Аддитивный генератор псевдослучайных чисел

Данные генераторы используют число, сгенерированное ими на 2 или более предшествующих рабочих этапах, чтобы вычислить следующее выходное число. В простейшем случае применяются лишь два предыдущих

числа в последовательности, например, в обыкновенной последовательности Фибоначчи или для извлечения чисел генератором на k шагов назад:

$$X_{n+1}=\left(X_{n-14}+X_{n-55}\right)\ mod\ \ m, n\geq 55$$

В данном выражении следующее значение Xn+1 зависит от двух значений, полученных на 14 и 55 шагов назад. Фибоначчи — это так именуемая отложенная последовательность.

В общем, вы можете использовать выражения, содержащие линейную комбинацию k предыдущих элементов строки:

$$X_n=\left(a_1 X_{n-1}+\cdots+a_{k-1}X_{n-k+1}\right)\ mod\ \ p$$

Далее рассмотрим полином:

$$x^k-a_1 x^{k-1}-\cdots-a_k=0$$

Период подобного генератора будет равен p^k-1 при условии, что полином представляется простым над полем GF(p) и хотя бы один из элементов от X_0 до $X_k\neq 0$.

На основе данных генераторов создано порядком алгоритмов шифрования, таких как: Mush, Pike и Fish.

Mush − генератор взаимно прореживающий. Это два аддитивных генераторов: A и B. При установке бита переноса A, часы B. При установке бит переноса B, часы A. Часы A и установите бит переноса при переполнении. Мы тактируем B и устанавливаем бит переноса на переполнение. Конечный результат − это XOR выходов A и B. Простой способ использовать те же, как и Fish.

Некоторые генераторы, основанные на использовании рекурсивных отношений между порядковыми выходными числами, такие как Pike, обеспечивают достаточную степень устойчивости к взлому и выдают числа с хорошими статистическими свойствами. Следовательно, этот метод можно использовать для защиты криптографической информации.

Fish является аддитивным генератором, основанным на методах, применяемых в прореживаемых генераторах. Он использует два аддитивных генератора, работающих по следующим формулам рекуррентности:

$$A_i = \left(A_{i-55} + A_{i-24} \right) \ mod \ 2^{32}$$

$$B_i = \left(B_{i-52} + B_{i-19} \right) \ mod \ 2^{32}$$

Данные последовательности прореживаются в соответствии с младшим значение бита B_i: при этом значение бита B_i равна 1 − используется пара, при этом 0 – не используется. C_i является последовательность употребляемых слов A_i, а D_i и является последовательностью употребляемых слов B_i . Для создания двух 32-битовых результатов K_{2j}и K_{2j+1} эти слова применяются парами - C_{2j}, C_{2j+1}, D_{2j}, D_{2j+1}». Этот генератор небесполезен для применения криптографии.

Pike использует три аддитивных генератора типа:

$$A_i = \left(A_{i-55} + A_{i-24} \right) \ mod \ 2^{32}$$

$$B_i = \left(B_{i-57} + B_{i-7} \right) \ mod \ 2^{32}$$

$$C_i = \left(C_{i-58} + C_{i-19} \right) \ mod \ 2^{32}$$

Схематичное изображение ГПСП показано ниже на рисунке 14.

Биты переноса C1, C2 и C3 сумматоров OR1, OR2 и OR3 используются, как управляющие биты. В зависимости от состояния их, синхронизация решает, которые регистры необходимо сдвигать. Если все C1, C2 и C3 одинаковые, то изменение происходит в каждом из трех регистров. И если два переносных бита имеют одно и то же состояние, они сдвигаются в соответствующих регистрах. Изменение происходит на 8 циклах.

Сейчас неизвестно, как взламывать этот шифр, что позволяет использовать этот шифр, что позволяет защитить криптографическую информацию.

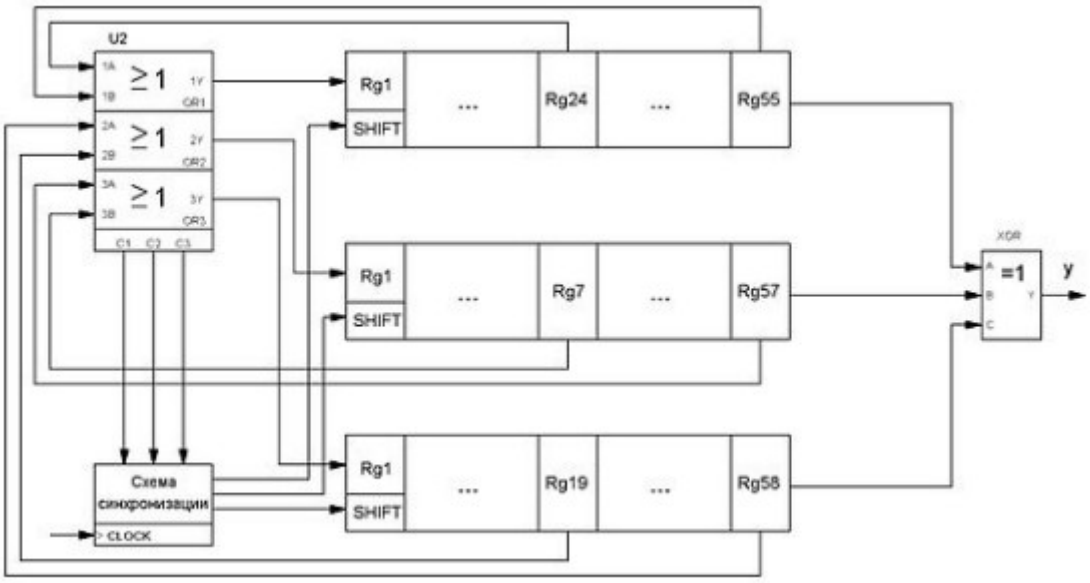

Рисунок 13 − Генератор PIKE

3.1.4 Регистр сдвига с обратной линейной связью

Самый популярный способ получения последовательности случайных чисел является регистр. Этот регистр представляет собой двоичные триггеры, значения которые сдвигаются по выбранному направлению внутри регистра для переноса биты и новых битов генерируются при изменении состояния каких-либо ячеек. Сдвиговые регистры с линейной обратной связью описываются мнимым многочленом над полем GF(2), который имеет общий вид:

$$K(x) = a_n x^p + a_{n-1} x^{p-1} + \cdots + a_1 x + 1,$$

где:

− a_n − число, которое принимает значение 0 или 1;

− x^k − местоположение регистра (0 или 1) в положении k;

− p − длина регистра.

Можно взять образующий полином, имеющий вид:

$$K(x) = 1 + x^2 + x^5 + x^7$$

На рис. 14 (см. ниже) показано представление РСЛОС, соответствующее этому полиному. В процессе работы значения операции генератора, содержащиеся в ячейках, сдвигаются вправо и самый левый бит в регистре заменяется битом, полученным применением операции «только или» последовательно к битам, содержащимся в ячейках с номерами 7, 5 и 2.

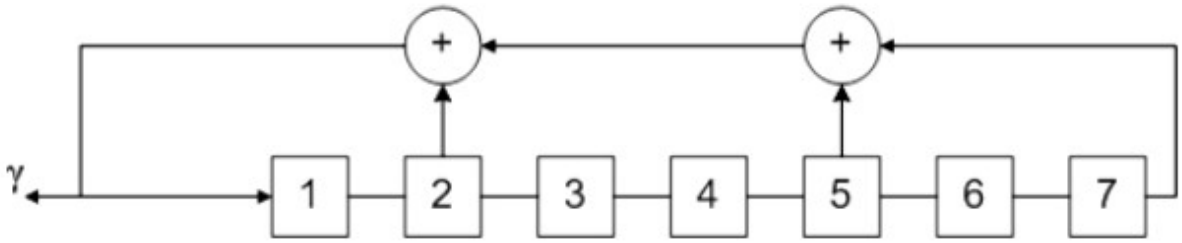

Рисунок 14 − РСЛОС с заданной функцией обратной связи

Для создания качественного РСЛОС, способного генерировать последовательность чисел с требуемым периодом и устойчивого к взлому, необходимо знать упомянутые выше свойства порождающих полиномов. В настоящее время эти свойства хорошо изучены математиками, что делает РСЛОС чрезвычайно перспективным инструментом в криптографии.

Первое, что нужно отметить, это период последовательности, РСЛОС. Из-за конечности множества возможных состояний регистра всегда происходит изменение гаммы: рано или поздно какое-то состояние должно произойти дважды. С этого момента гамма начинает повторяться. Длина повторяющейся части шкалы называется периодом регистра. Количество битов гаммы до того, как она начнет повторяться, называется пред периодом регистра. Обратите внимание, что период и предыдущий период зависят от начального состояния регистра. Для того чтобы максимальный период последовательности был равен 2^p минус 1 (больше быть не может), порождающий многочлен примитивен. Многочлен K(x) степени p называется примитивным, если он не делит нацело многочлен вида x^s минус 1 (s меньше 2^p минус 1). Существуют таблицы, содержащие список таких многочленов. В таблице 1 приведены некоторые полиномы степени 32 и выше.

Таблица 1 – Таблица коэффициентов некоторых примитивных многочленов степени 32 и более

Коэффициенты образующего многочлена	Максимальный период двоичной последовательности
(32,7,6,2,0)	$2^{32}-1$
(33,16,4,1,0)	$2^{33}-1$
(34,8,4,3,0)	$2^{34}-1$
(32,2,0)	$2^{35}-1$
(36,6,5,4,2,1,0)	$2^{36}-1$
(37,6,4,1,0)	$2^{37}-1$
(38,6,5,1,0)	$2^{38}-1$
(39,4,0)	$2^{39}-1$
(40,5,4,3,0)	$2^{40}-1$
(41,3,0)	$2^{41}-1$
(42,7,4,3,0)	$2^{42}-1$

Если говорить о начальном заполнении ключа РСЛОС, то он не должен быть только нулевым, поскольку в данном случае все выходные биты будут ненулевыми. Можно использовать выбранное пользователем заполнение или использовать энтропические источники, например, текущие часы, температуры процессора, и так далее. Для того, чтобы оценить качество сгенерированной последовательности, рекомендуется проводить различные тестирование случайности, об этом говорится в соответствующей части пояснительной записи.

РСОЛС могут использоваться как автономные генераторы ПСЧ, а также могут быть включены во многие другие схемы. Это обеспечивает более высокую устойчивость к растрескиванию или лучшие статистические свойства ПСЧ.

РСЛОС легко реализовать как программный или аппаратный модуль. Учитывая мощность современных компьютеров, они могут использоваться

для хранения, например, гаммы двоичных данных, а также сигналов в реальном времени.

3.1.5 Регистр сдвига с обратной связью по переносу

РСОСП представляет собой регистр сдвигов и другой так называемый регистр переносов. Рабочий механизм этой системы состоит в добавлении бит последовательности отвода и содержимого переносного регистра. Остаток 2 модуля становится новой нулевой битой основного регистра сдвигов. Целый результат, разделенный на две части, как видно на рис. 15, является новым значением переносного регистра.

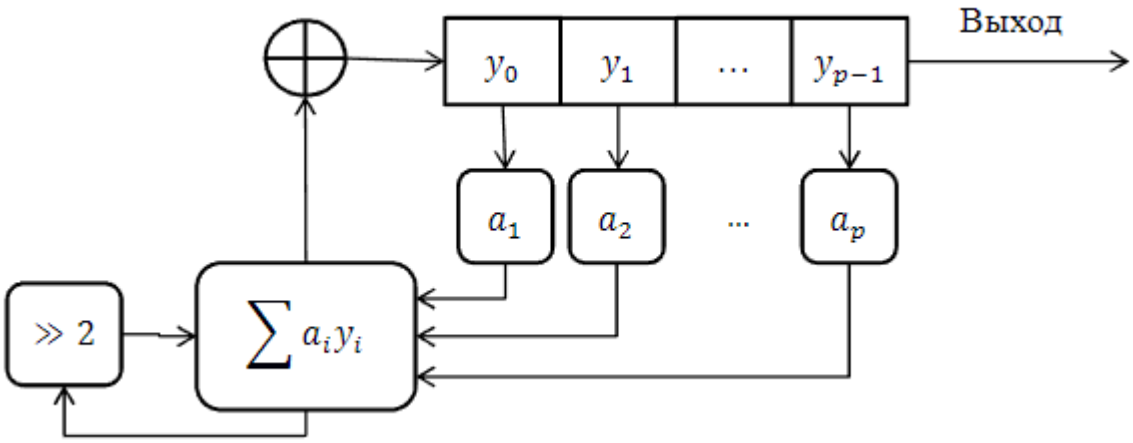

Рисунок 15 − Регистр сдвига с обратной связью по переносу

Максимальный период последовательности такого генератора:

$$N = 2a_1 + 2^2 a_2 + 2^3 a_3 + \cdots + 2^n a_n - 2$$

В отличие от большинства РСЛОС, для РСОСП существует задержка перед тем, как он войдет в режим цикла, т. е. он начнет генерировать циклическую последовательность. Еще одним недостатком является отсутствие определенного набора ключей, использование которых приводит к значительному ухудшению статистических свойств получаемой последовательности РСОСП.

Следовательно, введение регистра переноса приводит к большей сложности и защищенности от корреляционных атак, чем при использовании обычного РСЛОС. Однако необходимость использования РСОСП в криптографических целях сомнительна из-за особенностей, перечисленных выше. Как известно, один из вариантов РСОСП, выбранных проектом eSTREAM для выявления новых эффективных потоковых шифров (например, F-FCSR), позднее был исключен из списка конкурентов, поскольку осенью 2008 года продемонстрировал криптографическую уязвимость.

3.1.6 Регистр сдвига с нелинейной обратной связью

Линейность обычных регистров сдвигов с линейными обратными связями значительно ограничивает применение LFSR, поскольку упрощает прогнозирование поведения подобных регистров сдвигов. Таким образом, нелинейную функцию обратного сообщения можно применять вместо нелинейной. Обзор регистра сдвигов с нелинейной обратной связью показан на рисунке 16. Функция f (b_0, b_1,…, b_{m-1}) нелинейна.

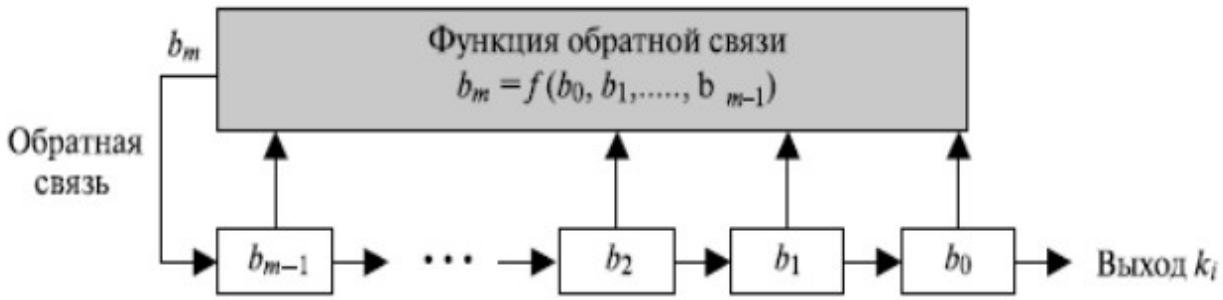

Рисунок 16 − Регистр сдвига с нелинейной обратной связью

Недостаток сдвигового регистра с нелинейными обратными связями - сложность его обобщенных теоретических исследований. Такие генераторы не могут предсказать период, а также другие важнейшие свойства последовательности выхода. Обстоятельство упрощает его использование на практике и упрощает ее применение. Для более широкого применения

используется комбинация нескольких РСЛОС, выходы которые объединяются в нелинейную функцию.

3.1.7 Комбинирующие генераторы с фиксированным тактированием

Увеличить сложность прогнозирования выходной последовательности генератора случайных чисел можно, подключив несколько РСЛОС в каскад. В более простых случаях правило можно выразить как линейную, так и нелинейную функцию, как видно на рис. 17.

Двумя простейшими реализациями комбинированного генератора являются генератор Геффе и пороговый генератор.

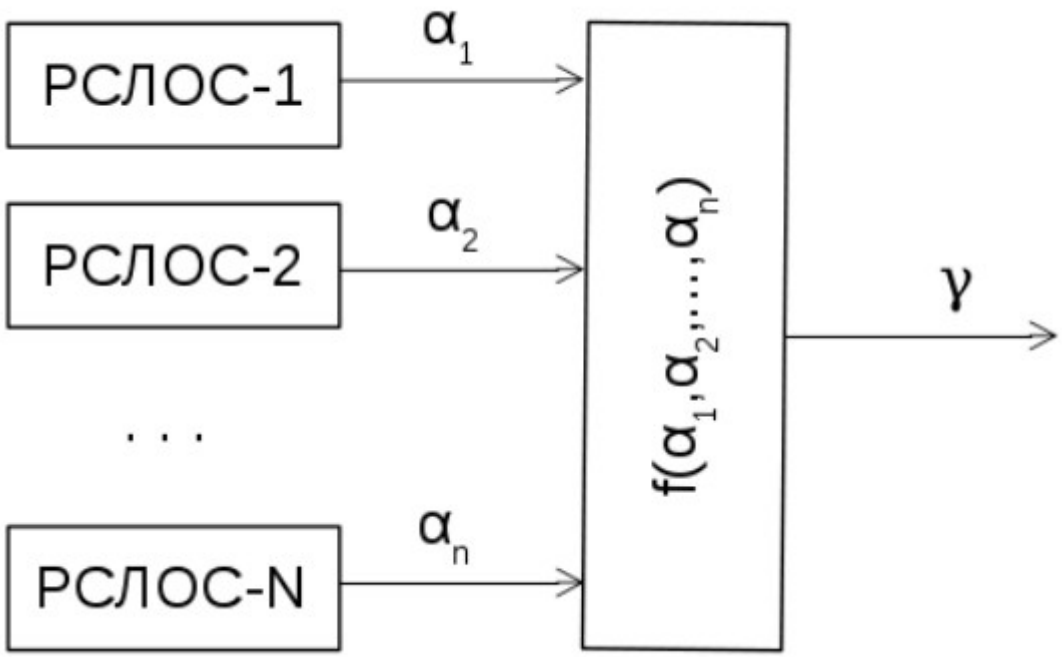

Рисунок 17 – Комбинирующий генератор

Генератор Геффе использует комбинацию из трех ЛРС. В данном генераторе ЛРС-1 и ЛРС-2 генерируют, а ЛРС-3 управляют. Комбинирующая функция $f(x_1, x_2, x_3)$ имеет вид:

$$f(x_1, x_2, x_3) = x_1 x_3 \oplus x_2 (x_3 \oplus 1)$$

Функция нелинейна. Размер сдвигового регистра с учетом, что длина сдвигового регистра является четным простым числом псевдослучайных последовательностей:

$$L=\left(2^{L_1}-1\right)\left(2^{L_2}-1\right)\left(2^{L_3}-1\right),$$

где L_i – длина i- ого регистра сдвига.

Генератор порога состоит из произвольного нечетного числа РСЛОС. За каждый такт генератора поступает бит, полученный в результате «голосования», если нулей на выходах сдвиговых регистров больше (на выходе выдается 0), то появляется 1, выходную функцию можно представить в следующем аналитическом виде:

$$f\left(\gamma_1,\gamma_2,\gamma_3\right)=\gamma_1\gamma_2+\gamma_1\gamma_3+\gamma_2\gamma_3,$$

где γ_i– выход i - ого регистра сдвига.

Рассматриваемый генератор, а также другие комбинированные генераторы подвергаются корреляционной атаке. Это позволяет одновременно определять состояния регистров сдвига в каскаде. Генератор Геффа характеризуется, в частности, тем, что 75% знаков выходной последовательности совпадают с выходом первого сдвигового регистра. Это дает возможность отсеять значения, не полученные от первого генератора, собрав определенное количество статистики. Этот процесс можно проследить и спрогнозировать состояние всего генератора. Генератор порогов также уязвим для этой атаки, но его сложность возрастает по мере увеличения числа сдвиговых регистров в каскаде.

Таким образом, комбинация генераторов является хорошим компромиссом между двумя или более сложными алгоритмами псевдослучайных чисел. Их можно использовать в криптографии.

3.1.8 Генераторы с переменным тактированием

Одним из способов разобраться с злоумышленником, пытаясь понять логию и прогнозировать поведение псевдослучайного генератора, является

включение генератора переменных тактовых сигналов в схему. Удаление выходного символа из сдвигового регистра или других операций происходит не при каждом тактовом импульсе, поступающем на генератор, как обычно, а при неизвестной логике злоумышленника. Таким образом, информация в генераторе может двигаться с задержкой и сам Генератор может не выдать символ на каждой тактовой импульс.

Такой генератор изготавливается по двум схемам: с внешней передачей или с самоуправлениями.

В случае управления внешним генератором включается управление сдвиговыми регистрами или автоматами. Первый применяется для определения момента снятия выходных символов из второй и последующего обработки или прямого передачи к входному генератору.

По схеме внешнего управления работает генератор «δ-шаг», а его сложный вариант «0-1 шаг» — генератор с усложненным алгоритмом. Также по схеме работает каскадный гольманский генератор и компрессионный генератор.

Если в двухступенчатой схеме управляющим автоматом является генератор фильтров на основе ЛРП длины n, а порождающим автоматом — ЛРП длины m (от одной из ячеек, гамма которых удалена), который выполняет каждый цикл в зависимости от знак управления τ или δ шагов, τ≠δ, то такой автомат называется генератором «δ-τ шагов» (при δ=0, τ=1 – генератором «стоп-вперед»). Генератор «шагов δ-τ» выполняется по рисунку 18. Статистические характеристики генерируемой последовательности зависят от генераторных параметров, например, «стоп вперед» подвергается атакам корреляции.

Усложнение генератора стоп-вперед, предложенное для улучшения статистических характеристик, - генератор перемежающего шага. Ее гамма формируется в виде суммы гаммы двух «стоп-вперёд» генераторов ЛРС-2 и ЛРС-3 длины м и р соответственно, а также общего управляющего ЛРС-1

длиной n. Если в i-ом такте ЛРС-2 продвигается, то ЛРС-3 простаивает, и наоборот. Ключом генератора являются начальные состояния всех ЛРС.

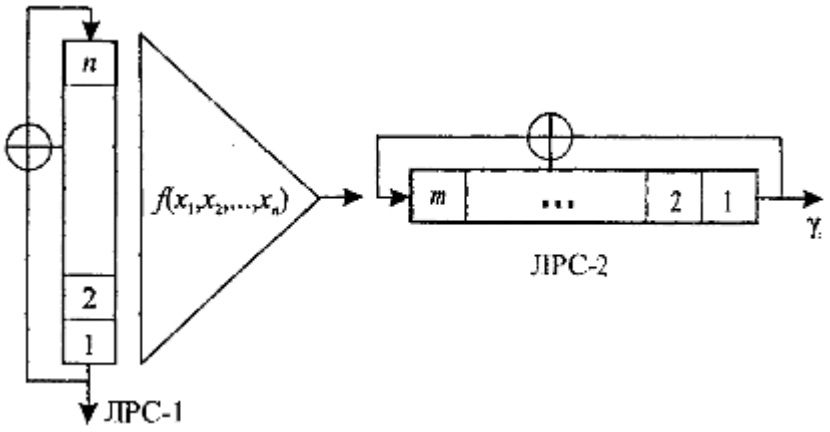

Рисунок 18 − Генератор «δ-τ шагов»

В отличие от схемы с регистром команды и генерации (или другого конечного автомата) в каскадном генераторе Гольмана, в другом, одна из команд управляется регистром движения. Такой генератор изображен на рисунке 19 (см. ниже). Если выход РСЛОС-1 в течение времени ti − 1− тактируется РСЛОС-2. Если выход РСЛОС-2 в течение времени ti – 1 − тактируется РСЛОС-3, и т.д. Последний выход РСЛОС − это выход генераторов. Если у всех ЛРС имеется длина n, а так же различные примитивные характеристические множества, период каскадной гаммы m достигает значения:

$$\left(2^n - 1\right)^m$$

При этом оценка линейной, сложность имеет значение:

$$n\left(2^n - 1\right)^{m-1}$$

В тоже время установлено корреляция выхода ЛРС-1 с гаммой m-каскадной генераторной гаммы, позволяющая хотя бы при m≤10, построить эффективную методическую последовательную вскрытию начального состояний ЛРС.

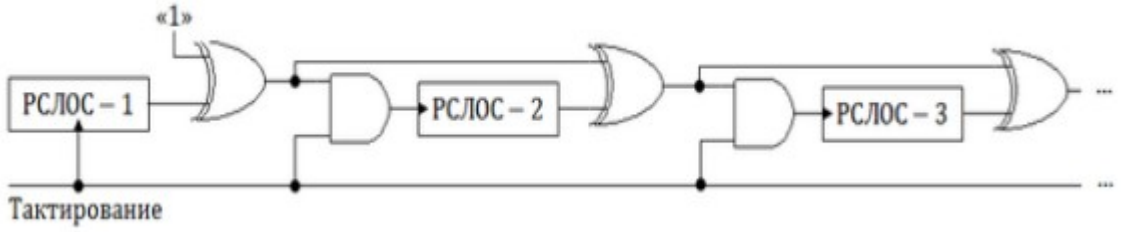

Рисунок 19 − Каскадный генератор Гольмана

Другой принцип движения − сжимающее устройство, построенное на основе параллельных работающих ЛРС−1 и ЛРС−2 с максимальным временем. Входные знаки гаммы снимаются с ячейки ЛРС−2, н но только в том случае, если управляющий знак ЛРС−1 равнен единице, в других тактах оба бита, генерируемые ЛРС−1 и ЛРС−2, не игнорируются.

Схема самоуправления представлена в генераторах типа генератора (δ, τ) − самоусечения и самосжимающий. Для этих генераторов используется лишь один регистр перемещения. В случае (δ,τ) самоусечения информация в ЛРС продвигается по δ шагам, если i−й знак гаммы равен 1и если неравенство информации в ЛРС продвигается по τ шагов (δ не равно τ). Период гаммы не превышает значения:

$$(3/4)(2^n - 1),$$

где n-длина регистра сдвига.

Самосжимающий генератор состоит из РСЛОС и двух ячеек, с которых снимается пара символов. Если первый цифра равна 0, то вторя не принимается. Эти генераторы могут выдать псевдослучайные последовательности, имеющие хорошие статистические свойства.

3.2 Анализ известных способов генерации псевдослучайных перестановок

В древности наиболее древний способ сокрытия информации был перемещение символов открытых сообщений по какому-то правилу, неизвестному постороннему человеку. Стойкость этого шифра определяется

количеством перемещенных элементов, расстоянием, на котором удаляются соседние знаки открытого кода. Шифр изменения может быть более надежен, если использовать комбинацию входящих сообщений и выходящих . В результате преобразования используется ранее используемая связка или ранее используемый блок открытых текстов.

Помимо перестановок символов открытого текста, они используются при скремблировании сигналов (в основном речи) и при построении криптографических протоколов. Далее в прикладных задачах информационной безопасности перестановки используются для обезличивания персональных данных операторами соответствующих систем. В перечисленных случаях применения перестановок возникает задача генерации их алгоритмами. Эти способы должны обеспечивать такие комбинированные комбинации, необходимые для защиты криптографической информации.

Генерация случайных перестановок основана на последовательности псевдослучайных чисел, т.е. статистические свойства случайности зависят от типа генератора псевдослучайных чисел. В качестве источника энтропии могут использоваться аппаратные датчики, т.е. истинно случайная величина, возникающая в результате измерения физической величины. Но второй способ не всегда удобен для практического применения.

Сначала рассматривается простейший способ генерации перемещений длины N. Он дает число, которое может использовано случайно, в случае применения истинной последовательности, вероятность возникновения того или иного элемента равна $\frac{1}{N}$. Представив отдельную переменную в виде элемента множества с всевозможными вариантами длины N, степень множеств A, то в данном случае вероятность получения конкретного перемещения равна вероятности получения конкретный результата. То есть в данном случае речь идет о равномерном распределении перестановок.

Простое решение генерации переменных, распределенных в равновероятной степени:

а) ввести переменные: i – устанавливается нуль счетчик цикла, P – пустое перемещение, N – длину генерируемой переменной;

б) если i не соответствует N, перейти на шаг в), а потом перейти на шаг ж);

в) сгенерировать случайное число r в интервале 0 – N - 1;

г) если число r уже получено раньше, то возвращаетесь к шагу в);

д) добавить новый элемент r в перестановку P;

е) вернуться к шагу б);

ж) конец алгоритма.

В данном алгоритме существует неопределенность задержки при выполнении шагов в) и г). Она быстро увеличивается с увеличением N. В данный алгоритм легко внести оптимизацию, выбрав массив невыбранных элементов. После этого на шаге в) можно создать случайный индекс, которая не превышает длины этого массива, она уменьшаться на 1 за каждую итерацию, а из массива извлекает элемент с этим индексом, а не получить абсолютное случайное значение от 0 до N - 1.

Другим алгоритмом, широко применяемым при создании случайных перестановок и весьма эффективным при реализации программы, является метод Фишера – Йетса тасовки. В этом алгоритме есть несколько видов. В качестве альтернативы предлагается вариант Дональда Кнута. Он обеспечивает простое соотношение между каждым из t! мощность множества перестановок из t – элементов возможных перемещений и последовательность чисел от c1 до ct – 1. Алгоритм состоит из следующего вида:

а) присвоить значение 0 переменной j;

б) равномерно распределенное между 0 и 1, сгенерировать случайное число U;

в) изменить число U случайным числом из интервала от 0 до j, и присвоить его k. Присвоить значению Xj значение Xk, а Xk – значение j;

г) увеличить j на 1. Если j меньше или равно t , возвратиться к шагу б).

В этом алгоритме предполагается получение новой случайной перестановки, но если задача будет перемешать какую-то заранее запрограммированную перестановку X случайно, то этот алгоритм выглядет следующим образом:

а) значение t присвоить j;

б) создать случайное число U, равномерно распределяемое между 0 и 1;

в) изменить число U случайным числом из интервала от 0 до j, и присвоить его k. Изменить значения Xk на Xj ;

г) уменьшить j на 1. Если j больше 1, возвратиться к шагу б).

Дополнительные требования могут быть предъявлены для генерируемых перестановок, кроме гарантии равновесия их возникновения во многих движениях в течение всей линейки, а также для генерируемых перестановок:

– элементы каждого нового изменения не должны находиться на одинаковых позициях, как и предыдущие изменения или изменения должны различаться на определенном количестве элементов;

– между элементами нового перемещения должны быть максимальное расстояние между их расположениями в предыдущей перестановке или расстояние между исходными элементами должно быть заданной.

Для соответствия этих требованиям необходимо правильно управлять генерируемыми перестановками и выбрать те, что соответствуют требованиям.

4 Обзор методов проверки статистических свойств генерируемых псевдослучайных числовых последовательностей

Тест псевдослучайной последовательности это совокупность методов определения меры близости, заданной псевдослучайной к случайному.

Генератор псевдослучайных числовых последовательностей в общем виде выдает в поток отдельные биты, которые затем преобразуются, чтобы получить случайное число методом срединного квадрата, который также может быть представлен двоичным способом. Из этих причин качество генератора оценивается статистическими свойствами потока бит и целого или дробного числа на исходе.

Чтобы проверить статистические свойства последовательности, используются тесты, их результаты представлены в виде графического изображения, что позволяет судить об отношении псевдослучайных или истинно-случайных последовательности. Графические тесты относятся к таким тестам, как построить гистограммы распределения последовательности элементов и проверить их неточность.

Тесты, которые выдают результаты графически, требуют от человека интерпретации. Таким образом, не может быть использовано для автоматического процесса проверок псевдослучайных цепочек и прочих «причинных» цепок. Пример результата графического тестирования показан на рисунке 20. Здесь по абсциссовной оси абсцисс отражены все сочетания из 3-х битов, а по ординате — разница частоты возникновения этого сочетания в последовательности 1/N, где N - общее число комбинаций, т.е. в этом случае 8. Численный статистический тест может автоматически сравнивать полученные результаты с определенным уровнем и на основании этого решать, успешно ли он прошел тот или иной тест. Чтобы получить более точную оценку, можно несколько раз провести тесты и рассчитывать вероятность того или иного результата в результате. В ряде случаев для того,

чтобы получить точность теста, необходимо увеличить число псевдослучайного значения, поступающего на вход теста.

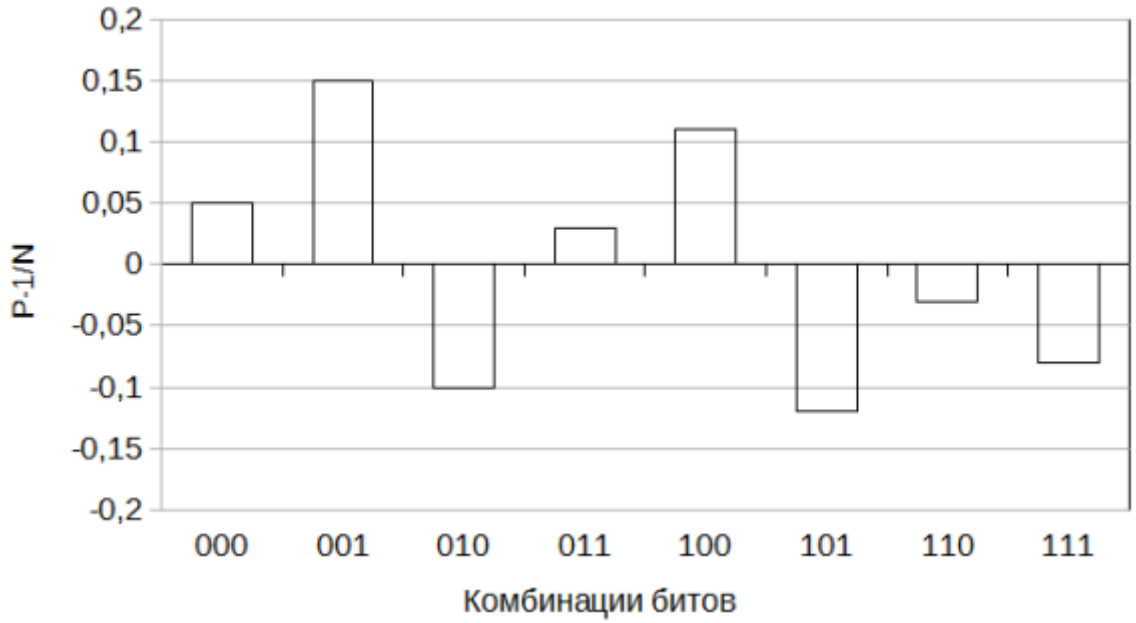

Рисунок 20 − Пример теста псевдослучайной последовательности, результаты которого выражены диаграммой

Далее представлено численное тестирование из набора тестов NIST. Исходные коды, содержащие эталонные тестовые функции, доступны бесплатно с сайта Национального института стандартов и технологий. Всего на сегодняшний день представлено 15 разных тестов. Вот несколько из них:

– тест частоты однобитного типа. Самый простой тест. Вычисляются частоты возникновения нулевых и единичных нулей в последовательности и исходя из полученного результата принимаются расчеты о том, какая степень их случайности.;

– тест на серию подряд идущий нуль или единицу. Говорят, что в настоящее время серию битов составляют только нули или единицы. Если слишком много таких сериалов, то его можно считать случайным.

В этом тест проверяет следующие условия;

– тест на спектральный анализ. При проведении тестирования псевдослучайных последовательностей используется дискретная

трансформация Фурье. Частотный спектр последовательности не должен явно иметь преобладающие амплитуды. Продолжение пиков говорит о периодической последовательности, которую можно просматривать в последовательном порядке. Это негативный фактор, определяющий степень случаев;

– тест серий. Тест считает количество последовательностей, перекрывающих битовые последовательности длины n, например, при n= 3, таких последовательностей 2^3 штук: 000, 001, 010, 011, 100, 101, 110, 111. В действительно случайном порядке вероятность возникновения каждой подобной последовательности достигает $K\frac{1}{N}$. Существенные колебания от этого показателя свидетельствуют о недостаточном качестве генерируемых последовательностей;

– тест на частичную сумму. Новые последовательности представляют собой число минус 1 и единицы — единицы. Таким образом, считается частичная сумма из н элементов, от 1 предусмотрены два режима: счет с окончания последовательности или с начала последовательности. Полученные значения сумм S_1, S_2, S_3 и так далее выбирают максимальную. Затем этот показатель применяется для того, чтобы оценить успешность прохождения тестирования.

На рисунке 21 показан набор тестов, который был использован в программном скремблере.

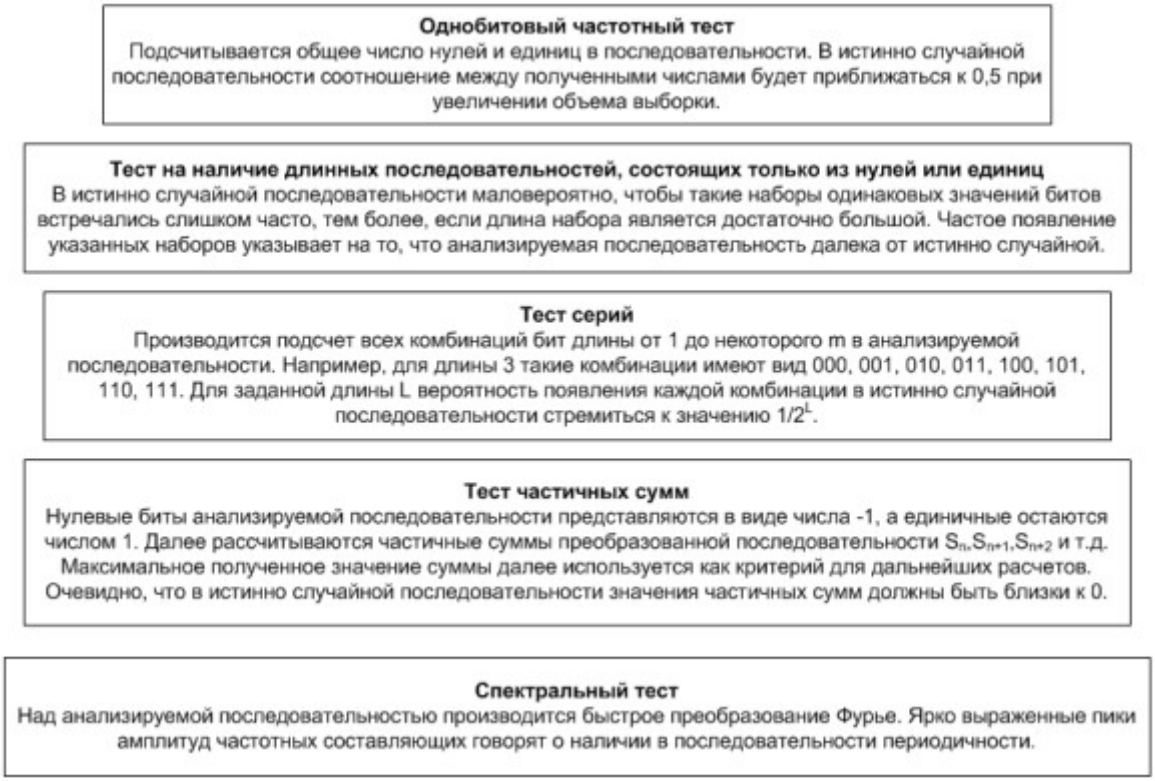

Рисунок 21 − Проверка статистических свойств псевдослучайной числовой последовательности.

Набор NIST предусматривает более сложный тест, однако успешность проведения вышеупомянутых тестов говорят уже о хорошем статистическом свойстве параллельного процесса. Наиболее сложные тесты – это тесты линейной сложности, универсальный метод статистического Маурера.

5 Программная реализация скремблера оцифрованного речевого сигнала

5.1 Общее описание программы и интерфейса пользователя

Программное обеспечение выполнено в виде графических приложений для компьютеров, работающих под операционной системой, поддерживаемой исполняемой средой Qt5 Windows, Linux и MacOS. На рис. 22 показано окно главного окна программы.

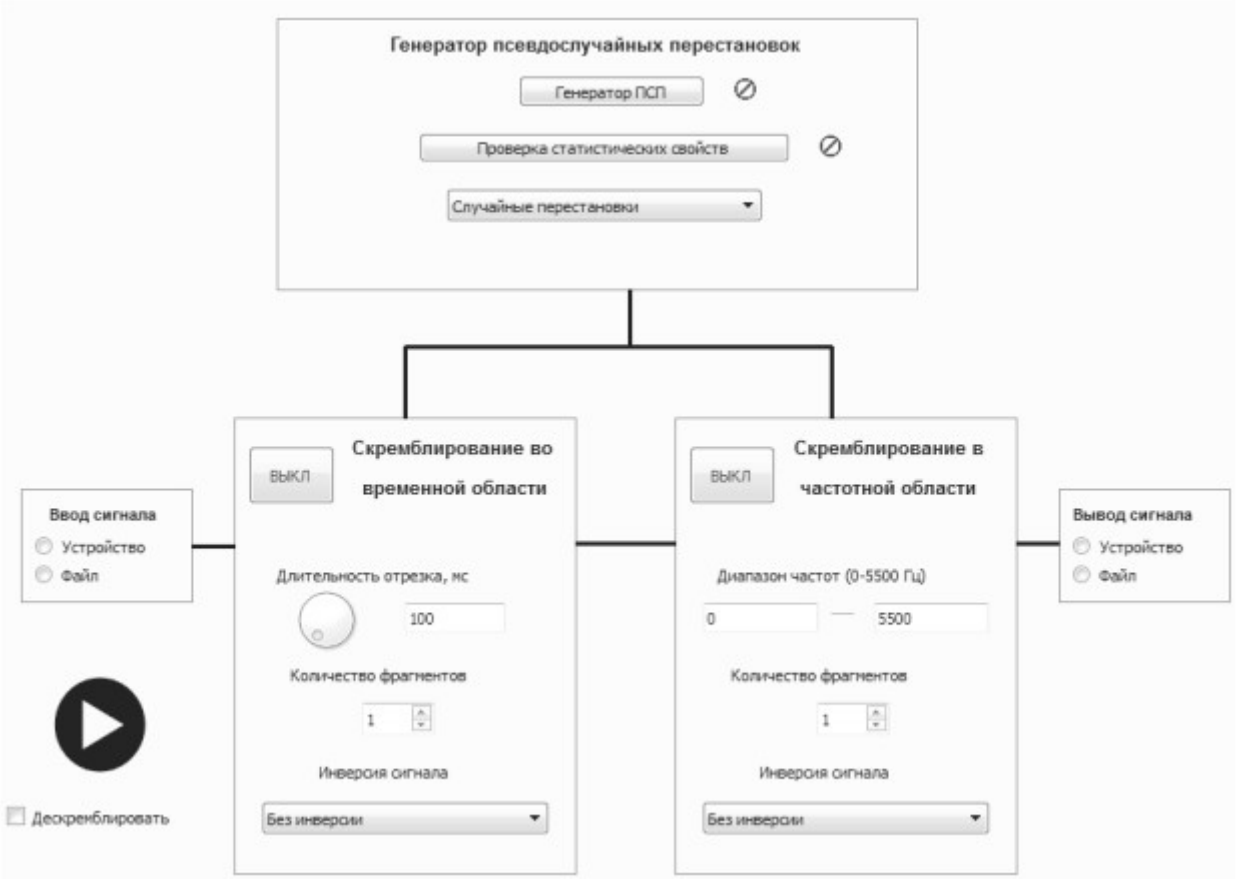

Рисунок 22 − Главное окно программы

В графическом интерфейсе реализованы элементы управления, разработанные в виде элементов:

– в разделах Ввод сигнала и Вывод сигнала вы можете выбрать источник входа файл формата WAV, или встроенную аудиосистему, например микрофон, чтобы выбрать источник входа;

– переключатель «Дескремблировать» позволяет задать режим работы программы (скремблирование или дескремблирование);

– секция « Скремблирование во временной области» позволяет задавать длину отрезка речевого сигнала, над которым производится скремблирование;

– Секция « Скремблирование в частотной области » позволяет определить границы спектров, для которых осуществляется скремблирование полной инверсией случайного фрагмента;

– секция «Генератор псевдослучайных перестановок» позволяет настраивать псевдослучайный генератор, проверять статистические характеристики полученной им последовательности, выбрать режим перемещения случайный или фиксированный, заданный пользователем;

– кнопка «Старт/Стоп», посредством которой происходит старт процесса генерации или дегенерации сигнала или остановка его процесса.

Весь код программного обеспечения на языке C++ состоит из набора специализированного класса. Реализованы следующие классы:

– в классе GuiScrambler описывается главное окно программного обеспечения и обеспечивается логическая связь с другими компонентами;

– класс IOManager управляет выводом и вывозом речевых сигналов. Он выполняет необходимые буферы, настройки устройств, файлы. В своем произведении используется в качестве вспомогательного класса аудиобуфера, который описывает циклические буферы, которые хранят отсчеты исходного и преобразованных сигналов;

– класс RandGenManager реализует генерирование псевдослучайного числа и перестановки. Для работы использует в качестве вспомогательного класса ShiftLine, описывающий линейный регистр с обратной связью;

– класс DigitalProcessing, реализующий прямой цифровой обработку сигналов. Для работы использует в качестве вспомогательного класса FIR фильтр, который реализует фильтр с конечно импульсным характеристикам;

– классы GuiCheck, GuiRNG, GuiSelectDev описывают графические диалоги, применяемые для того, чтобы проверить статистические свойства псевдослучайного генератора и задавать параметры такого устройства.

В программе может быть самостоятельное средство для того, чтобы обеспечить конфиденциальность речи и служит инструментом для оценки качества речи при различных характеристиках. Реализованные в программе алгоритмы можно перевести на более низкие языки, чтобы использовать их в устройствах портативного скремблирования.

Спектакль в целом рабочем порядке показан на рис. 23.



Рисунок 23 − Общий порядок работы скремблера

5.2 Ввод и вывод сигнала

Для ввода и вывода речевых сигналов используется класс IO Manager. Он выполняет следующую функцию:

– инициализация входных и выходных устройства или файла, подготовка к чтению или записи;

– реализация таймера для определения интервала времени, по которому нужно читать следующий отрезок речевого сигнала;

– буферизация ввода и вывода;

– переводить полученные отчеты речевым сигналом на цифровой блок и принимать их из выхода этого блока;

– завершить процесс ввода-вывода сигналов, закрыть входные и выходные устройства и файлы, освободить память, занятую временными изменениями, используемые при выполнении операций вывода-вывода.

В главном окне графического управления ввод параметров устройств и файлов происходит через главное окно управления, созданное в соответствующем классе. Название выбранного компьютера и путей файлов записаны переменными членами класса IOManager m_inFile, m_outFile, m_inDev и m_outDev.

Классный конструктор устанавливает аудиоданные формата: частоты дискретизации - 11025 Гц, частота отсчетов - 16-битный целое с чередованием бит LittleEndian, количество каналов – 1 моно. Данный формат сохраняется в переменной QAudioFormat, которая называется m_audioFormat.

Инициализация осуществляется в зависимости от выбранных входных и выходных устройств и файлов. Возможно, в конфигурации управления вводом и выводом четыре варианта:

1) исходные сигналы поступают с входа, преобразованные сигналы записываются в файл;

2) исходные сигналы поступают с входной машины, преобразованные сигналы поступают на выходную машины.;

3) исходные сигналы поступают из файла и преобразованные сигналы поступают на выходные устройства;

4) исходные сигналы поступают из файла и преобразованные сигналы записываются в файл.

Отчеты сигналов с выхода помещаются в входную аудио буферную систему, реализованную классом аудио буферной системы, которая является наследником класса QIODevice, методы читать и записывать которую

переопределены для работы буфера как циклического буфера.. Тот же буфер устанавливается перед сигналом и преобразуется в него преобразованный аудиоданными. Циклизация буфера требует постоянной передачи информации на вывод звука для того, чтобы избежать истощения внутренней буфера, которая может вызвать ошибки в выводе. Если циклический буфер не обладает уникальными данными для передачи аудиоустройства, то часть информации берется из начала данных и повторяется.

Если аудиосистема выбрана в качестве входного или выходного сигнала, то входная информация берется один раз в течение определенного промежутка времени, которое задается пользователю. Если необходимо скремблировать используя файлы входных и выходных файлов входных и выходных, заданные разряды сигнала берут раз в пять мс., не ожидая поступления данных В случае использования низкопроизводительного процессора и памяти, этот интервал должен быть увеличен, поскольку буфер входа и выхода может иссякнуть, поскольку обработка сигналов будет медленнее, чем новые.

Взаимодействие с входным и выходным устройствами осуществляется с классами, которые предоставляет набор библиотек Qt, т.е. классы QAudioInput, QAudioOutput, QAudioFormat, QAudioDeviceInfo. После указания названия устройства, формата аудио-данных и входной или выходной буферизации IODevice никакие дополнительные действия по буферированию не требуются.. В устройствах, представленных объектами соответствующего класса, можно самостоятельно считывать или записывать данные на буферы при необходимости. Этот режим называется push-режимом в отличие от push-режима, когда нужно вручную посылать данные в буфера через определенное время.

В таблицах 2 и 3 приведено описание класса. В таблице 2 описаны переменные класса, в таблице 3 – функции-члены класса.

Таблица 2 − Переменные класса IOManager

Наименование переменных	Описание переменных
enum IOState {Unknown, Dev, File}	Ввод или вывод не определен, устройство или файл
bool m_didOnceWait	Флаг, использующий одноразовую задержку вывода
bool isNowScrambling	Флаг, применяемый для того, чтобы проверить факт скремблирования в настоящее время
IOState m_inputState	Режим входа
IOState m_outputState	Режим выхода
quint32 m_chunkDuration	Длительность промежуток сигнала на скремблирование
quint32 m_chunkSize	Длина проволоки для крепления в байтах для крепления
QByteArray m_waveInBuf	Входный буфер массива байт с алгоритмом следования Little Endian
QByteArray m_waveOutBuf	Входной буфер массива байт с алгоритмом следования Little Endian
QFile m_inFile	Файл входа
QFile m_outFile	Файл выхода
QAudioInput* m_inDev	Указатель, по которому осуществляется доступ к входной аудиосистеме
QAudioOutput* m_outDev	Указатель доступа к выходным аудиоустройствам
QAudioFormat m_audioFormat	Формат аудио сигнала
QAudioDeviceInfo m_inDevInfo	Информация об устройстве входа
QAudioDeviceInfo m_outDevInfo	Информация об устройстве выхода
audiobuffer* m_inAudioBuffer	Указатель для входного циклического аудиобуфера
audiobuffer* m_outAudioBuffer	Указатель для выходного циклического аудиобуфера
QTimer* m_newChunkTimer со входа	Указатель таймера, который инициирует чтение сигнала

Таблица 3 − Функции-члены класса IOManger

Наименование функций	Наименование функций
IOManager(QObject *parent = nullptr)	Класс конструктора
bool prepareIO()	Инициализация устройств входа и выхода
bool seekWAVEdata()	Выбор позиции чтения входных WAVE-файлов, соответствующую началу звукового сигнала
bool writeWAVEheader()	Записать в выходном файле заголовок WAVE в соответствии с заданным форматом
void Finalize()	Завершение процесс ввода-вывода сигналов, корректно закрыть устройства и файлы
void ioError(QString desc) void	Сигнал Qt: отправьить сообщение о ошибке вывода вывода, указав причину
chunkReadyForScrambling(QByteArra y chunk)	Сигнал Qt: отправить сообщение, что сигнал получен от входа, и передайте его в обработку
void ChangeDev(QAudioDeviceInfo audioInfo, bool isIn);	Слот Qt: Обработка запроса о смене входа или выхода
void getMore();	Слот Qt: Обработка запроса о получении следующего сигнала с входа
void pushOut(QByteArray out);	Слот Qt: Обработка запроса на вывод отрезка обработки

В Приложении А приведено объявления класса IOManager на языке C++.

5.3 Генератор псевдослучайных чисел

При анализе плюсов и минусов различных псевдослучайных генераторных чисел было решено реализовать сжатие генератора, состоящего из двух регистров сдвига, в которых выходная битка одной определяет,

должна ли подавляться последняя битка на другой и подаваться на выход текущего цикла, рис. 24.

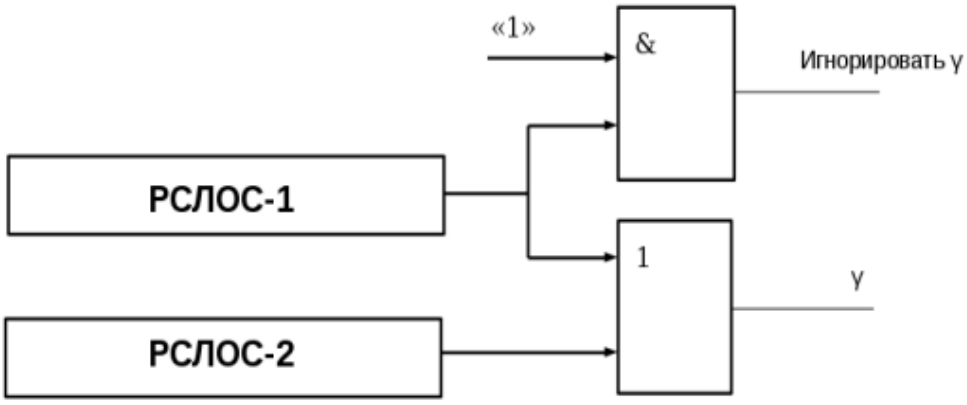

Рисунок 24 – Сжимающий генератор

ГПСЧ реализуется в классе C++ называемом RandGenManager, выполняющем основную задачу генерирования случайных целых чисел в заданной области. Чтобы выполнить эту функцию, класс содержит две сдвиговые регистры, которые представляют собой экземпляры других классов, созданных специально для описывания сдвиговых регистров: ShiftLine.

Для функционирования ГПСП надо сначала выбрать параметры сдвиговых регистров — длину, начальное заполнение и функцию линейной обратной связи. Эти параметры задаются через специальное диалоговое окно (см. рис. 25). Пользователь вводит характеристический полином, описывающий сдвиговый регистр в формате "x1+xa+xb...", введенная строка проверяются по соответствию формату регулярными выражением, а потом на основе данного полинома определяются длины сдвиговых регистров и определены места, где можно считывать значения битов, чтобы вычислить функцию обратного обращения.

Начальные интервалы выбираются пользователем из файла текста с двумя строками, состоящими из нуля и единицы. В первой строке определяется первый интервал первого записи, в второй строке - второй. Если для проверки работы программы нужно проверить, то программа

предусматривает возможность устанавливать пред заполнение в результате хэш – функции в текущем системном времени, в этом случае потеряется возможность шифрования в будущем из-за того, что значение ключа неизвестно для пользователя программы.

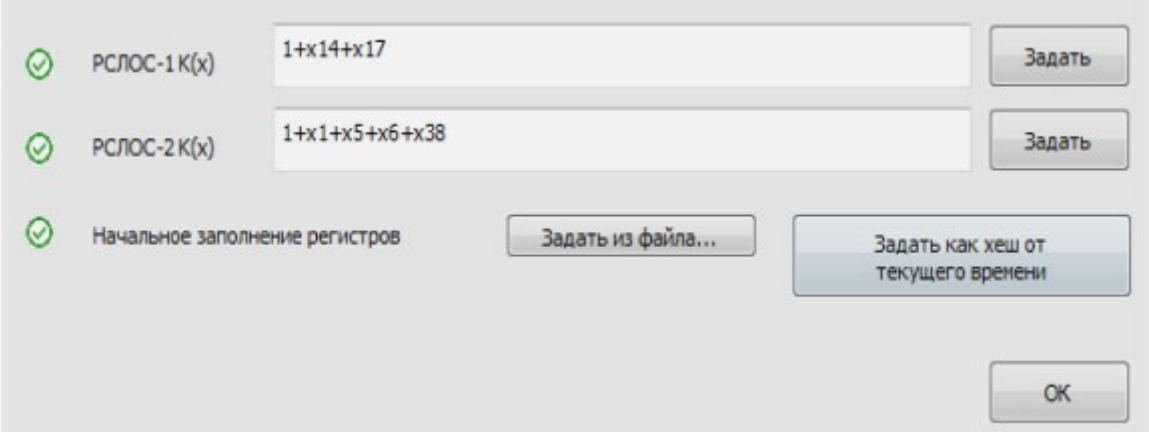

Рисунок 25 – Диалог выбора параметров генератора

GPSG может генерировать последовательность уникальных битов, а также 32-битные случайные числа. Если нужно сгенерировать число случайное из определенного диапазона половины диапазона от a до b, выполняются следующие действия:

а) путем последовательной генерации 32 отдельных битов получается 32-битное (без знаковое) число. Вероятность получения заданного 32-битного числа равна 1/232, учитывая, что вероятность генерации одного битового значения равна 0,5;

б) максимальное число 32-битного чисел, то есть 232, разделяется на количество случайных чисел, которые попадают в нужный диапазон и из которых следует получить число случайных чисел;

в), полученное число в шаге а), делится на число в шаге б), и полученная часть результата берется целая. Полученное значение, является требуемым числом, из заданного полуинтервала от a до b, если оно будет равно a, что может быть маловероятно, от него будет отнесена единица.

Внутренняя структура регистра сдвига представлена в виде вектора (QVector, эквивалентного стандартному векторному контейнеру C++),

содержащего логические значения. При изменении значений в регистрах в начале ячейки добавляется бит возврата, а потом последний элемент удаляется из вектора. Для этого метода требуется дополнительная манипуляция с памятью, чтобы изменить длину вектора, однако его легко выполнить и отложить. При необходимости эту операцию можно оптимизировать, так как чаще всего ее выполняет генератор.

Это число 32-битное формируется в последовательном применении операции побитовой или над данным числом и новой битой, полученной из регистра сдвигов, с последующей операцией побитовой сдвиги числа в левую сторону.

В таблицах 4 и 5 показаны данные и функции классов ShiftLine и RandGenManager.

Таблица 4 – Функции и переменные класса ShiftLine

Наименование переменной или метода	Описание переменной или метода
void Shift()	Производит изменение значений ячейков регистра в одной позиции вправо
void setXORs(QVector power)	Устанавливается ячейка, с которой происходит съем битовых значений
void setCells(QVector bits)	Устанавливает значения ячеек в регистре
QVector getCells()	Возвращает массив значений ячеек регистра
QVector getXORs	Возвращает позиции съемов значений ячеек регистра
size_t getLength()	Возвращает длину регистра сдвига
bool getBit() const	Возвращает значение крайнего правого бита регистра
QVector Cells	Массив, в котором содержатся значения каждой ячейки регистра
QVector XORs	Массив, содержащий позиции съёмов значений значений регистра ячеек
int length	Длина записи

Таблица 5 – Переменные и функции класса RandGenManager

Наименование переменных или функций	Описание переменных или функций
quint32 GetSingleBit()	Генерирует один бит случайный
quint32 GetUInt32(quint32 min, quint32 max)	В полуинтервале генерирует случайные числа от min до max
QVector GetPermTable(int n)	Возвращается случайная перестройка
ShiftLine SL1	Регистр сдвига 1
ShiftLine SL2	Регистр сдвига 2
quint32 m_max32unsigned	Число 2^{32}
void sendRandNums(QVector nums)	Сигнал Qt Object – отправка последовательности случайного чисела
void sendPermTable(QVector table)	Сигнал Qt Object – случайная перестановка
void Kx1_toChange(QVector powers)	Слот Qt Object – изменяет позиции съемки первых регистров
void Kx2_toChange(QVector powers)	Слот Qt Object – изменяет позиции съемки вторых регистров
void SLstate_toChange(QVector bits1,	Слот Qt Object – регистр сдвига
QVector bits2)	изменяет значения ячеек
void OnRequestRandNums(quint16 n)	Слот Qt Object – обрабатывает запросы для получения последовательности случайных чисел определенной длины
void OnRequestPermTable(quint16 n)	Слот Qt Object – обрабатывает запросы для получения случайной перестановки определенной длины

Объявления классов C++, которые реализуют генератор псевдослучайных чисел и генератор перестановок, приведены в Приложении Б.

5.4 Генератор псевдослучайных перестановок

Случайные изменения генерируются функциями GetPermTable(int n) класса RandGenManager, описанными в предыдущей части пояснительной записи. Как аргумент принимается длинная перестановка. Перестановка, является последовательностью чисел 0 - n, в том числе n, в котором все эти числа переставлены случайно.

Функция генерирования перестановки использует числа случайных значений, сгенерированные генератором сжатия. Новую перестановку заданной длины получают в результате алгоритма следующего:

а) создается массив unused, который содержит все числа в перестановке длины n;

б) Функции GetUInt32() генерирует случайное значение rand для полуинтервала от 0 до максимума. Значение максимум – длина массива unused;

в) к выходной перестановке добавляется число, стоящее на позиции rand в массиве неиспользованных чисел, то есть unused[rand]. Затем это число удаляется из массива без использования чисел и длина unused уменьшается на 1;

г) шаги б) и в) повторяются, элементов должно входить в выходную перестановку, то есть еще n минус 1 раз.

В случае получения случайных перестановок проверяется соответствие следующей точки зрения: в перестановки любые две соседние цифры не должны быть в прямом порядке, то есть должны быть отсутствуют такие комбинации числа, как «1,2», «5,6» и т.д. Это необходимо для повышения качества шифрования речи, так как могут быть распознаны два последовательных фрагмента речи, что приведет к нарушению ее конфиденциальности. Если такая комбинация чисел обнаруживается при проверке перестановки, то эта пара чисел меняет местами. Вам нужно проверить только один раз, потому что перестановка двух последовательных чисел гарантирует, что не будут созданы новые последовательные

комбинации, которые в противном случае пришлось бы находить и исправлять снова.

Поэтому на выходе псевдослучайного генератора появляются перестановки заданного размера, проверенные на то, что в них не было пар чисел в естественном порядке счетов.

5.5 Проверка статистических свойств псевдослучайных последовательностей числа

Для того чтобы проверить статистические свойства псевдослучайной числовой последовательности, использовались эталонные выполнения функций, входящих в пакет статистических испытаний, предоставленных Национальным институтом стандартов и технологий США (NIST). В этом пакете представлена программная реализация различных бинарных тестов, выполненных в виде кодов на C++. Все тесты принимают на вход определенное число случайной биты и производят над ними определённые вычисления, чтобы более точную оценку нужно взять несколько подобных отрезков последовательности битов, сформированных один генератором. В результате получается число P – пороговое значение для того, чтобы признать успешность проведения тестирования. Если P больше 0,01, то с вероятностью в 99% последовательности признают случайными, если P более 0,1, то доверительность – 90% и т. д. Доверительная вероятность задается заранее и в программе она установлена в 99%.

Для описания алгоритмов применяются следующие символы:

–n – длина исследуемых битовых последовательностей;

– ε– исследуемый битовый набор, определяемый как набор битов от ε_1 $до$ ε_n;

–$erfc(x)$ – функция ошибок;

–$igamc(a,b)$ – неполная функция гаммы.

Тест побитовой частоты выполняется функцией , выполняется по описанному ниже алгоритму.

Нули в последовательности бит преобразуются в минус 1 и остаются в минус 1. Тогда все числа преобразованных последовательностей складываются в одну сумму:

$$S_n = X_1 + X_2 + \cdots + X_n$$

$$X_i = 2\varepsilon_i - 1$$

К примеру, если ε имеет вид 10110, то:

$$S_n = 1 + (-1) + 1 + 1 + (-1) = 1$$

Рассчитывается значение S_{obs}, а после P:

$$S_n = \frac{|S_n|}{\sqrt{n}}$$

$$P = erfc(S_n / 2)$$

Проверка длинных строк, состоящих только из нулей или единиц, выполняется функцией Runs и выполняется по описанному ниже алгоритму.

Сначала рассчитывается число π:

$$\pi = \frac{\sum_j \varepsilon_j}{n}$$

Например, если ε *имеет вид* 10110001, *то* π рассчитывается таким образом:

$$\pi = \frac{4}{10} = \frac{2}{5}$$

То есть число единиц распределяется по длине битового строка.

Проверяет, выполнялся ли ранее побитовый тест частоты. Если нет или оно было выполнено и результат отрицательный, то P равно 0, алгоритм останавливается (тест не пройден).

Вычисляется $V_n(obc)$:

$$V_n(obc) = \sum_{k=1}^{n-1} r(k) + 1$$

При вычислении $V_n(obc)$ учитывается следующее:

$$\begin{cases} r(k) = 0, если \ \ \varepsilon_k = \varepsilon_{k+1} \\ r(k) = 1, если \ \ \varepsilon_k \neq \varepsilon_{k+1} \end{cases}$$

Например, в последовательности с видом 10110001, $V_n(obc)$ определяется так:

$$V_8(obc)=(1+1+0+1+0+0+1)+1=5$$

Далее вычисляется P:

$$P=erf\,c\left(\frac{\left|V_n(obc)-2n\pi(1-\pi)\right|}{2\sqrt{2n\pi}(1-\pi)}\right)$$

Полученное значение является числовым рейтингом прохождения данного теста.

Проверка частичных сумм реализована функцией CumulativeSums и выполняется по описанному ниже алгоритму.

Нули в последовательности бит преобразуются в минус 1 и единицы оставляют число 1. Новую последовательность определяют последовательность X:

$$X=x_1,x_2,\ldots,x_n$$

$$x_i=2\varepsilon_i-1$$

Например, при последовательности ε имеет вид 10110 – X будет состоять из чисел: 1, - 1, 1, 1, - 1.

По выбранному режиму прямая или обратная расчетная сумма вычисляется частично Sn элементами преобразованных последовательности, от первого до последнего элементов результаты, как правило, будут различны. К примеру, ε - последовательности типа 1011001, первые частичные суммы будут иметь прямой порядок:

$$S_1=1$$

$$S_2=S_1-1=0$$

$$S_3=S_2+1=1$$

$$S_4=S_3+1=2$$

Число z принимается как максимальное число, выбранное на многочисленных промежуточных результатах.

Значение P вычисляется по формуле:

$$P = 1 - \sum_{\frac{\left(\frac{-n}{z}+1\right)}{4}}^{\frac{\left(\frac{n}{z}-1\right)}{4}} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}} \right) \right] + \sum_{\frac{\left(\frac{-n}{z}+1\right)}{4}}^{\frac{\left(\frac{n}{z}-1\right)}{4}} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}} \right) \right],$$

где Ф(x) – является функцией распределения нормальных случайных величин.

Спектральные тесты для определения периодичности последовательности осуществляется функцией DiscreteFourierTransform и выполняются в соответствии с описанием ниже алгоритмом.

Нули последовательности бит преобразуются в минус 1 и единицы остаются числом 1. Новую последовательность определяют последовательность X:

$$X = x_1, x_2, \ldots, x_n$$

$$x_i = 2\varepsilon_i - 1$$

Например, если последовательность ε равна 10110, то X будет состоять из чисел: 1, - 1, 1, 1, - 1.

Для последовательности X используется прямое преобразование Фурье DFT(X), полученные спектры сохраняются в переменной S.

Вычисление последовательности M:

$$M = \left| S' \right|,$$

где S' − это первая часть элемента S.

Число T рассчитывается:

$$T = \sqrt{\left(\log \frac{1}{0{,}05} \right) n}$$

Предполагая, что рассматриваемая последовательность битов действительно случайна, 95% значений M не должны превышать это число. Далее вычисляются числа N_0 и N_1. Первое число — $0{,}95\frac{n}{2}$, а второе — количество значений в M, которые меньше T.

Рассчитывается переменная d:

$$d = \frac{(N_1 - N_0)}{0{,}25 \cdot 0{,}95 \cdot 0{,}05}$$

А затем рассчитывается число P:

$$P = erfc\left(\frac{|d|}{\sqrt{2}}\right)$$

Тест на подсчет перекрывающихся серий определенной длины реализован функцией Serial и выполняется по описанному ниже алгоритму.

Первые m - 1 бит, взятые с начала, добавляются к исходной последовательности ε. Число m — это длина максимальной тестируемой серии, т. е. последовательности из m битов.

Прямой подсчет производится при встрече некоторых значений рядов длины m, m - 1 и m - 2. Ряды могут перекрываться. Например, в последовательности 10110010101 имеются следующие значения ряда длины 3 в заданном числе: v_{000} - 0, v_{001} - 1, v_{010} - 2, v_{011} - 1, v_{100} - 1 v_{101} - 3, v_{110} - 1, v_{111} - 0.

Вычисление трех значений:

$$\psi_m^2 = \frac{2^m}{n}\sum_{i_1\ldots i_m} v_{i_1\ldots i_m}^2 - n$$

$$\psi_{m-1}^2 = \frac{2^m}{n}\sum_{i_1\ldots i_{m-1}} v_{i_1\ldots i_m-1}^2 - n$$

$$\psi_{m-2}^2 = \frac{2^m}{n}\sum_{i_1\ldots i_{m-2}} v_{i_1\ldots i_{m-2}}^2 - n$$

Например, серии длины 3, и последовательность ε вида 10110010101 длиной 11, ψ_m^2 рассчитывается так:

$$\psi_3^2 = \frac{2^3}{11}(0+1+2+1+1+3+1+0) - 10 \approx -3{,}45$$

Рассчитываются значения $\nabla\psi_m^2$ и $\nabla^2\psi_m^2$:

$$\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2$$

$$\nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2$$

Вычисляются P_1 и P_2 по формулам:

$$P_1 = igamc\left(2^{m-2}, \nabla\psi_m^2\right)$$

$$P_2 = igamc\left(2^{m-3}, \nabla^2\psi_m^2\right)$$

Оба значения должны быть больше 0,01, чтобы пройти тест.

Пользователь может определить длину входной битовой строки, над которой выполняются вычисления, и количество этих строк через специальное графическое диалоговое окно (см. рис. 26).

В нем он может видеть результаты тестирования по каждому отдельному тесту. Если тест не пройден, значение P в таблице выделяется красным цветом, в противном случае — зеленым.

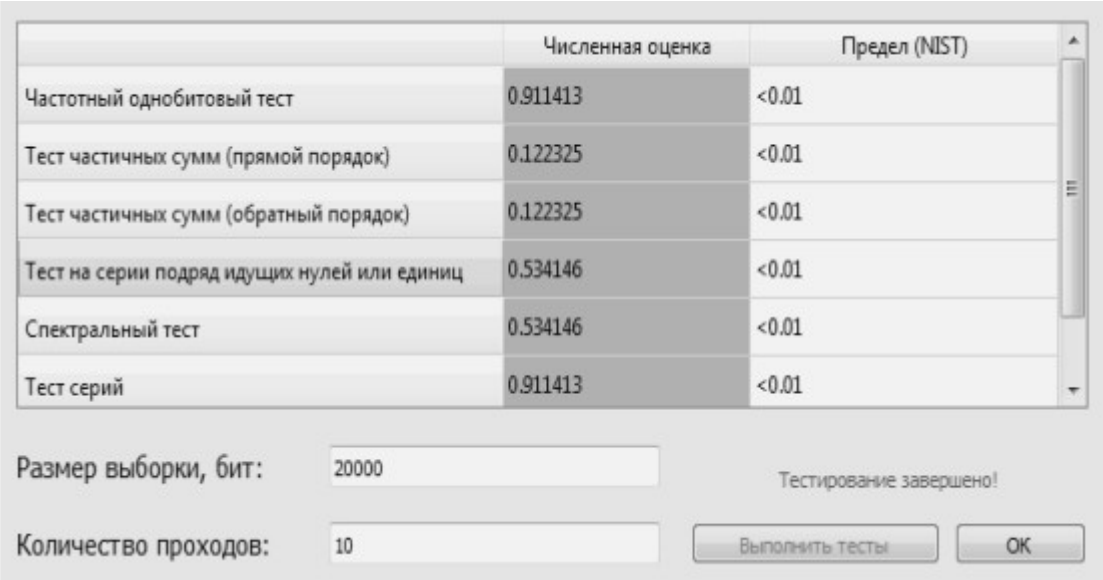

Рисунок 26 – Диалог проверки статистических свойств псевдослучайной последовательности

Программный модуль, разработанный с целью проверки статистики псевдослучайных последовательностей чисел, обеспечивает достоверность использования генератора и выбранных параметров для его использования для криптографических защит информации и блокирования речевого сигнала.

5.6 Цифровая обработка сигнала

Все функции для непосредственной обработки речевых сигналов, т.е. для их скремблирования, включены в отдельную категорию DigitalProcessing. Эти функции включают в себя:

– перестановка фрагмента сигнала в временную или частотную область;

– инверсия сигнала в временном или частотном районе;

– промежуточные преобразования форматов данных изменения порядка перехода байтов, изменение массива байтов в массив сигналов и т.д.

– быстрые дискретные преобразования Фурье прямо и обратно ;

– частотный фильтр сигнала.

Отсчеты сигналов речи в виде массивного байта с порядком движения Little Endian вносятся в функцию doScrambling , выполняющую их последовательную обработку в зависимости от заданных пользователем параметров программы. Скремблирование происходит сначала в области временной, а потом в области частотной. Если не выбраны временные и частотные скремблирования, то данные просто передаются на выход, не изменяя.

Учитывая, что помимо скремблирования программа должна уметь выполнять обратное преобразование - дескремблирование, то операции должны выполняться в прямом или обратном порядке. А поскольку в случае динамического перемешивания используются случайные наборы чисел (для реверсирования случайных фрагментов) или случайные перестановки, полученные от ГПСП, они предварительно генерируются, а затем берутся в прямом или обратном порядке. Реализация этого принципа необходима, так как не имеет значения, в каком порядке генерировались случайные числа и в каких операциях использовались те или иные множества. Генерация наборов случайных чисел или перестановок выполняется функцией genKeys. Полученные данные помещаются в массив m_keys.

Более специфичными функциями, чем doScrambling, являются функции doTimeScrambling и doFreqScrambling, которые эта функция вызывает и которые производят временное или частотное скремблирование соответственно.

Функции Permute, Inverse и InverseFreq переставляют и инвертируют данные во временной или частотной области. В первом случае обрабатываются записи сигналов, а во втором — комплексные значения, полученные в результате применения прямого дискретного преобразования Фурье к сегменту сигнала. Частотный диапазон для выполнения перестановок и инверсий задается пользователем программы.

Прямое преобразование Фурье выполняется с помощью функции БПФ, которая получает отсчеты сигнала в качестве входных данных и создает его частотное представление в виде комплексных чисел, отражающих амплитуду и фазу отдельных частотных компонентов сигнала. Обратное преобразование Фурье выполняется путем вызова функции iFFT. В этом случае возьмем спектр комплексного сопряжения S*(k), полученный при инверсии фазы каждого частотного компонента ДПФ, а затем применяем к полученным спектрам операцию прямой трансформации Фурье, а результат снова будет комплексным сопряжением, см. рисунок 27.

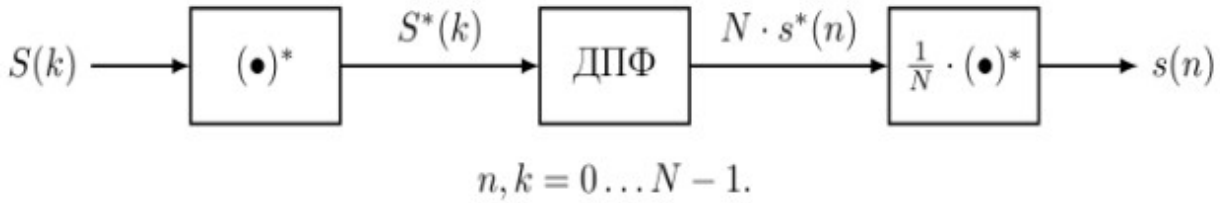

Рисунок 27 – Вычисление обратного ДПФ с использованием прямого ДПФ

Для корректной работы реализованных функций класса предусмотрена функция printVecToFile, перегружаемая для того, чтобы вывести сегмент сигнала, который представлен во временных или частотных областях, в файл текста.

Если производится частотная скремблирование, то после обратного преобразования сигнал Фурье перед выводом фильтруется. Например, при скреблировании частоты от 300 Гц до 3400 Гц эффективная частота голоса, амплитуды частот составляющих 0 – 300 Гц и 3400 Гц – 5000 Гц

уменьшается. Программная фильтрация реализована в виде отдельной FIR-класса.

Перед применением фильтра необходимо инициализировать его, рассчитывая коэффициенты по заданному частотному диапазону. Фильтр реализован в виде фильтров с конечным импульсным характеристикам и обладает длиной задержки в 80 сигнальных точках. Коэффициенты рассчитываются методом прямоугольного окна.

При расчете коэффициентов в программе используются следующие переменные:

– x– аргумент;

–Ω_c – частота среза (нормированная) для ФНЧ или ФВЧ;

– Ω_{low}, Ω_{high} - частота среза (нормированная) для полосового фильтра.

В фильтрах с низкими частотами используется формула, позволяющая рассчитать коэффициенты:

$$C_i = \Omega_c \, sinc\left(\Omega_c \, x\pi\right)$$

В высокочастотных фильтрах используются формулы по расчету коэффициентов: первая – для нечетных порядков, вторая – для четных порядков:

$$C_i = sinc\left(x\pi\right) - \Omega_c \, sinc\left(\Omega_c \, x\pi\right)$$

$$C_i = \cos\left(\Omega_c \, x\pi\right) / \pi / x + \cos\left(x\pi\right)$$

В полосовом фильтре используется формула, позволяющая рассчитать коэффициенты:

$$C_i = \left(\cos\left(\Omega_{low} \, x\pi\right) - \cos\left(\Omega_{high} \, x\pi\right)\right) / \pi / x$$

Программный фильтр КИХ работает прямой схемой по рисунку 28.

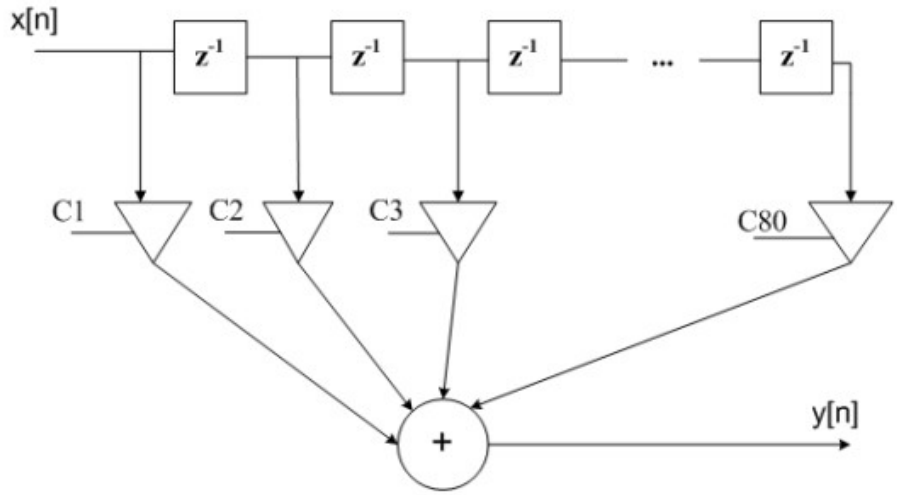

Рисунок 28 – КИХ-фильтр

В таблице 6 перечислены переменные класса Digital Processing, а в таблице 7 перечислены функции-члены этого класса. В таблице 8 перечислены переменные-члены и функции класса FIR.

Таблица 6 – Переменные класса DigitalProcessing, реализующего цифровую обработку сигнала

Название переменных	Описание переменных
bool m_isDescrambler	Флаг, указывает на режим работы программы дискремблера или скремблера
int m_freqFragmentsNum	Число фрагментов в области частотной
int m_timeFragmentsNum	Число фрагментов в области временной
int m_freqRangeMin	Нижние границы диапазона частоты, в которых происходит скремблирование
int m_freqRangeMax	Верхние границы диапазона частоты, в которых происходит скремблирование
bool m_isTimeScrambling;	Флаг – можно ли временно скремблировать
bool m_isFreqScrambling;	Флаг – можно ли частотно скремблировать
bool m_isFixedPerm;	Флаг используется для фиксированной перестановки или случайной перестановки
QVector m_fixedTimePermTable;	Содержит временную фиксированную перестановку, если ее задали
QVector m_fixedFreqPermTable;	Содержит частотную фиксированную перестановку, если ее задали

Продолжение таблицы 6

Название переменных	Описание переменных
inverseType m_timeInverse;	Инверсия во времени
inverseType m_freqInverse;	Инверсия по частоте
QVector m_input;	Входные отслеживания сигналов, подлежащих скремблированию
QByteArray m_output;	Байты выходных сигналов
QVector< QVector > m_keys;	Массив параметров случайных скремблирований /дескремблирований
QVector m_receivedRandNums;	Буфер хранения случайного числа для инверсии случайного фрагмента
QVector m_receivedPermTable;	Буфер со случайными перестановками

Таблица 7 – Функции-члены класса DigitalProcessing, реализующего цифровую обработку сигнала

Название функций	Описание функций
DigitalProcessing(QObject *parent = nullptr)	Класс конструктора
void printVecToFile(QVector vec, QString filename);	Вывод векторных элементов типа qint16 для файла отладочной функции
void printVecToFile(QVector> vec)	Вывести элементы типа compex в файл для отладочной функции
QVector ByteArrLEtoVec(QByteArray arr)	Преобразовать массив байт, организованный в порядке Little Endian, в массив сигналов
QByteArray VecToByteArrLE(QVector vec)	Преобразовать массив отсчетов в массив байт с порядком движения Little Endian
QVector< QVector > genKeys()	Сгенерирование случайных параметров, которые применяются для дескремблирования / скремблирования текущих отрезков сигнала
QVector<std::complex> FFT(QVector<std::complex> chunkComp)	Провести функцию - обертки последовательности Фурье
QVector getSingleKey(int fragments, int type)	Сгенерировать случайный набор чисел или перестановку

Продолжение таблицы 7

Название функций	Описание функций
QVector Inverse(QVector chunk, bool isFreq)	Провести инверсию во временных областях
QVector<std::complex> compConjugat(QVector> chunkComp)	Возвращает исходный массив комплексного сопряжения к заданному массиву комплексного порпяжения
QVector doTimeScrambling(QVector chunk)	Провести скремблирование во временных областях
QVector doFreqScrambling(QVector template QVector Permute(QVector chunk, bool isFreq) chunk)	Провести скремблирование в частотных областях
QVector<std::complex> InverseFreq(QVector<std::complex> chunk)	Провести инверсию в частотных областях
QVector iFFT(QVector<std::complex> chunkComp)	Проведение обратной трансформации Фурье с помощью алгоритма прямой быстрой трансформации Фурье
bool FFTAlgo(float *Rdat, float *Idat, int N, int LogN, int Ft_Flag)	Функция для непосредственного производства быстрого преобразования Фурье
QVector<std::complex> CopyHalfFFT(QVector<std::complex< float>> half)	Копировать левую половину массива комплексов чисел в правой половине
void requestRandNums(quint16 n)	Сигнал Qt Object: запросить набор из чисел случайные
void requestPermTable(quint16 n)	Сигнал Qt Object: запросить случайную перестановку
void doScrambling(QByteArray chunk)	Слот Qt Object: Произвести скремблирование
void receiveParams(procParams params)	Слот Qt Object: обработка запроса на получение скремблированых параметров
void receiveRandNums(QVector nums)	Слот Qt Object: обработка запроса на получение случайных наборов чисел
void receivePermTable(QVector table)	Слот Qt Object: обработка запроса на получение случайных перестановок

Таблица 8 – Методы е переменные класса FIR, реализующего фильтр КИХ

Наименование методов или переменных	Описание переменных или методов
int m_FIRlen	Длина фильтра КИХ - по умолчанию равна 80
bool m_isPopulated	Флаг, устанавливаемый в смысле истины, когда в линии задержки накопилось 80 отсчетов
QVector ks	Массив коэффициентов фильтра
QVector Cells	Массив отсчетов, содержащихся в линии фильтра задержки
enum TFIRPassTypes {firLPF, firHPF, firBPF, firALLPASS}	Перечисление типа фильтра ФНЧ, ФВЧ, полосового или пропускающего
FIR()	Класс конструктора
void SetFilter(int freq_min, int freq_max)	Установка полосы пропускания фильтров
QVector Process(QVector data)	Провести фильтрацию отрезка сигнала
qint16 getSample(qint16 x)	Рассчитать следующий отсчет отфильтрованного сигнала
void RectWinFIR(double *FirCoeff, int NumTaps, TFIRPassTypes PassType, double OmegaC, double BW)	Рассчитать коэффициент фильтра методом прямоугольного окна

Заключение

В выпускной квалификационной работе был проанализирован литературный обзор и создан программный шаблон оцифрованных речевых сигналов. Также было решено построение скремблера речевого сигнала цифровой обработки сигнала, построение псевдослучайных генераторов и проверку статистических характеристик, генерируемых последовательностями.

В ходе обзора программных алгоритмов генерации псевдослучайных последовательностей чисел и псевдослучайных перестановок, р были рассмотрены алгоритмы такие, как средней квадратной метод сравнения, линейно-полиномиальные методы сравнивания, аддитивных генераторов, генераторов на основе линейно-нелинейных регистров, генераторов на основе линейно-нелинейных обратных связей, комбинаций генераторов постоянного и переменного времени. В числе алгоритмов псевдослучайной генерации рассматриваются тривиальные алгоритмы, алгоритм Дональда Кнута, алгоритм Фишера-Йейта.

Также были проанализированы методики проверки статистической псевдослучайной битовой строки на близость к настоящей вероятности. Рассмотрены разные тестовые наборы и выбраны несколько тестовых наборов из набора NIST для выполнения программного скремблера.

Разработана программа для оцифровки речевых сигналов с помощью языка C++ и библиотеки Qt5. Реализованы такие подсистемы, как: подсистема вывода и вывода сигналов, подсистема генерирования псевдослучайного числа и перестановки и так далее.

Результаты выпускной квалифицированной работы можно использовать для того, чтобы защитить конфиденциальность речи при помощи разработанной программы и анализировать работу скриптов с различными характеристиками.

Список используемых источников

1. Петраков А.В. Основы практической защиты информации. М.: Радио и связь, 1999.-638 с.
2. Мясникова Е.Н. Объективное распознавание звуков речи. Л.: Энергия, 1967. -149 с.
3. Слеповичев И.И. Генераторы псевдослучайных чисел: учебное пособие: сайт / STUDYLIB URL: https://studylib.ru/doc/6222742/Генераторы псевдослучайных чисел (дата обращения: 15.04.2023).
4. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке С. М.: Диалектика, 2017. - 1040 с.
5. Кибзун А.И. Теория вероятностей и математическая статистика. Базовый курс с примерами и задачами / Горяинова Е.Р., Наумов А.В. - М.: ФИЗМАТЛИТ, 2007. - 235 с.
6. Дональд Э. Кнут Искусство программирования том 1 Основные алгоритмы третье издание; под общей редакцией докт. физ. мат. наук, проф. Козаченко Ю.В. - М.: ООО «И.Д. Вильямс», 2018. - 720 с.
7. Иванов М.А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / Чугунков И.В. - М.: КУДИЦ-ОБРАЗ, 2003. - 240 с.
8. Лайонс Р. Цифровая обработка сигналов. 2-е издание. Пер. с англ. под редакцией Бритова А.А. - М.: ООО «Бином-Пресс», 2006 . - 656 с.
9. Многофункциональный синтез СПД. Скремблирование и дескремблирование: сайт / Официальный сайт СанктПетербурского государственного университета. URL: http://opds.spbsut.ru/data/_uploaded/mu/siupd/lections/05.pdf/ (дата обращения: 20.04.2023).
10. Канал тональной частоты: сайт / Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Канал тотальной частоты/ (дата обращения: 22.04.2023).
11. Белова И.М. Компьютерное моделирование: учебно-методическое пособие. - М: МГИУ, 2007. - 81 с., Русский.
12. PIKE: сайт / Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/PIKE/ (дата обращения: 26.04.2023).
13. Токарева Н.Н. Симметричная криптография. Краткий курс: учебное пособие / Новосиб. гос. ун-т. Новосибирск, 2012. - 234 с.
14. eSTREAM: сайт / Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/ESTREAM/ (дата обращения: 29.04.2023).
15. Фомичев В.М. Дискретная математика и криптология : курс лекций онлайн. М.:ДИАЛОГ. МИФИ, 2003. - 400 с..
16. Тестирование псевдослучайных последовательностей: сайт / Википедия - свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Тестирование псевдослучайных последовательностей/ (дата обращения: 1.05.2023).

17. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications: National Institute of Standards and Technology. URL: https://dl.acm.org/doi/pdf/10.5555/2206233/(дата обращения: 5.05.2023).

Приложение А

Объявление класса IOManager на языке C++, реализующий ввод и вывод сигнала

```
/* КЛАСС РЕАЛИЗУЮЩИЙ ВВОД И ВЫВОД СИГНАЛА */
class IOManager : public QObject
{
    Q_OBJECT
public:
    //Конструктор
    explicit IOManager(QObject *parent = nullptr);

    //Подготовить устройства ввода/вывода и файлов
    bool prepareIO();

    bool seekWAVEdata() ; //Перейти к отчетам сигнала в WAVE-файле
    bool writeWAVEheader(); Записать заголовок WAVE-файла

    void Finalize(); // Прекратить операции ввода/вывода

    enum IOstate {Unknown, Dev, File}; //Режим ввода/вывода
    bool m_didOnceWait; //Флаг для первоначальной жадержки
    bool isNowScrambling;//Флаг, обозначающий процесс скремблирования
    IOstate m_inputState, m_outputState; //Режим ввода/вывода
    //Величина фрагмента для скремблирования (отсчеты 16 bit)
    quint32 m_chunkDuration;
    quint32 m_chunkSize;
```

```
        //Буферизация файлового ввода/вывода
        QByteArray m_waveInBuf;
        QByteArray m_waveOutBuf;
        //Чтение/запись WAVE файлов
        QFile m_inFile;
        QFile m_outFile;
        //Входные и выходные аудиоустройства
        QAudioInput* m_inDev;
        QAudioOutput* m_outDev;
        //Формат сигнала и информация об устройствах
        QAudioFormat m_audioFormat;
        QAudioDeviceInfo m_inDevInfo;
        QAudioDeviceInfo m_outDevInfo;
        //Циклические буферы на входе и выходе
        audiobuffer* m_inAudioBuffer;
        audiobuffer* m_outAudioBuffer;
        //Таймер обращения ко входу
        QTimer* m_newChunkTimer;
private:
signals:
        void ioError(QString desc); //Отладка – ошибка ввода/вывода
        //Отправить отрезок сигнала для скремблирования
        void chunkReadyForScrambling(QByteArray chunk);
public slots:
        //Изменить входное или выходное устройство
        void ChangeDev(QAudioDeviceInfo audioInfo, bool isIn);
        //Прочесть следующий отрезок сигнала на выход
```

```
        void getMore();
        //Послать преобразованный отрезок сигнала на выход
        void pushOut(QByteArray out);
};
```

Приложение Б

Объявления классов на языке C++, реализующих генератор псевдослучайных чисел и перестановок

```
/* КЛАСС РЕАЛИЗУЮЩИЙ РЕГИСТР СДВИГА С ЛИНЕЙНОЙ ОБРАТНОЙ СВЯЗЬЮ */
class ShiftLine{
public:
    ShiftLine(); //Конструктор
    void Shift(); //Произвести сдвиг значений ячеек
    void setXORs(QVector<int>powers){ //Установить позиции съема
        XORs = powers;
    }
    void setCells(QVector<bool>bits){            //Установить значения ячеек
        Cells = bits;
    }
    QVector<bool>getCells(){ return Cells; } //Возвращает значения ячеек
    QVector<int>getXORs(){ return XORs; }  //Возвращает позиции съема
    size_t getLength(){ return length; }        //Возвращает длину регистра
    //Возвращает крайний правый бит регистра

    bool getBit() const { return Cells[length-1]; }
private:
    QVector<bool> Cells; //Значения ячеек регистра
    QVector<int> XORs; //Значения мест съема
    int length; //Длина регистра
};
```

```
/* КЛАСС РЕАЛИЗУЮЩИЙ ГЕНЕРАТОР ПСП И ПЕРЕСТАНОВОК НА ОСНОВЕ СЖИМАЮЩЕГО ГЕНЕРАТОРА ИЗ ДВУХ РСЛОС */

class RandGenManager : public QObject

{

    Q_OBJECT

public:

    explicit RandGenManager(QObject *parent = nullptr);


    quint32 GetSingleBit(); //Сгенерировать случайную перестановку

    //Сгенерировать случайное число в полуинтервале [min,max)

    Quint32 GetUInt(quint32 min, quint32 max);

    QVector<int>  GetPermTable(int  n);  //Сгенерировать случайную перестановку


    ShiftLine SL1; //Регистр сдвига 1

    ShiftLine SL2; // Регистр сдвига 2

private:

    quint32 m_max32unsigned; //2^32

signals:

    void sendRandNums(QVector<int> nums); //Отправить случайные числа

    void sendPermTable(QVector<int>  table);  //Отправить случайную перестановку

public slots:

    void Kx1_toChange(QVector<int>powers); //Изменить места съема РСЛОС1

    void Kx2_toChange(QVector<int>powers); //Изменить места съема ЗСЛОС2

    //Изменить состояния РСЛОС 1 и 2

    void SLstate_toChange(QVector<bool>bits1, QVector<bool>bits2);
```

```
    //Обработка запроса на генерацию случайных чисел
    void OnRequestRandNums(quint16 n);
    //Обработка запроса на генерацию случайной перестановки
    void OnRequestPermTable(quint16 n);
};
```