Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный
университет информатики и
радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Программирование на языках
высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему
«Шифратор файлов»

Студент С. С. Черняк

Руководитель Е. В. Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

(подпись)

2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Черняку Станиславу Сергеевичу*
Тема проекта *Шифратор файлов*

3. Исходные данные к проекту: example.txt (пробный файл формата .txt)

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

1. Лист задания.
2. Введение.
3. Обзор литературы.
4. Функциональное проектирование.
4.1. Структура входных и выходных данных.
4.2. Разработка диаграммы классов.
4.3. Описание классов.
5. Разработка программных модулей.
5.1. Разработка схем алгоритмов.
5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов)
6. Результаты работы.
7. Заключение
8. Литература
9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Диаграмма классов.
2. Схема алгоритма метода encryptECB
3. Схема алгоритма метода decryptECB

6. Консультант по проекту (с обозначением разделов проекта) Е.В. Богдан

7. Дата выдачи задания *15.09.2023г*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %

разделы 2, 3 – 10 %;

разделы 4 к –20 %;

разделы 5 к – 35 %;

раздел 6,7,8 – 5 %;

раздел 9 к – 5%

оформление пояснительной записки и графического материала к 15.12.22 – 10 %

Защита курсового проекта с 21.12 по 28.12.23г

РУКОВОДИТЕЛЬ_____ Е. В. Богдан

(подпись)

Задание принял к исполнению_____ С.С.Черняк

(дата и подпись студента)

СОДЕРЖАНИЕ

1. Задание на курсовую работу

Овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++ ,овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++. Разработать программу “File Encryptor” с использование среды разработки Qt.

Qt предлагает ряд преимуществ, которые делают его популярным выбором среди разработчиков:

1. Простота использования: Qt имеет хорошо документированную структуру, что облегчает его освоение. Он имеет простую, однородную структуру, что упрощает процесс разработки.

2. Разработка GUI: Qt предоставляет широкий спектр компонентов для создания графических пользовательских интерфейсов. В частности, Qt Quick позволяет быстро и легко создавать интерфейс с использованием специального языка под названием QML.

3. Кроссплатформенность: Qt позволяет разработчикам создавать приложения, которые будут работать на различных платформах, включая настольные и мобильные устройства. Это делает Qt мощным инструментом для разработки кроссплатформенных приложений.

4. Гибкость: Qt может использоваться для создания широкого спектра приложений, от настольных и мобильных приложений до специализированного оборудования и встроенных систем.

В целом, использование Qt в C++ дает много преимуществ для разработчиков, которые хотят создавать кроссплатформенные приложения с графическим пользовательским интерфейсом. Это мощный и гибкий инструмент, который можно использовать в широком спектре приложений.

2. ВВЕДЕНИЕ

В современном мире обеспечение информационной безопасности представляет собой неотъемлемую часть цифровой эпохи, где защита конфиденциальных данных выходит на первый план. Одним из ключевых инструментов в этом контексте является шифрование файлов. Шифрование файлов представляет собой сложный процесс преобразования информации в формат, непригодный для восприятия, с последующим восстановлением исходных данных только при наличии специального ключа.

Целью данного курсового проекта является разработка файлового кодировщика на языке программирования C++. В рамках проекта осуществляется изучение и реализация различных методов шифрования, алгоритмов и структур данных, необходимых для эффективного решения задачи обеспечения безопасности. Это обеспечивает более глубокое понимание принципов симметричного шифрования, а также создание инструментов, способных эффективно защищать конфиденциальные данные от несанкционированного доступа.

Курсовой проект "Шифратор файлов на C++" не только предоставил теоретические знания, но и развил практические навыки в области шифрования, программирования и информационной безопасности. Полученные знания оказываются полезными не только для защиты личных данных, но и для работы с конфиденциальными корпоративными файлами,+ подчеркивая важность современных подходов к безопасности в цифровой эре.

В качестве результата курсового проекта были разработаны и реализованы файловый кодировщик на языке программирования C++, который позволяет эффективно защищать конфиденциальные данные от несанкционированного доступа. Я познакомился с практиками разработки программного обеспечения, связанными с шифрованием, и написали код, который демонстрирует глубокое понимание принципов работы разработанных алгоритмов.

Осуществленный курсовой проект "Шифратор файлов на C++" не только укрепил знания в области шифрования и информационной безопасности, но и привлек внимание к важности современных подходов к защите конфиденциальных данных в цифровой эре. Полученные знания и практические навыки смогут быть полезными для дальнейшего обучения и работы в области информационной безопасности и программирования.

3. ОБЗОР ЛИТЕРАТУРЫ

Литературные ресурсы, предоставленные различными авторами и институтами, предлагают обширное понимание алгоритма Advanced Encryption Standard (AES) и его внедрения в современные системы шифрования данных. Книги, такие как предоставляют теоретическую основу для понимания принципов работы AES.

Официальный стандарт AES, предоставленный Национальным институтом стандартов и технологии США, содержит официальные спецификации и рекомендации по реализации алгоритма. Это важный ресурс для тех, кто стремится понять стандартные протоколы и процедуры, связанные с AES.

Официальная документация: Сайт Qt: Официальный сайт Qt содержит обширную документацию, включая руководства, API—справочники, примеры кода и другие материалы.

Руководства и Введение в Qt:

Getting Started: Раздел "Getting Started" в документации обычно предоставляет информацию о том, как установить Qt, настроить среду разработки и создать простое приложение. Overview: Введение в фреймворк, его основные концепции и принципы.

Создание графического интерфейса: Qt Widgets: Информация о виджетах Qt, базовых элементах управления, таких как кнопки, поля ввода и другие. Qt Quick и QML: Документация о создании интерфейсов с использованием декларативного языка QML и фреймворка Qt Quick.

Работа с сетью и базами данных: Qt Network: Инструменты для работы с сетью, включая HTTP—запросы, сокеты и другие. Qt SQL: Информация о работе с базами данных, включая поддержку различных СУБД.

Многозадачность и Параллелизм: Qt Concurrency: Раздел, посвященный поддержке многозадачности и параллелизма в Qt.

Межплатформенная разработка: Platform Notes: Рекомендации и особенности для кроссплатформенной разработки на разных операционных системах. Deployment: Инструкции по развертыванию Qt—приложений на различных платформах.

Примеры кода и Учебные проекты:Qt Examples: Обширный набор примеров кода для различных компонентов Qt.Qt Tutorials: Учебные проекты и туториалы, позволяющие освоить различные аспекты фреймворка.

Обновления и Дополнительные ресурсы: Блог Qt: Официальный блог с новостями, статьями и обновлениями от команды разработчиков. Дополнительные ресурсы: Дополнительные материалы, такие как видеоуроки, вебинары и другие образовательные ресурсы. Qt предоставляет обширные средства для создания высококачественных приложений, и его документация является важным ресурсом для разработчиков, стремящихся освоить этот фреймворк.

3.1 РАССМОТРЕНИЕ МЕТОДОВ И АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ

3.1.1. Симметричное шифрование

Существующие методы симметричного шифрования представляют разнообразные подходы к обеспечению безопасности и эффективности данных. Один из наиболее распространенных методов — Advanced Encryption Standard (AES). Расширенный стандарт шифрования успешно применяется в силу своей высокой степени защиты и эффективности. Данный алгоритм работает с блоками данных, обеспечивая надежное шифрование. В данном контексте мы рассмотрим не только реализацию AES в режиме ECB (Electronic Codebook), но и обратим внимание на процесс шифрования и дешифрования данных. Отметим также, что для генерации безопасных хэш-сумм ключей шифрования применяется хэш-функция, например SHA-256. Это дополнительный уровень безопасности, который обеспечивается при обработке данных.

3.1.2. Хэширование

Одним из важных элементов в контексте решения поставленной задачи является применение алгоритма SHA256 (Secure Hash Algorithm 256-bit). Хэш в данном случае используется для генерации уникальных и непредсказуемых значений, которые служат ключами для алгоритма AES в режиме ECB. Рассмотрим более детально интеграцию SHA-256 в процесс генерации и обновления симметричных ключей шифрования.

3.1.3. Управление ключами

Процессы генерации и хранения ключей представляют собой важный аспект симметричного шифрования. Разработка методики безопасной генерации и хранения секретных ключей, необходимых для эффективного шифрования и расшифровки файлов, является ключевым компонентом в обеспечении безопасности данных.

3.1.4. Интерфейс пользователя

Создание пользовательского интерфейса становится неотъемлемой частью решения задачи. Проектирование простого и интуитивно понятного интерфейса, который позволяет пользователям выбирать файлы для шифрования и устанавливать параметры шифрования, играет важную роль в обеспечении удобства использования разработанного инструмента.

4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

4.1 Структура входных и выходных данных

Для решения задачи был выбран язык программирования C++ и методология объектно-ориентированного программирования. В процессе разработки программы были использованы различные возможности языка C++, которые будут описаны ниже.

4.1.1 Входные данные:

- Файл, который требуется зашифровать.
- Пароль (который будет случайно сгенерирован) для шифрования файла.

4.1.2 Выходные данные:

- Зашифрованный файл в нечитаемом формате.
- Пароль для каждого файла (должен быть сохранен в безопасном месте).

4.2. Разработка диаграммы классов

Диаграмма классов данной работы показана в приложении А.

4.3. Описание классов.

Для создания программы шифрования и дешифрования файлов с использованием метода AES в C++ с интерфейсом в Qt, вы можете создать следующую структуру классов:

4.3.1 Класс MainWindow

MainWindow — класс qtWidget являющий основным окном приложения.

Описание полей класса:

- `QString file_path` — путь к файлу который мы собираемся шифровать/расшифровывать.
- `bool isPasswordVisible` — булевая переменная, возвращающая скрыт ли пароль для пользователя в интерфейсе или нет.

Описание методов:

- `void updateButtonText()` — Метод класса MainWindow для обновления текста на кнопке в зависимости от расширения файла.
- `MainWindow(QWidget *parent)` — Конструктор класса MainWindow
- `~MainWindow()` — Деструктор класса MainWindow.
- `void on_pushButton_file_clicked()` — Метод класса MainWindow, вызываемый при нажатии кнопки выбора файла.
- `void password_operations()` — Метод класса MainWindow для обработки операций с паролями.
- `void on_lineEdit_password_textChanged(const QString text)` — Метод класса MainWindow, вызываемый при изменении текста в поле ввода пароля.
- `void check_password()` — Метод класса MainWindow для проверки совпадения паролей и окрашивания полей в зависимости от результата.
- `void encryptFile()` — Метод класса MainWindow для шифрования файла.
- `void copyFile(const QString &sourceFilePath, const QString &destFilePath)` — Метод класса MainWindow для копирования файла.
- `void deleteFile(const QString &filePath)` — Метод класса MainWindow для удаления файла.
- `void decryptFile()` — Метод класса MainWindow для расшифрования файла.
- `void on_pushButton_clicked()` — Метод класса MainWindow, вызываемый при нажатии основной кнопки.

4.3.2 Класс UI_create_password

UI_create_password — класс qtWidget являющийся окном, вызываемым при создании случайного пароля.

Описание полей класса:

- `bool isLower` — булевая переменная, возвращающая будет ли пароль состоять из символов нижнего регистра.
- `bool isUpper` — булевая переменная, возвращающая будет ли пароль состоять из символов верхнего регистра.
- `bool isNumber` — булевая переменная, возвращающая будет ли пароль состоять из цифр.
- `bool isSymbol` — булевая переменная, возвращающая будет ли пароль состоять из символов (таких как !@#$%^&*).
- `bool isCopy` — булевая переменная, возвращающая копировать ли пароль в буфер.

Описание методов класса:

- `void on_horizontalSlider_actionTriggered(int action)` — Метод класса create_password, вызываемый при изменении положения слайдера.
- `void on_pushButton_ok_clicked()` — Метод класса create_password, вызываемый при нажатии кнопки "OK".

4.3.3 Класс Password

Password — класс реализующий создание пароля из случайных символов

Описание методов класса:

- `Password()` — Конструктор класса Password.
- `QString createPassword(int length, bool isUpper, bool isLower, bool isNumbers, bool isSymbols, bool isCopy)` — Метод класса Password для создания пароля.

4.3.4 Класс SHA256

SHA256 — класс реализующий хеширование пароля, методом SHA256.

Описание полей класса:

- `uint32_t H[8]` — Инициализация переменных хеш-значения.

Описание методов класса:

- `std::string preprocess(const std::string& input)` — Метод класса SHA256 для предварительной обработки входных данных.
- `void processBlock(const uint8_t* block)` — Метод класса SHA256 для обработки блока данных.
- `std::string hash(const std::string& input)` — Метод класса SHA256 для вычисления хеша строки.

4.3.5 Класс AES

AES — класс, реализующий шифрование данных нашего файла методом AES Electronic Codebook

Описание полей класса:

- `static constexpr unsigned int Nb = 4` — Поле класса, которое представляет количество столбцов в состоянии шифра.

• `static constexpr unsigned int blockBytesLen` — Поле класса, которое используется для определения размера блока данных при работе с алгоритмом шифрования.

• `unsigned int Nk = 8` — поле класса, представляющее количество ключевых слов. Значение по умолчанию установлено в 8, что соответствует размеру ключа в 256 бит.

• `unsigned int Nr = 14` — поле класса, представляющее количество раундов в алгоритме AES.

Описание методов класса:

• AES() — Конструктор класса AES

• `void SubWord(unsigned char *a)` — Заменяет каждый из четырех байтов массива a на соответствующий байт из sbox.

• `void RotWord(unsigned char *a)` — Циклический сдвиг байтов массива a на одну позицию влево.

• `void XorWords(unsigned char *a, unsigned char *b, unsigned char *c)` — Выполняет побитовую операцию XOR для каждой из четырех пар байтов массивов a и b, сохраняя результат в массиве c.

• `unsigned char xtime(unsigned char b)` — Выполняет операцию умножения байта b на x в поле Галуа.

• `void Rcon(unsigned char *a, unsigned int n)` — Генерирует раундовый константный массив для ключа.

• `void KeyExpansion(const unsigned char key[], unsigned char w[])` — Расширяет ключ для использования в алгоритме AES.

• `void AddRoundKey(unsigned char state[4][Nb], unsigned char *key)` — Выполняет операцию побитового XOR для каждого элемента состояния и соответствующего ключа.

• `void SubBytes(unsigned char state[4][Nb])` — Заменяет каждый элемент состояния на соответствующий элемент из sbox.

• `void ShiftRow(unsigned char state[4][Nb], unsigned int i, unsigned int n)` — Циклический сдвиг строки i влево на n позиций.

• `void ShiftRows(unsigned char state[4][Nb])` — Выполняет циклический сдвиг каждой строки состояния влево на соответствующее количество позиций.

• `void MixColumns(unsigned char state[4][Nb])` — Выполняет операцию MixColumns над состоянием.

• `void EncryptBlock(const unsigned char in[], unsigned char out[], unsigned char *roundKeys)` — Шифрует блок данных с использованием ключа и выполняет необходимые операции.

• `unsigned char *EncryptECB(const unsigned char in[], unsigned int inLen, const unsigned char key[])` — Шифрует данные в режиме ECB (Electronic Codebook).

• `void InvSubBytes(unsigned char state[4][Nb])` — Заменяет каждый элемент состояния на соответствующий элемент из inv_sbox.

• `void InvMixColumns(unsigned char state[4][Nb])` — Выполняет операцию InvMixColumns над состоянием.

• `void InvShiftRows(unsigned char state[4][Nb])` — Выполняет обратный циклический сдвиг каждой строки состояния влево на соответствующее количество позиций.

• `void DecryptBlock(const unsigned char in[], unsigned char out[], unsigned char *roundKeys)` — Дешифрует блок данных с использованием ключа и выполняет необходимые операции.

• `unsigned char *DecryptECB(const unsigned char in[], unsigned int inLen, const unsigned char key[])` — Дешифрует данные в режиме ECB (Electronic Codebook).

4.3.6 Класс File

File — класс, реализующий различные операции с файлами.

Описание методов класса:

• `QByteArray readFile(const QString &fileName)` — Считывает содержимое файла и возвращает его в виде QbyteArray.

• `Void writeFile(const QString &fileName, const QByteArray &data)` — Записывает данные в файл с указанным именем.

• `quint64 fileSize(const QString &fileName)` — Возвращает размер файла в байтах.

5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

5.1 Разработка схем алгоритмов

Метод encryptECB(const unsigned char in[], unsigned int inLen, const unsigned char key[]) шифрует предоставляемые данные. Схема метода показана в приложении Б.

Метод DecryptECB(const unsigned char in[], unsigned int inLen, const unsigned char key[]) расшифровывает предоставляемые данные. Схема метода показана в приложении В.

5.2 Разработка алгоритмов

5.2.1 Метод encryptECB() класса AES

В данном методе представлен алгоритм шифрования данных алгоритмическим методом AES с помощью ключей.

Шаг 1: Производится инициализация ключа шифрования, загрузка исходного ключа в массив расширенных ключей KeyExpansion. Этот массив представляет собой расширенные ключи, генерируемые на основе исходного ключа.

Шаг 2: Для каждого блока данных выполняется операция XOR каждого байта с соответствующим байтом ключа раунда. Этот ключ раунда берется из массива расширенных ключей.

Шаг 3: Каждый байт блока данных заменяется соответствующим байтом из sbox — таблицы замен. Это шаг повышает стойкость шифра к различным видам атак.

Шаг 4: Производится циклический сдвиг строк матрицы состояния влево на определенное количество позиций. Это действие вносит дополнительную перестановку в структуру данных.

Шаг 5: Каждый столбец матрицы состояния умножается на фиксированный многочлен в поле Галуа. Это действие обеспечивает диффузию данных внутри блока.

Шаг 6: К блоку данных применяется операция поблочного сложения с раундовыми ключами. Это обеспечивает дополнительную перестановку и перемешивание данных.

Шаг 7: Операции Шага 3 – Шаг 6 проводятся несколько раундов (Nr — 1), где Nr – количество раундов, зависящее от длины ключа. Это обеспечивает дополнительный уровень безопасности и сложности шифрования.

Шаг 8: Выполняются те же операции, что и в предыдущих раундах, за исключением Шага 5. Это последний раунд, где не выполняется операция умножения столбцов.

Шаг 9: Вывод результата.

5.2.2 Метод decryptECB() класса AES

В данном методе представлен алгоритм дешифрования данных алгоритмическим методом AES с помощью ключей.
Шаг 1: Прежде всего, на основе переданного ключа генерируются раундовые ключи с использованием процедуры KeyExpansion. Эти ключи будут использоваться для дешифрования данных.
Шаг 2: Последний раундовый ключ добавляется к зашифрованному блоку данных. Это осуществляется через операцию XOR, где каждый байт блока данных складывается с соответствующим байтом последнего раундового ключа.
Шаг 3: Каждая строка матрицы состояния сдвигается вправо на определенное количество позиций. Этот шаг направлен на восстановление исходной структуры данных после шифрования.
Шаг 4: Каждый байт зашифрованных данных заменяется на соответствующий байт из inv_sbox. Этот процесс является обратной операцией к замене байтов, проведенной в процессе шифрования.
Шаг 5: Каждый блок данных матрицы состояния складывается с соответствующим раундовым ключом. Этот шаг представляет собой обратную операцию к поблочному сложению, выполненному при шифровании.
Шаг 6: Каждый столбец матрицы состояния умножается на фиксированный многочлен в поле Галуа. Этот шаг является обратной операцией к операции умножения столбцов, проведенной в процессе шифрования.
Шаг 7: Эти операции повторяются несколько раундов (Nr — 1), где Nr — количество раундов, зависящее от длины ключа. Каждый раунд восстанавливает часть исходных данных.
Шаг 8: В последнем раунде выполняются те же операции, что и в предыдущих, за исключением Шага 6. Это последний шаг в восстановлении исходных данных.
Шаг 9: Результатом является дешифрованный блок данных, который представляет собой исходные данные перед их зашифровкой.

6. РЕЗУЛЬТАТ РАБОТЫ

На рисунке 6.1 изображена начало работы программы. В интерфейсе со старта программы доступны кнопки: открытие файла, создание пароля, вставка пароля из буфера, копирование пароля в буфер, показать/скрыть пароль, очистка пароля из строки для редактирования, а так же главная кнопка, при нажатии которой файл будет зашифрован/расшифрован.

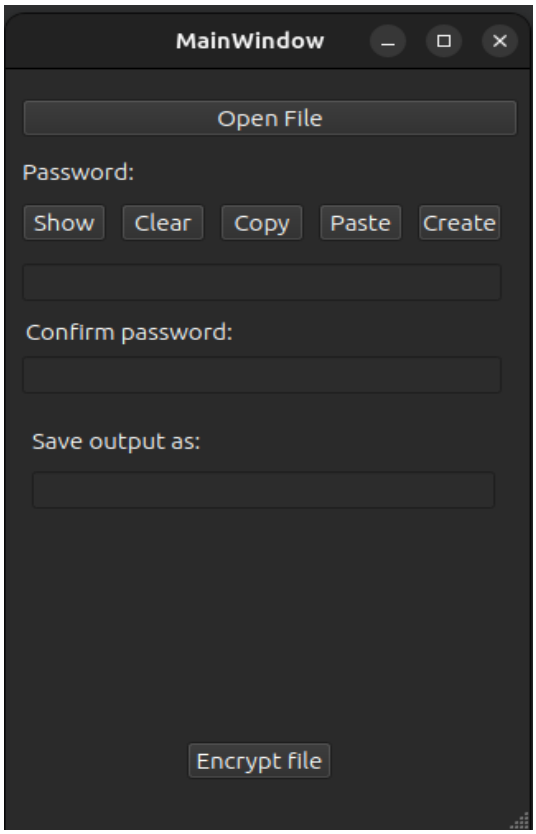

Рисунок 6.1 — Начало работы программы

На рисунке 6.2 показана работа кнопки создания пароля. При нажатии кнопки, открывается дополнительное окно с настройками пароля. В новом окне пользователь может настроить будет ли пароль состоять из символов верхнего регистра, нижнего регистра, цифр, специальных символов. Так же какую длину будет иметь пароль, и копировать ли сгенерированный пароль в буфер. После нажатия кнопки “ОК” в основном окне, в поле Password и Confirm password будет добавлен сгенерированный пароль, который будет скрыт под специальным символом. Если поля Password и Confirm password совпадают, то данные поля будут окрашены в зеленый цвет, если не совпадают – в красный (Рисунок 6.4).

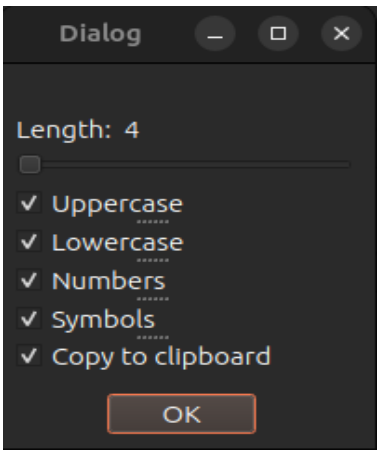

Рисунок 6.2 — Дополнительное окно создания пароля

На рисунке 6.3 отражено содержание файла, который мы будем шифровать.

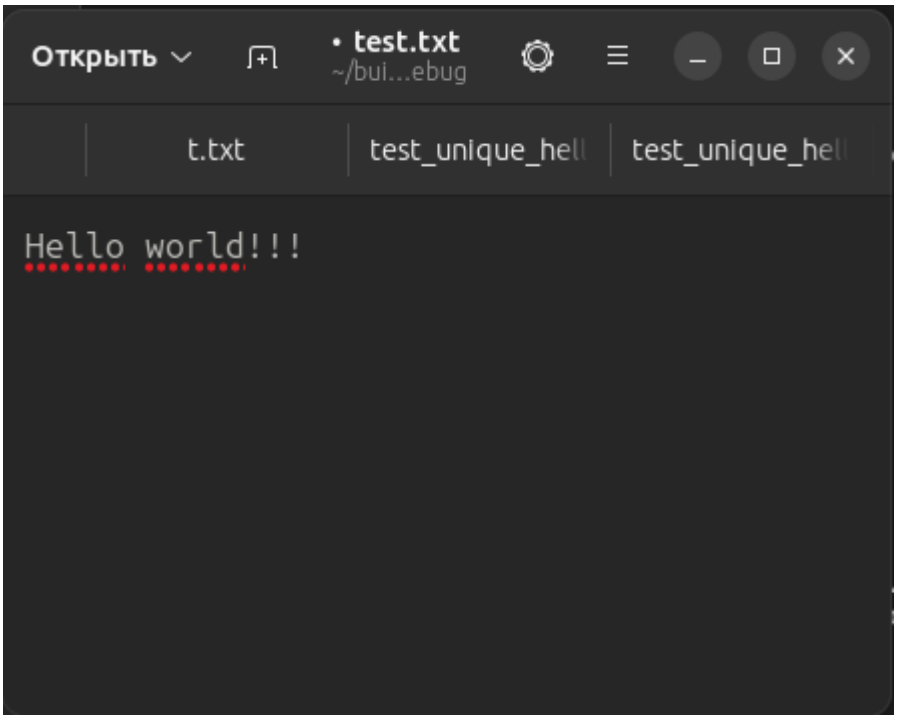

Рисунок 6.3 — Исходный файл

На рисунке 6.4 отражена программа перед шифрованием файла. При создании индивидуального пароля, нужно его подтвердить, то есть ввести точно такой же пароль в отведенное для этого места. Так же выбирается путь, куда будет сохранен зашифрованный файл, путь можно оставить по умолчанию.

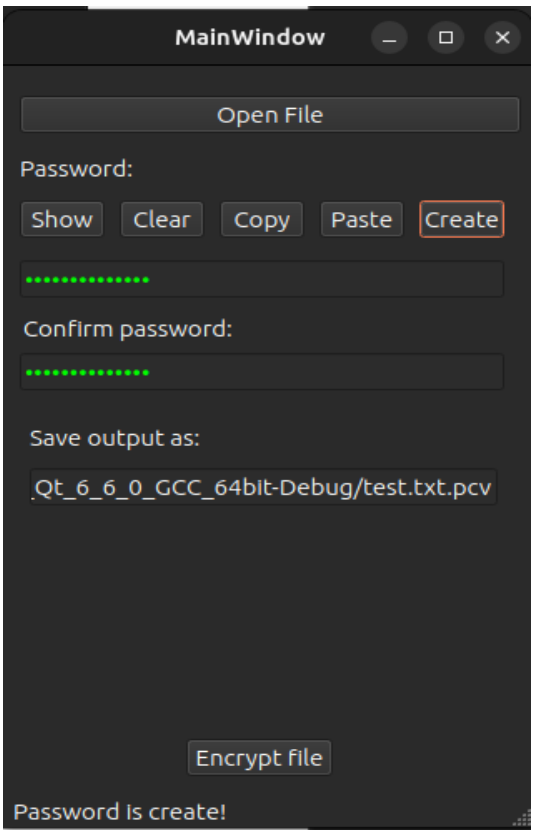

Рисунок 6.4 — Интерфейс программы перед шифрованием файла

На рисунке 6.5 показан зашифрованный файл, в нечитабельном формате .pcv

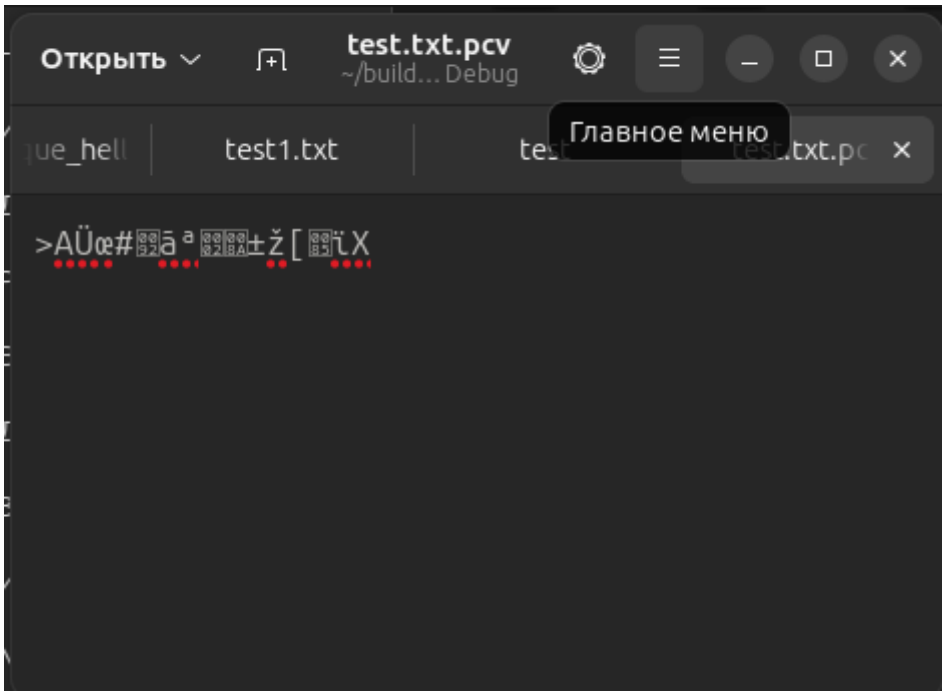

Рисунок 6.5 — Зашифрованный файл

На рисунке 6.6 отражен интерфейс программы перед расшифровкой файла. Для расшифровки файла нам нужно ввести пароль, которым мы шифровали наш файл, так же кнопка Create и поле Confirm Password являются не активными, так как это бессмысленно по логике нашей программы; кнопка Encrypt file заменяется на Decrypt file.

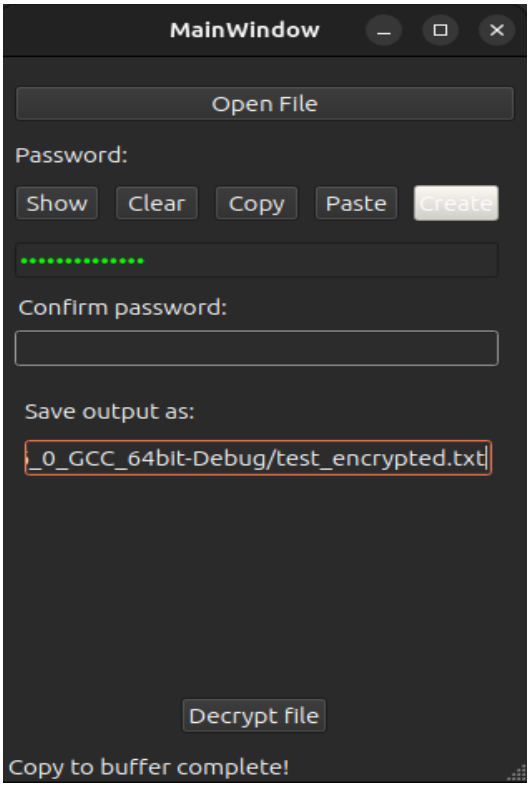

Рисунок 6.6 — Интерфейс программы перед расшифровкой файла

На рисунке 6.7 показан расшифрованный файл.

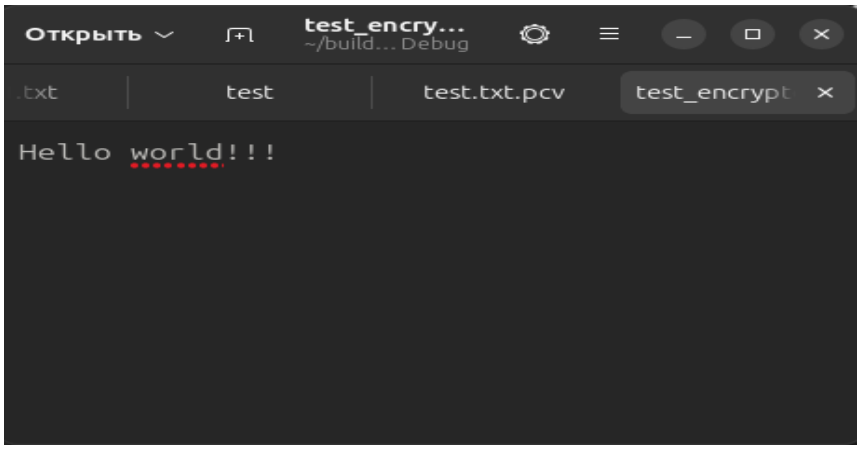

Рисунок 6.7 — Расшифрованный файл

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была рассмотрена реализация алгоритма шифрования файлов методом Advanced Encryption Standard (AES). AES представляет собой симметричный блочный шифр, широко применяемый для обеспечения конфиденциальности данных в современных информационных системах.

Алгоритм AES основан на подстановочно — перестановочной сети с использованием нескольких раундов операций SubBytes, ShiftRows, MixColumns и AddRoundKey. Эти операции обеспечивают высокий уровень безопасности шифрования и устойчивость к различным методам атак.

В рамках работы был реализован класс, предоставляющий функционал для шифрования и дешифрования файлов с использованием AES в режиме Electronic Codebook (ECB). Разработанный программный модуль позволяет безопасно обрабатывать файлы различных форматов, обеспечивая сохранность данных и конфиденциальность информации.

Процесс шифрования включает в себя чтение файла блоками, применение алгоритма AES к каждому блоку данных, и запись результата обратно в файл. Дешифрование выполняется обратным образом, что обеспечивает восстановление исходного файла.

Основное внимание уделено обеспечению эффективности и удобства использования разработанного программного модуля. Тестирование и анализ результатов продемонстрировали правильное функционирование алгоритма и возможность применения его к файлам различных размеров и типов.

В заключение, разработанный модуль представляет собой эффективный инструмент для обеспечения безопасности файлов, что делает его актуальным и полезным компонентом в области информационной безопасности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Рожнова, Н. Г. Вычислительные машины, системы и сети. Дипломное проектирование : учебно-метод.пособие / Н. Г. Рожнова, Н. А. Искра, И. И. Глецевич. – Минск : БГУИР, 2014. – 96 с. : ил.

[2] Шлее М. - Qt4. Профессиональное программирование на C+/ Шлее М. - Л.:Наука, 2013. - 770 с.

[3] Программирование на C++ [Электронный ресурс]. -Электронные данные. Режим доступа: https://metanit.com/cpp/tutorial/ -Дата доступа: 23.11.2023.

[4] Ефишов, Иван Иванович. Таинственные страницы. Занимательная криптография / Иван Ефишов. — М.: Манн, Иванов и Фербер, 2016. — 240 с.

[5] Как устроен AES [Электронный ресурс]. -Режим доступа: https://habr.com/en/articles/112733/ . -Дата доступа: 21.10.2023.

[6] Симметричный алгоритм блочного шифрования Advanced Encryption Standart [Электронный ресурс]. -Режим доступа: https://habr.com/en/articles/534620/ . -Дата доступа: 22.10.2023.

[7] Как работает AES (Advanced Encryption Standard). Объяснение для гуманитариев типа меня [Электронный ресурс]. -Режим доступа: https://vc.ru/dev/656195-kak-rabotaet-aes-advanced-encryption-standard-obyasnenie-dlya-gumanitariev-tipa-menya. -Дата доступа: 10.10.2023.

[8] Qt for Beginners [Электронный ресурс]. –Режим доступа: https://wiki.qt.io/Qt_for_Beginners. –Дата доступа: 19.11.2023.

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ В

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Г

(обязательное)

Диаграмма классов