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 hash

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++[1]. Разработать приложение "Шифратор файлов на C++" с использование среды разработки Qt. Вот несколько общих целей, которые могут быть установлены для освоения ООП:

Понимание Основных Принципов ООП:

Цель: Освоить базовые концепции ООП, такие как инкапсуляция, наследование и полиморфизм. Почему это важно: Это обеспечит более глубокое понимание организации кода и его взаимодействия.

Навыки Проектирования Классов и Объектов:

Цель: уметь создавать классы и объекты, определять их атрибуты и методы. Почему это важно: это является основой ООП и позволяет структурировать код для более легкого понимания и поддержки.

Применение Инкапсуляции:

Цель: использовать инкапсуляцию для скрытия внутренних деталей реализации классов и предоставления публичного интерфейса. Почему это важно: это способствует безопасности кода и облегчает его сопровождение.

Мастерство В Обработке Наследования: Цель: понимать, как использовать наследование для создания иерархии классов и расширения функциональности. Почему это важно: это позволяет эффективно использовать и пере использовать код.

Овладение Исключениями и Обработкой Ошибок: Цель: Знание, как обрабатывать исключения и ошибки в объектно-ориентированных программах. Почему это важно: это повышает устойчивость программы к ошибкам и улучшает ее отказоустойчивость.

Работа с Дизайн-паттернами:

Цель: Знание и применение распространенных дизайн-паттернов.
Почему это важно: Дизайн-паттерны предоставляют bewmee эффективные и проверенные подходы к решению типичных задач в ООП.
Эти цели могут служить отправной точкой для разработки программистом плана обучения и практического применения концепций ООП в реальных проектах.

Преимущество Qt:

Простота использования: Qt имеет хорошо документированную структуру, прост в освоении и имеет простую, однородную структуру.

Разработка GUI: Qt предоставляет широкий спектр компонентов для создания графических пользовательских интерфейсов, включая Qt Quick,

который позволяет быстро и легко создавать интерфейс с использованием специального языка под названием QML.

Qt — это кроссплатформенный фреймворк для разработки приложений на C++. Он предоставляет разработчикам множество инструментов и библиотек для создания графического интерфейса пользователя, работы с сетью, базами данных, мультимедиа и многого другого. Ниже перечислены некоторые преимущества Qt:

Кроссплатформенность: Qt позволяет создавать приложения, которые могут работать на различных операционных системах, таких как Windows, macOS, Linux, Android и iOS. Это упрощает разработку и позволяет достичь большей аудитории.

Мощный графический интерфейс: Qt предоставляет разработчикам множество инструментов для создания красивых и функциональных пользовательских интерфейсов. Он также поддерживает множество стилей и тем оформления, что позволяет создавать приложения с различным дизайном. 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, от настольных и мобильных приложений до специализированного оборудования и встроенных систем.

Богатая функциональность: Qt предоставляет множество библиотек и инструментов для работы с сетью, базами данных, мультимедиа, графикой и многим другим. Это позволяет разработчикам создавать приложения с различными функциями и возможностями.

Открытый исходный код: Qt является свободным и открытым фреймворком, что позволяет разработчикам использовать его бесплатно и вносить свои изменения в исходный код.

Широкое сообщество: Qt имеет большое сообщество разработчиков, которые создают и поддерживают множество библиотек и инструментов для Qt. Это позволяет разработчикам быстро находить решения для своих проблем и получать поддержку от других разработчиков.

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

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. AES является также алгоритмом блочного шифрования. Это простая и легко реализуемая на различных платформах концепция шифрования, которая сводится к подстановкам и перестановкам цифр в блоках обрабатываемых данных.

Блочный шифр работает с отдельными блоками данных фиксированной длины (в случае AES это 128 бит) на которые разбивается вся шифруемая информация, если длина сообщения меньше длины блока, то оно дополняется до размера блока.

На рисунке 3.1.1 показана схема алгоритма AES в режиме ECB

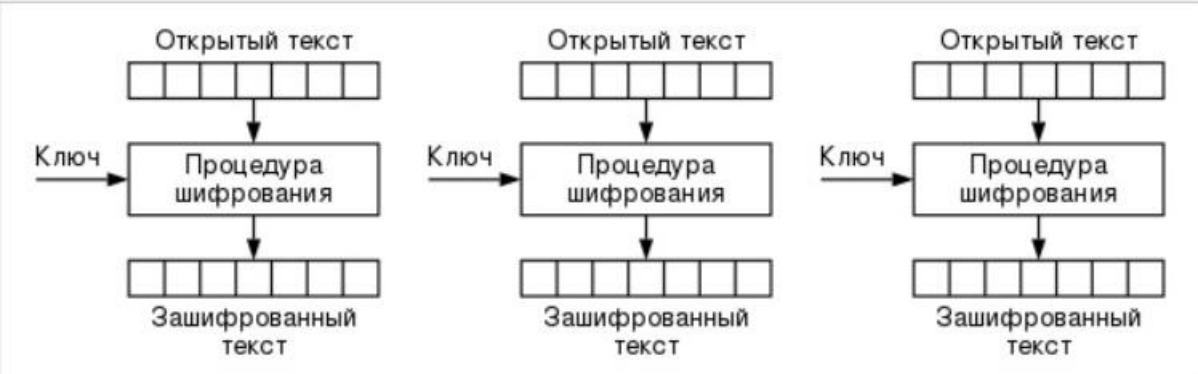

Рисунок 3.1.1 – Схема алгоритма

Раундовые ключи вырабатываются из ключа шифра K с помощью процедуры расширения ключа(KeyExpansion), в результате чего формируется массив раундовых ключей, из которого затем непосредственно выбирается необходимый раундовый ключ.

Каждый раундовый ключ имеет длину 128 бит (или 4 четырехбайтовых слова w_i, w_{i+1}, w_{i+2}, w_{i+3}, а длина в битах всех раундовых ключей равна 128 бит * (14 раундов+1) = 1920 бит (или 60 четырхбайтовых слова w_0, w_1, w_2, … , w_{58}, w_{59}). Первые четыре слова w_0, w_1, w_2, w_3 в ключевом массиве заполнены ключом шифра, из остальных выработанных слов выбираются слова для ключа раунды).

Новые слова w_{i+4}, w_{i+5}, w_{i+6}, w_{i+7} следующего раундового ключа определяются из слов w_i, w_{i+1}, w_{i+2}, w_{i+3} предыдущего ключа на основе уравнений представленных на рисунке 3.1.1.2

$$w_{i+5} = w_{i+4} \oplus w_{i+1};$$
$$w_{i+6} = w_{i+5} \oplus w_{i+2};$$
$$w_{i+7} = w_{i+6} \oplus w_{i+3}.$$

Рисунок 3.1.2 – уравнение подбора новых ключей

Рассмотрим подробнее преобразования раунда шифрования.

Первой операцией раунда шифрования является операция SubBytes. Операция выполняет нелинейную замену байтов, выполняемую независимо с каждым байтом матрицы State. Замена обратима и построена путем комбинации двух преобразований над входным байтом:

Нахождение инвертированного элемента относительно умножения в поле Галуа.

Выполнение некого аффинного преобразования: умножение инвертированного байта на многочлен $a(x) = x^4 + x^3 + x^2 + x + 1$ и суммирование с многочленом $b(x) = x^6 + x^5 + x + 1$.

Процесс замены байтов с помощью таблицы подстановки иллюстрирует рисунок 3.1.2

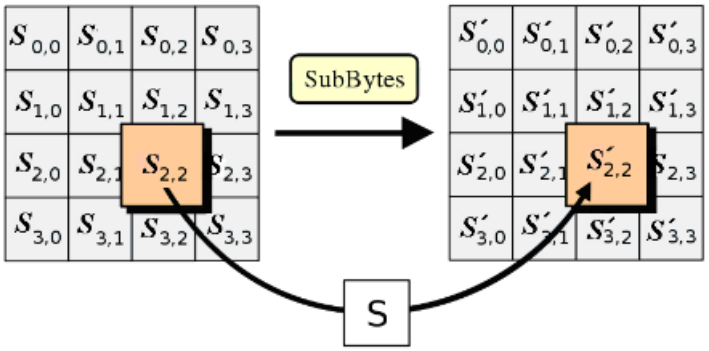

Рисунок 3.1.2 – Схема процедуры SubBytes

Созданную на основе этой операции специальную таблицу замен байтов в шестнадцатеричной системе называют S-боксом.

Второй операцией является ShiftRows – операция применяется к строкам матрицы State – ее первая строка неподвижна, а элементы нижних трех строк циклически сдвигаются вправо на 1, 2 и 3 байта соответственно. На рисунке 3.1.3 представлена схема данной процедуры.

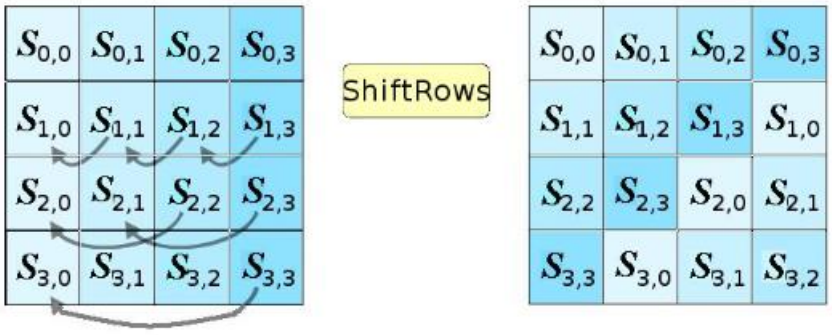

Рисунок 3.1.3 – Схема процедуры ShiftRows

Третьей операцией является MixColumns. С помощью этой операции выполняется перемешивание байтов в столбцах матрицы State. Каждый столбец этой матрицы принимается за многочлен над полем Галуа и умножается на фиксированный многочлен, все коэффициенты многочленов над полем Галуа – байты. Такую операцию можно записать в матричном виде (Рисунок 3.1.4)

$$\begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} s_0' \\ s_1' \\ s_2' \\ s_3' \end{pmatrix}.$$

Рисунок 3.1.4 – Матричный вид операции MixColumns

Заключительной четвертой процедурой является AddRoundKey. Данная процедура побитово складывает элементы переменной RoundKey и элементы переменной State по принципу: i-й столбец данных складывается с определенным 4-байтовым фрагментом расширенного ключа W[4r + 1], где r – номер поточного раунда алгоритма. При шифровании первое сложение ключа раунда происходит до первого выполнения операции SubBytes. На рисунке 3.1.5 показана схема данного алгоритма.

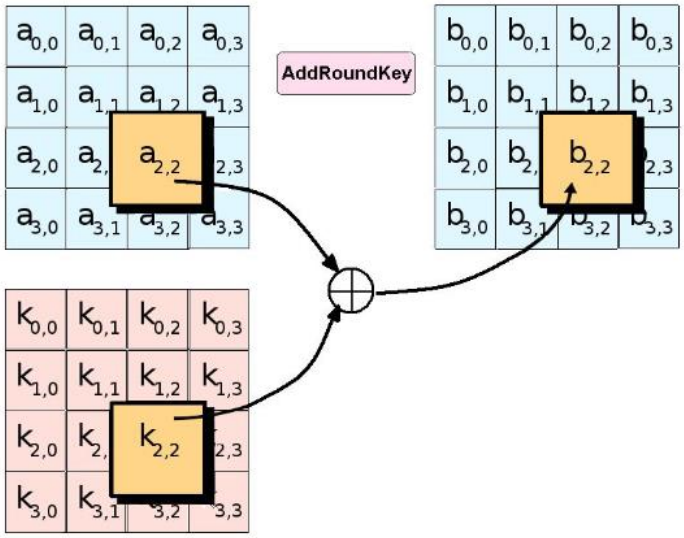

Рисунок 3.1.5 – схема алгоритма AddRoundKey

Рисунок 3.1.6 демонстрирует свойства перемешивания информации в ходе шифрования алгоритмом AES. Видно, что два раунда обеспечивают полное рассеивание и перемешивание информации. Достигается это за счет использования функций ShiftRows и MixColumns. Операция SubBytes придает шифрованию стойкость против дифференциального криптоанализа, а операция AddRoundKey обеспечивает секретную случайность.

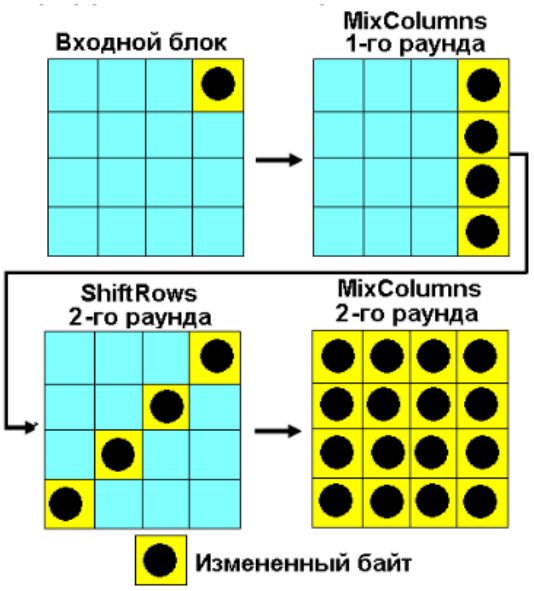

Рисунок 3.1.6 – Схема перемешивания информации

Отмечу также, что для генерации безопасных хэш-сумм ключей шифрования применяется хэш-функция SHA-256. Это дополнительный уровень безопасности, который обеспечивается при обработке данных.

3.2 Обзор аналогов

3.2.1 VeraCrypt

VeraCrypt[8] является мощным инструментом для шифрования файлов и дисков. Он поддерживает различные алгоритмы шифрования, включая AES, Serpent, и Twofish. VeraCrypt также позволяет создавать зашифрованные контейнеры и полные зашифрованные диски.

Преимущества: Открытый исходный код, кросс-платформенность, поддержка различных алгоритмов шифрования.

3.2.2 AES Crypt

AES Crypt[9] - простой и легкий в использовании инструмент для шифрования файлов. Использует стандарт Advanced Encryption Standard (AES) для обеспечения безопасности данных.

Преимущества: Простой интерфейс, кросс-платформенность, отсутствие необходимости установки дополнительных программ.

3.2.3 Cryptomator

Cryptomator[10] разработан для обеспечения безопасного шифрования файлов в облачных хранилищах. Он создает виртуальные шифрованные контейнеры для хранения данных в облаке.

Преимущества: Открытый исходный код, простой интерфейс, автоматическое шифрование данных в облачных хранилищах.

3.2.4 BitLocker (для Windows)

BitLocker[11] - встроенное средство шифрования дисков в операционной системе Windows. Поддерживает различные режимы шифрования, включая AES-CBC и XTS-AES.

Преимущества: Интегрирован в Windows, обеспечивает шифрование всего диска.

Каждый из упомянутых шифраторов файлов имеет свои преимущества и недостатки, и выбор зависит от конкретных потребностей пользователя, операционной системы и предпочтений в использовании.

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

В данном проекте был выбран язык программирования C++, а также применена методология объектно-ориентированного программирования (ООП). Далее рассмотрим более подробно основные этапы решения задачи, включая входные и выходные данные.

4.1.1 Входные данные

На этапе ввода программа принимает файл для шифрования и пароль, который может быть сгенерирован случайным образом, в зависимости от выбора пользователя. Приложение спроектировано для считывания файлов любого типа, что делает его универсальным инструментом для обработки данных. Благодаря использованию различных возможностей языка C++, программа обрабатывает файлы различных размеров и с разными именами, гарантируя корректное выполнение операций шифрования.

4.1.2 Выходные данные

По завершении работы приложения получается зашифрованный файл в формате (.pcv), который представляет собой нечитаемый для пользователя файл. Выбор данного формата осуществлен произвольно, и в случае необходимости его можно легко изменить на любой другой формат. Важно отметить, что для последующей расшифровки файла необходимо сохранить использованный пароль. Без сохраненного пароля процесс расшифровки будет невозможен.

Таким образом, разработанное приложение на языке C++ с применением объектно-ориентированного программирования обеспечивает гибкость и универсальность в обработке файлов, а также гарантирует безопасность данных через использование пароля для шифрования и последующей расшифровки файлов.

4.2. Разработка диаграммы классов

Диаграмма классов - это инструмент в языке моделирования UML, который используется для визуализации структуры классов в системе, их атрибутов, методов, интерфейсов и отношений между ними. Эта диаграмма

обычно применяется при проектировании архитектуры, документировании системы, уточнении требований, а также для поддержки системы.

Диаграмма классов иллюстрирует модели данных даже для очень сложных информационных систем. На ней представлены в рамках, содержащих три компонента:

- В верхней части написано имя класса. Имя класса выравнивается по центру и пишется полужирным шрифтом;
- В средней части перечислены атрибуты (поля) класса;
- В нижней части перечислены методы класса.

Диаграмма классов служит для визуализации статического представления системы, представляя различные аспекты приложения. Она представляет собой графическое представление статического представления системы и представляет различные аспекты приложения и также может включать в себя классы, их атрибуты, методы и отношения между классами, такие как наследование, агрегация, ассоциация, множественность ассоциации, обобщение, зависимость, использование, реализация и композиция.

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 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 файла методом AES
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.

`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.

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 `std::string hash(const std::string& input)` хеширует предоставляемые ключи. 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.

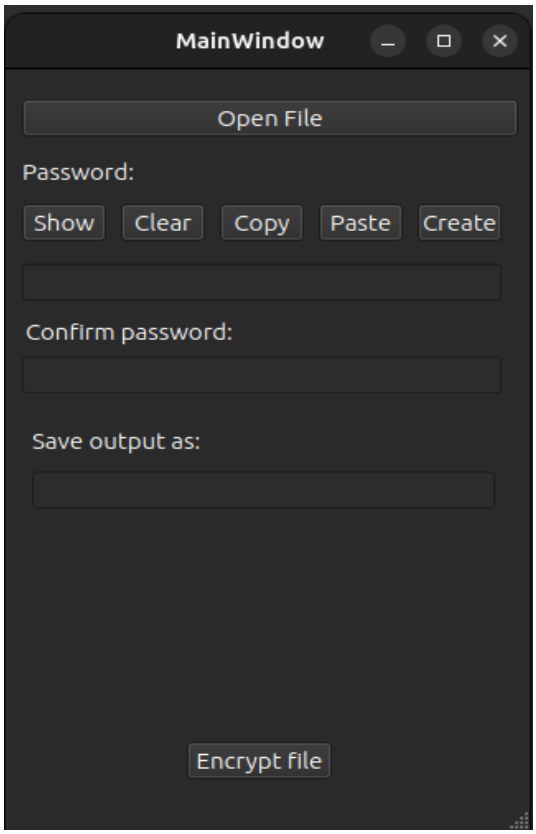

Рисунок 6.1 — Начало работы программы

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 (Рисунок 6.4), если не совпадают – в красный.

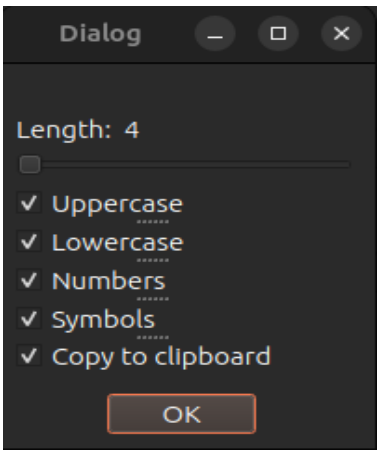

Рисунок 6.2 — Дополнительное окно создания пароля

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

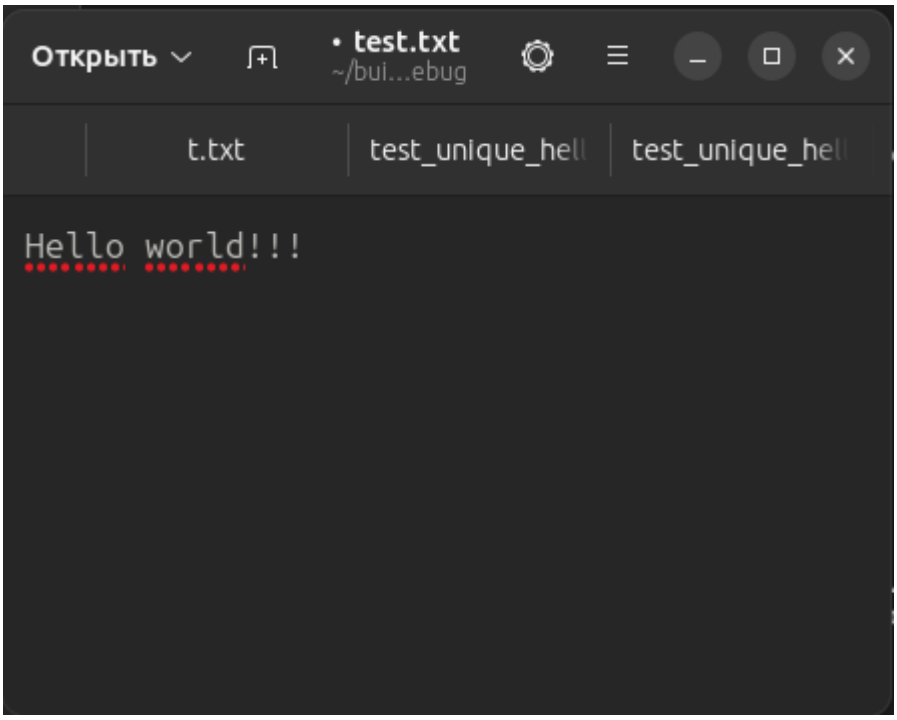

Рисунок 6.3 — Исходный файл

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.

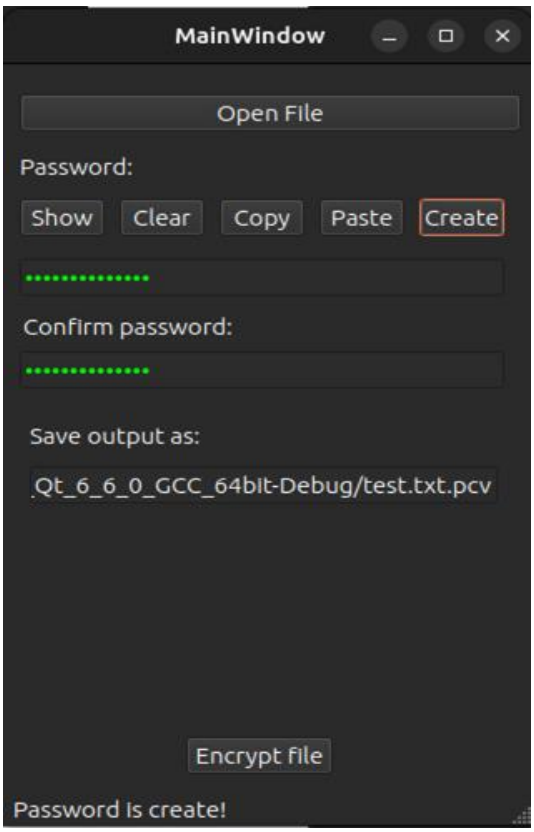

Рисунок 6.4 — Интерфейс программы перед шифрованием файла

На рисунке 6.5 показан зашифрованный файл в нечитабельном формате .pcv

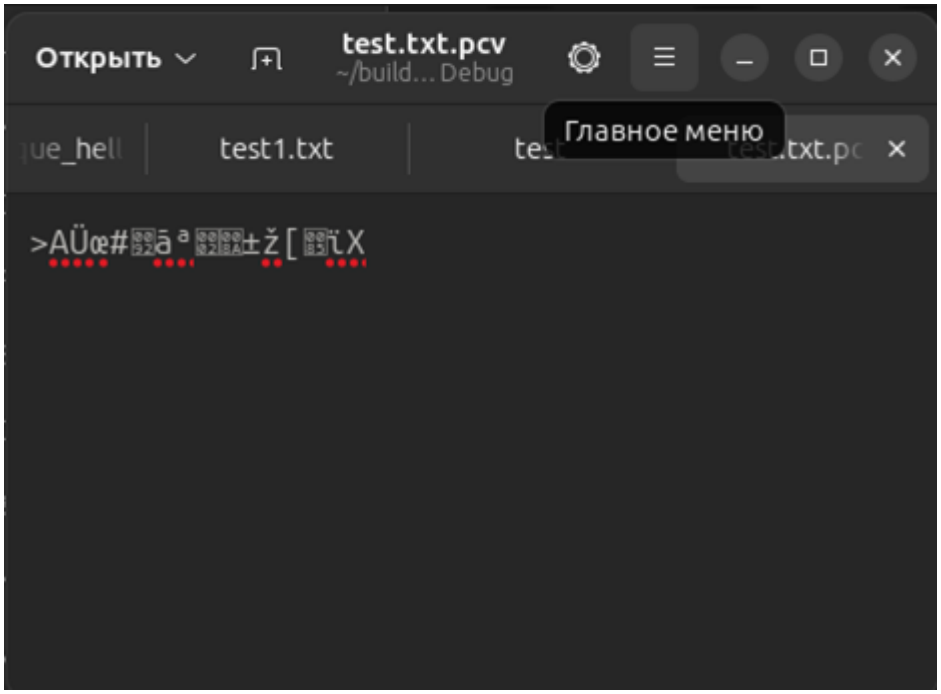

Рисунок 6.5 — Зашифрованный файл

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.

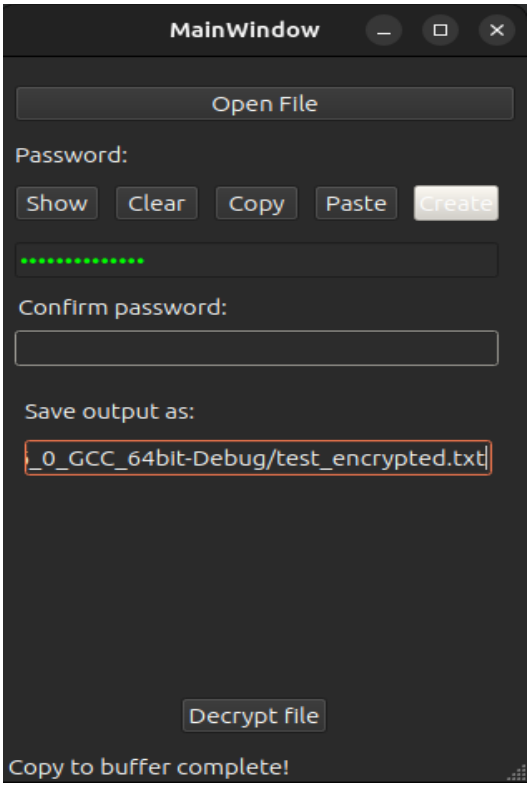

Рисунок 6.6 — Интерфейс программы перед расшифровкой файла

На рисунке 6.7 показан расшифрованный файл.

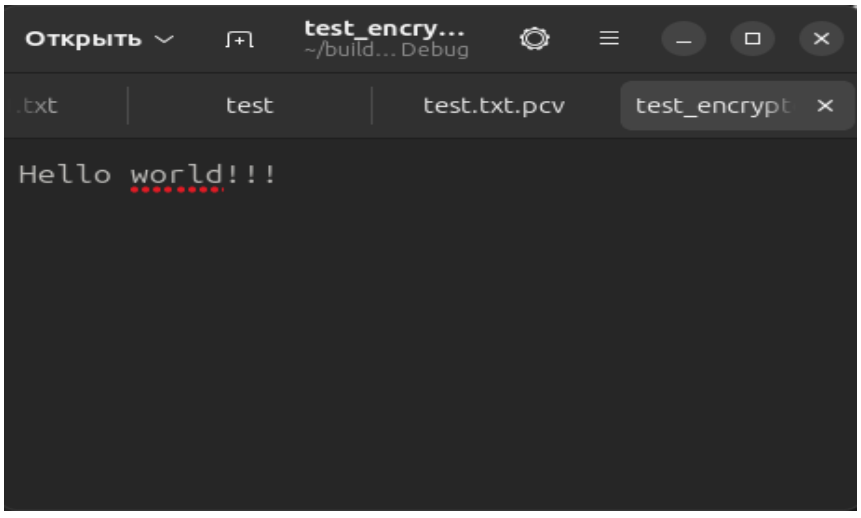

Рисунок 6.7 — Расшифрованный файл

ЗАКЛЮЧЕНИЕ

Итак, изучение объектно-ориентированного программирования (ООП) выдало себя как преобразующий этап в моем понимании и подходе к разработке программного обеспечения. Погружение в мир ООП расширило мои знания и компетенции, обогатив мой инструментарий в области программирования.

Одним из ключевых выводов, которые я сделал в процессе написания курсовой работы, является то, что ООП не только предоставляет эффективные средства организации кода, но и обеспечивает более высокий уровень абстракции, что делает разработку программ более интуитивной и гибкой. Использование классов и объектов позволяет создавать модульные, легко читаемые и поддерживаемые системы.

В наше современное время, когда требования к программному обеспечению постоянно растут, ООП становится неотъемлемой частью профессиональной подготовки разработчика. Адаптивность кода, возможность масштабирования проектов и повторного использования компонентов становятся критическими аспектами успешной разработки. ООП предоставляет технологический фреймворк для разработки сложных, но гибких систем, способных адаптироваться к изменяющимся требованиям бизнеса.

В заключение, наша разработанная программа шифрования файлов, разработанная на платформе Qt, представляет собой мощный инструмент для обеспечения безопасности и конфиденциальности данных. Был успешно реализован алгоритм шифрования Advanced Encryption Standard (AES), который является одним из наиболее надежных и широко используемых симметричных блочных шифров в современных информационных системах.

Алгоритм AES, основанный на подстановочно-перестановочной сети, включает несколько раундов операций, таких как SubBytes, ShiftRows, MixColumns и AddRoundKey, обеспечивающих высокий уровень безопасности и защиту от различных методов атак, включая криптоанализ и подбор ключа. В нашей программе мы реализовали эти операции с высокой эффективностью и точностью, чтобы обеспечить надежную защиту файлов и данных.

Для дополнительного уровня безопасности, мы также использовали алгоритм SHA256 для шифрования ключей пароля. SHA256 - это криптографическая хеш-функция, которая обеспечивает непростой и надежный метод генерации ключей пароля. Это гарантирует, что доступ к зашифрованным файлам возможен только при использовании правильного пароля, что повышает безопасность и защиту ваших данных.

Основываясь на анализе результатов тестирования, мы убедились в правильном функционировании программы и ее способности обрабатывать файлы различных размеров и типов. Мы также применили строгие стандарты качества и безопасности при разработке программы, чтобы гарантировать ее надежность и защиту данных.

Одна из ключевых особенностей программы является удобный и интуитивно понятный пользовательский интерфейс, который делает ее простой в использовании для шифрования и дешифрования файлов. Также я предоставил подробную документацию и руководство пользователя, чтобы облегчить начало работы с программой и максимально использовать ее потенциал.

В целом, наша программа шифрования файлов, основанная на алгоритме AES и алгоритме SHA256, представляет собой надежное и эффективное решение для обеспечения безопасности данных. Она обладает высоким уровнем защиты и гарантирует конфиденциальность ваших файлов. Я уверен, что наша программа будет полезна для широкого круга пользователей, которые ищут надежное и удобное решение для защиты своих файлов от несанкционированного доступа.

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

[8] VeraCrypt [Электронный ресурс]. -Режим доступа: https://www.veracrypt.fr/en/Home.html. -Дата доступа: 25.11.2023.

[9] AES Crypt [Электронный ресурс]. -Режим доступа: https://www.aescrypt.com/. -Дата доступа: 25.11.2023.

[10] Cryptomator [Электронный ресурс]. -Режим доступа: https://cryptomator.org/. -Дата доступа: 25.11.2023.

[11] BitLocker [Электронный ресурс]. -Режим доступа: https://learn.microsoft.com/ru-ru/windows/security/operating-system-security/data-protection/bitlocker/. -Дата достпа: 25.11.2023.

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема метода encryptECB()

ПРИЛОЖЕНИЕ В

(обязательное)

Схема метода hash()

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов