**Актюбинский региональный университет имени К.Жубанова**

**Факультет физики и математики**

**Кафедра информатики и информационных технологий**

**Проект по дисциплине:** Технологии кибербезопасности

**ОП:** 7М06103 - «Вычислительная техника и программное обеспечение»

**Тема:** «Разработка приложения для шифрования и дешифрования файлов на языке C#»

**Выполнил:** Кемал С.Қ., магистрант 2 курса

**Проверила:** Шангытбаева Г.А. доцент, PhD

**Актобе, 2024**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc184406113)

[1 Основы проектной реализации 4](#_Toc184406114)

[1.1 Определение функциональных требований 4](#_Toc184406115)

[1.2 Обзор методов шифрования 6](#_Toc184406116)

[1.3 Техническая подготовка 7](#_Toc184406117)

[1.3.1 Выбор технологий и инструментов 7](#_Toc184406118)

[1.3.2 Архитектура приложения 7](#_Toc184406119)

[2. РЕАЛИЗАЦИЯ 10](#_Toc184406120)

[2.1 Реализация метода Цезаря 10](#_Toc184406121)

[2.2 Код шифра Виженера 13](#_Toc184406122)

[2.3 Алгоритм Плейфера. 14](#_Toc184406123)

[2.4 Пример использования AES для работы с файлами. 16](#_Toc184406124)

[ЗАКЛЮЧЕНИЕ 20](#_Toc184406125)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 23](#_Toc184406126)

**ВВЕДЕНИЕ**

В современном мире, где цифровые технологии имеют решающее значение для жизни людей, защита данных стала важным компонентом безопасности. В наши дни миллиарды устройств и систем ежедневно обрабатывают и передают огромные объемы информации, что делает их потенциальными целями для кибератак. Инфраструктура национальной безопасности, базы данных здравоохранения, финансовые системы и персональные смартфоны все подвержены риску.

*Актуальность исследования*. Утечки данных или атаки на системы могут иметь разрушительные последствия. Например, взлом базы данных большого госпиталя может привести к раскрытию конфиденциальной информации о пациентах, задержкам в медицинских процедурах или даже к угрозе их жизни. Такие инциденты в бизнесе могут причинить многомиллионные потери и причинить непоправимый ущерб репутации компании. Отчеты показывают, что кибератаки могут привести к ущербу в 10 триллионов долларов к 2023 году [1].

Кибербезопасность требует непрерывного прогресса. Это не только проблема для профессионалов; она касается всех, кто работает над разработкой цифровых решений. Будущие разработчики должны быть готовы к разработке новых инструментов и технологий для защиты данных. Например, количество атак с использованием фишинга и вредоносного ПО увеличилось на 20% в 2020-х годах. Это показывает, что внедрение более сложных алгоритмов шифрования является жизненно важным [2].

Кроме того, разработка таких стандартов шифрования, как AES и RSA, позволила значительно улучшить защиту данных. Эти стратегии используются повсеместно, от защиты онлайн-платежей до защиты переписки. Технологии шифрования не только защищают информацию, но и являются важным средством предотвращения утечек данных, что особенно важно в наше время, когда количество подключенных устройств растет [3].

*Цель исследования***.** Данное исследование нацелена на изучение основ работы простых схем шифрования данных с использованием пользовательских паролей, а также анализ их эффективности в контексте защиты информации в современных цифровых системах. В центре внимания лежат базовые алгоритмы шифрования и дешифрования, а также их значимость для обеспечения безопасности личной, финансовой и корпоративной информации.

*Задачи исследования***.** Для достижения поставленной цели определены следующие задачи:

*Анализ и обобщение ключевых принципов шифрования данных.* Рассмотрение простых схем шифрования, включая их возможности и ограничения, а также изучение основных методов, таких как симметричные и асимметричные алгоритмы.

*Исследование значения паролей в системах защиты данных.* Оценка их роли как средства защиты информации и анализ их эффективности в различных контекстах.

*Повышение осведомленности о важности обучения* в сфере кибербезопасности и подготовка специалистов, что позволяет расширить понимание принципов шифрования.

*Сравнительный анализ* простых и современных алгоритмов шифрования, выявляющий сферы, где простые схемы могут оставаться актуальными.

В результате выполнения этих задач исследование стремится предоставить полное представление о роли шифрования данных и паролей в современных системах защиты, а также указать пути для оптимизации защиты информации в условиях растущих угроз цифровой безопасности.

*Практическая значимость***.** Данного проекта заключается в разработке методов шифрования, которые обеспечивают надежную защиту различных типов данных, включая текст, видео, фото и аудио. Результаты исследования могут быть применены для защиты персональной информации, предотвращая несанкционированный доступ к текстовым сообщениям, личным фотографиям, видео и аудиофайлам.

Также разработанные алгоритмы могут быть внедрены в корпоративные системы для обеспечения безопасности документов и коммуникаций, что позволяет компаниям минимизировать финансовые и репутационные риски. В сфере медицины и образования шифрование обеспечивает защиту конфиденциальных данных, таких как медицинские записи и учебные материалы.

Кроме того, использование алгоритмов шифрования актуально для защиты авторских прав на мультимедийный контент, исключая его незаконное копирование или распространение. В государственных и военных структурах методы шифрования аудио- и видеоданных могут использоваться для обеспечения безопасности стратегически важных данных.

Таким образом, проект представляет собой универсальное решение для обеспечения защиты информации в условиях роста цифровых угроз.

# **1 Основы проектной реализации**

Разработка системы защиты данных с использованием методов шифрования с применением пользовательских паролей требует четкой постановки задач и определения функциональных требований. В данном контексте особое внимание следует уделить поддержке современных алгоритмов шифрования, удобству взаимодействия пользователя с системой и возможности работы с разнообразными форматами данных.

## **1.1 Определение функциональных требований**

*Поддержка нескольких алгоритмов шифрования****.***Система должна обеспечивать возможность работы с различными алгоритмами шифрования. В первую очередь, необходимо внедрить поддержку алгоритма AES (Advanced Encryption Standard), который является одним из наиболее надежных и широко используемых методов симметричного шифрования. AES обеспечивает высокий уровень безопасности данных, что делает его идеальным для защиты конфиденциальной информации, в том числе текстовых, аудиовизуальных файлов и документов.Кроме того, следует предусмотреть возможность использования дополнительных алгоритмов, таких как шифр Цезаря и алгоритм Виженера для базового уровня защиты и обучения пользователей принципам шифрования. Эти алгоритмы обеспечат легкий доступ к основам криптографии и пониманию принципов работы шифрования и дешифрования [4].

*Интуитивно понятный пользовательский интерфейс (Windows Forms)****.***Разработка интерфейса приложения должна быть ориентирована на простоту и удобство взаимодействия. Windows Forms, как одна из самых популярных платформ для создания настольных приложений, обеспечит высокий уровень комфорта для пользователей. Интерфейс должен включать элементы управления, такие как:

* Поля для ввода текста или загрузки файлов для шифрования.
* Выпадающие списки или кнопки для выбора алгоритма шифрования.
* Кнопки для запуска процессов шифрования и дешифрования.
* Поля для ввода пароля, который будет использоваться в качестве ключа шифрования.
* Поле для отображения статуса процесса (например, "Шифрование завершено" или "Ошибка при шифровании").

Важной частью интерфейса является простота и логика размещения элементов, что позволяет пользователю без проблем понять, как использовать приложение даже при отсутствии специальных технических знаний.

*Возможность работы с различными форматами файлов****.***Приложение должно поддерживать работу с различными форматами данных, включая текстовые, аудиовизуальные и графические файлы. Это включает:

* + - * + *Текстовые файлы (.txt, .csv, .docx)* — возможность шифрования и дешифрования простых и структурированных текстов.
        + *Аудиофайлы (.mp3, .wav, .aac)* — возможность защиты аудиозаписей, что особенно важно для защиты личных или коммерческих аудиоконтентов.
        + *Видеофайлы (.mp4, .avi, .mov)* — поддержка шифрования видеоматериалов, чтобы предотвратить несанкционированный доступ к видеоконтенту.
        + *Графические файлы (.jpg, .png, .bmp, .gif)* — защита изображений и фотографий, что важно для хранения личных и профессиональных данных.

Система должна быть способна эффективно обрабатывать эти файлы, обеспечивая не только их шифрование и дешифрование, но и корректное сохранение структуры данных при восстановлении.

*Обеспечение совместимости и безопасности.* Приложение должно поддерживать криптографические стандарты и алгоритмы, соответствующие современным требованиям безопасности. В случае использования AES следует использовать ключи с длиной 128, 192 или 256 бит, в зависимости от уровня защиты, который требуется.

Для повышения уровня безопасности следует внедрить алгоритмы проверки целостности данных, такие как хэш-функции (SHA-256), чтобы обеспечить надежность данных и предотвратить их повреждение или несанкционированное изменение при шифровании и дешифровании.Примеры использования системы:

* + - *Защита личных данных:* пользователь может зашифровать текстовый документ с конфиденциальной информацией или фотоальбом с личными фотографиями.
    - *Защита коммерческой информации:* предприятие может использовать систему для защиты видеоуроков, рабочих презентаций или базы данных с клиентскими данными.
    - *Обмен зашифрованными файлами:* возможность шифрования файла перед отправкой по электронной почте или через мессенджеры для предотвращения несанкционированного доступа.

**1.2 Обзор методов шифрования**

Шифрование является важным компонентом защиты данных, и выбор алгоритма зависит от целей, требуемого уровня безопасности и сложности реализации. В данной исследование рассматриваются четыре различных метода шифрования, каждый из которых имеет свои особенности, преимущества и недостатки.

*Метод Цезаря (шифр Цезаря).* Этот метод шифрования был предложен Юлием Цезарем, который использовал его для защиты военных сообщений. Он относится к простейшим алгоритмам шифрования и основывается на подстановке символов. В шифре Цезаря каждый символ исходного текста смещается на фиксированное число позиций в алфавите. Например, при смещении на 3 буква «А» становится «Г», «Б» — «Д» и так далее. Этот метод обладает исторической значимостью и прост в понимании и реализации, но его безопасность оставляет желать лучшего из-за легкости взлома методом подбора [5].

*Шифр Виженера.* Шифр Виженера, в отличие от метода Цезаря, использует ключ, который повторяется по длине текста. Он обеспечивает более высокий уровень защиты, поскольку каждый символ исходного текста шифруется с использованием разных смещений, зависящих от ключа. Это делает метод устойчивым к простому криптоанализу, в отличие от шифра Цезаря, который легко взламывается методом подбора. Шифр Виженера подходит для защиты текстов средней важности и используется, например, в задачах, связанных с личной перепиской [6].

*Шифр Плейфера.* Шифр Плейфера использует таблицу с буквами, которая состоит из 5x5 (или другого размера) сетки, в которой каждая пара символов обрабатывается вместе. Пара символов заменяется другой парой, находящейся в той же строке, колонке или в их пересечении, что обеспечивает определенный уровень защиты. Этот метод сложнее по сравнению с Цезарем и Виженером и требует больше времени на шифрование и дешифрование данных. Он эффективен для защиты текстов, где требуется повышенная безопасность, но не подходит для современных приложений из-за ограниченного размера алфавита и менее гибкой структуры [7].

*AES (Advanced Encryption Standard).* AES является современным стандартом симметричного шифрования, который применяется для защиты данных во многих современных системах. Он обеспечивает высокий уровень безопасности благодаря использованию блоков фиксированной длины (128 бит) и ключей длиной 128, 192 или 256 бит. AES использует несколько раундов преобразований, включая замену байтов, перестановку и комбинацию ключей, что делает его устойчиивым к большинству современных атак, включая атаки грубой силы. AES используется в различных областях, включая защиту финансовых данных, коммуникаций и хранилищ конфиденциальной информации [8].

Метод Цезаря и шифр Виженера могут быть полезны для обучения и защиты незначительных данных, в то время как шифр Плейфера может использоваться для защиты текстов в ограниченных условиях. AES, в свою очередь, является стандартом для современных систем защиты данных благодаря своей высокой безопасности и широкому применению в различных областях. Выбор алгоритма зависит от требований к уровню защиты и объему обрабатываемых данных [9].

## **1.3 Техническая подготовка**

### **1.3.1 Выбор технологий и инструментов**

Visual Studio является мощной средой разработки, поддерживающей создание приложений на C# и .NET. Она предоставляет удобные инструменты для отладки, тестирования и оптимизации кода, что делает ее отличным выбором для разработки программного обеспечения.

Windows Forms идеален для создания настольных приложений с простыми и интуитивно понятными интерфейсами. Он поддерживает широкую кастомизацию элементов управления, что позволяет разрабатывать удобные интерфейсы для шифрования данных. Интеграция с .NET Framework обеспечивает доступ к библиотекам для работы с файлами и криптографией, включая алгоритм AES. Кроме того, Windows Forms легко поддерживать и обновлять, что обеспечивает долгосрочную стабильность приложения [10].

### **1.3.2 Архитектура приложения**

Общий обзор архитектуры.Приложение разработано с использованием Windows Forms, что обеспечивает удобный и интуитивно понятный интерфейс для пользователя. Архитектура приложения разделена на несколько ключевых модулей, каждый из которых выполняет свою функцию:

Пользовательский интерфейс (Windows Forms):

* + Основное окно приложения, содержащее текстовые поля (textBox1, textBox2, textBox3), кнопки (button1, button2, button3) и выпадающий список (comboBox1) для выбора алгоритма шифрования.
  + Диалоговые окна, такие как openFileDialog1 и saveFileDialog1, используются для выбора файлов и их сохранения.

Модуль управления алгоритмами шифрования:

* + Содержит классы и методы для выполнения различных криптографических операций, таких как шифрование и дешифрование. Реализованы алгоритмы, например, AES, DES или другие.

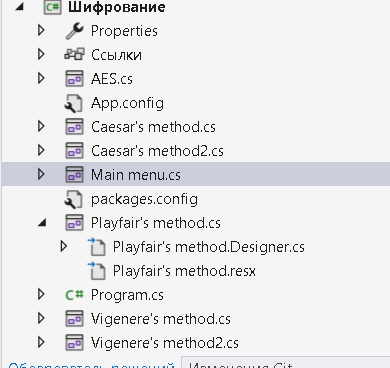


Рисунок 1. Архитектура проекта.

Модуль обработки файлов:

* + Обеспечивает работу с файлами — чтение данных из выбранных файлов и их запись после шифрования или дешифрования.

Логика приложения заключается в последовательном выполнении следующих шагов:

Выбор алгоритма шифрования:

* + Пользователь выбирает алгоритм шифрования из выпадающего списка (comboBox1). При выборе алгоритма приложение вызывает соответствующий метод из модуля управления алгоритмами шифрования.

Обработка ввода пользователя (текст или файл):

* + Если пользователь решает вводить текст вручную, данные из текстового поля (textBox1) передаются в модуль обработки алгоритма шифрования.
  + Если пользователь выбирает работу с файлом, приложение использует диалоговые окна (openFileDialog1 для открытия и saveFileDialog1 для сохранения файлов), чтобы загрузить и сохранить данные. Содержимое файла передается в метод шифрования/дешифрования.

Вывод результата (зашифрованный/расшифрованный текст):

* + Результат шифрования или дешифрования отображается в текстовом поле (textBox2) или записывается в файл, выбранный пользователем через диалоговое окно (saveFileDialog1).

Диалоговые окна и их роль:

Диалоговые окна играют важную роль в взаимодействии пользователя с приложением. Они обеспечивают удобный процесс выбора файлов и их сохранения:

*openFileDialog1:* Позволяет пользователю выбрать файл для чтения данных. После выбора файла его содержимое загружается в приложение для дальнейшей обработки.

*saveFileDialog1:* Предназначено для сохранения зашифрованного или расшифрованного файла. Пользователь выбирает папку и задает имя файла для сохранения результата.Пример работы приложения:

1. Пользователь запускает приложение и выбирает алгоритм шифрования.

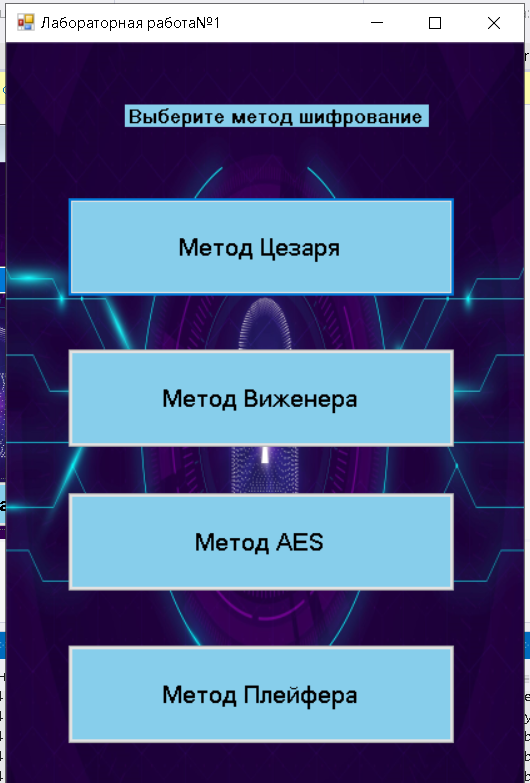


Рисунок 2. Главный экран проекта.

1. Вводит текст в поле или выбирает файл через openFileDialog1.
2. Нажимает кнопку для выполнения операции (например, button1 для шифрования).
3. После выполнения операции результат отображается в textBox2 или сохраняется в файл с помощью saveFileDialog1.
4. При необходимости, пользователь может просмотреть и сохранить зашифрованный или расшифрованный файл.

Эта структура позволяет легко расширять приложение, добавляя новые алгоритмы шифрования, улучшая интерфейс и интегрируя дополнительные функции работы с файлами.

# **2 РЕАЛИЗАЦИЯ**

## **2.1 Реализация метода Цезаря**

Метод Цезаря работает путем замены каждой буквы текста на букву, которая находится на фиксированное количество позиций дальше в алфавите. Мой код реализует этот алгоритм следующим образом:

private void button1\_Click(object sender, EventArgs e)

{

// Открытие диалога для выбора исходного файла

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Word Documents|\*.docx";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

// Чтение текста из исходного Word-файла

string sourceFilePath = openFileDialog.FileName;

string message = GetTextFromDocx(sourceFilePath);

// Получаем значение для сдвига из текстового поля

int k = Convert.ToInt32(textBox1.Text) % alphabet.Length;

string coded = string.Empty;

// Шифрование текста

foreach (char symbol in message)

{

if (symbol == empty)

{

coded += empty;

}

else

{

int index = alphabet.IndexOf(symbol);

if (index != -1)

{

int newIndex = (index + k) % alphabet.Length;

coded += alphabet[newIndex];

}

else

{

coded += symbol;

}

}

}

// Сохранение результата в новый Word-файл

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Word Documents|\*.docx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

string destinationFilePath = saveFileDialog.FileName;

SaveTextToDocx(destinationFilePath, coded);

MessageBox.Show("Зашифрованный текст успешно сохранен!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

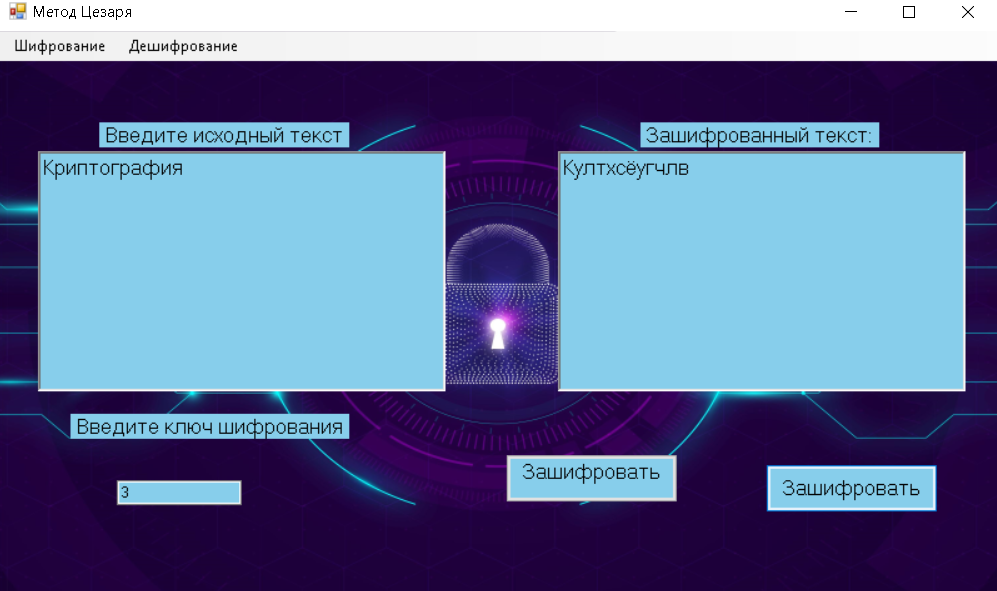


Рисунок 3. Метод Цезаря

**Объяснение кода**

*Инициализация переменных:* Строка alphabet содержит алфавит, который используется для шифрования текста. В данном случае это русский алфавит. Переменная empty представляет пробел, который не шифруется и добавляется в зашифрованный текст без изменений.

*Чтение текста из документа Word:* Метод GetTextFromDocx открывает .docx файл и извлекает его текст, передавая его в метод шифрования.

*Сдвиг и шифрование текста:* Значение сдвига k вводится пользователем и нормализуется с помощью k % alphabet.Length, чтобы избежать ошибок при больших значениях. Цикл foreach перебирает каждый символ текста. Если символ — пробел, он добавляется без изменений. Если символ найден в алфавите, его индекс сдвигается на k позиций, а новый индекс вычисляется как (index + k) % alphabet.Length. Символы, не входящие в алфавит (например, знаки препинания), не изменяются и добавляются в текст как есть.

*Сохранение зашифрованного текста:* Зашифрованный текст сохраняется в новый .docx файл с помощью метода SaveTextToDocx, после чего выводится сообщение об успешной записи.

*Пример шифрования:* Исходный текст-"привет мир", сдвиг – 3, получается зашифрованный текст - "сртгхс пкг".

*Особенности и ограничения:* Алфавит фиксирован, и только символы в нем могут быть зашифрованы. Метод работает только с кириллическими символами. Для других языков нужно изменить алфавит. Позиции символов вычисляются с помощью индексов, что упрощает сдвиг.

*Преимущества и недостатки:* Преимущества это простота реализации, подходит для обучения и небольших объемов данных. Недостатки - легко взломать методом перебора, ограничение на использование только заданного алфавита.

## **2.2 Код шифра Виженера**

Шифр Виженера — это метод шифрования, использующий ключ, состоящий из последовательности букв, чтобы зашифровать текст. В отличие от простого шифра Цезаря, где весь текст сдвигается на одно и то же количество позиций, шифр Виженера использует переменный сдвиг, зависящий от символов ключа.Вот реализация шифра Виженера на C#:

string[,] table = new string[,]

{

{"абвгдеёжзийклмнопрстуфхцчшщъыьэюя"},

{"бвгдеёжзийклмнопрстуфхцчшщъыьэюяа"},

// и так далее, сдвигая алфавит в каждой строке

};

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 4. Реализация метода Цезаря.

**Шифрование текста**:

* + Создание повторяющегося ключа для выравнивания его длины с длиной текста.
  + Циклическое прохождение по тексту и вычисление символов шифра с использованием таблицы Виженера.

for (int i = 0; i < text.Length; i++)

{

if (text[i] == empty)

{

done += ' ';

}

else

{

stroka = alpabet.IndexOf(text[i]);

stolbec = alpabet.IndexOf(list[i]);

done += table[stroka, 0][stolbec];

}

}

**Обработка нажатия кнопки**:

* + Считывание входного текста и ключа.
  + Запуск шифрования и вывод результата.

private void button1\_Click(object sender, EventArgs e)

{

string text = Convert.ToString(richTextBox1.Text);

string shifr = Convert.ToString(textBox1.Text);

}

Эти фрагменты демонстрируют ключевую логику и работу программы шифрования текста методом Виженера.

* 1. **Алгоритм Плейфера**

Шифр Плейфэра — это классический метод шифрования текста, использующий матрицу 5x5, заполненную буквами алфавита. Важно помнить, что буква J обычно заменяется на I для удобства. Рассмотрим, как реализовать данный алгоритм в коде на языке C#.

Создание ключевой матрицы:

Первая задача — создать матрицу 5x5, где каждая клетка будет содержать символ из ключа и алфавита. Символы не повторяются, и матрица заполняется уникальными буквами.

**Шифрование и расшифрование**

В зависимости от положения символов в матрице применяются следующие правила:

* Одинаковая строка: Символы сдвигаются вправо при шифровании и влево при расшифровании.
* Одинаковый столбец: Символы сдвигаются вниз при шифровании и вверх при расшифровании.
* Разные строки и столбцы: Символы меняются местами по диагонали.

private string Cipher(string input, bool decode)

{

string result = "";

for (int i = 0; i < input.Length; i += 2)

{

int[] pos1 = GetPosition(input[i]);

int[] pos2 = GetPosition(input[i + 1]);

if (pos1[0] == pos2[0]) {

// Одинаковая строка: сдвигаем вправо/влево.

result += keyMatrix[pos1[0], (pos1[1] + (decode ? -1 : 1) + 5) % 5];

result += keyMatrix[pos2[0], (pos2[1] + (decode ? -1 : 1) + 5) % 5];

} else if (pos1[1] == pos2[1]) {

// Одинаковый столбец: сдвигаем вверх/вниз.

result += keyMatrix[(pos1[0] + (decode ? -1 : 1) + 5) % 5, pos1[1]];

result += keyMatrix[(pos2[0] + (decode ? -1 : 1) + 5) % 5, pos2[1]];

} else {

// Разные строки и столбцы: меняем символы по диагонали.

result += keyMatrix[pos1[0], pos2[1]];

result += keyMatrix[pos2[0], pos1[1]];

}

}

return result;

}

*Пояснение:* Метод шифрования и расшифрования обрабатывает каждую пару символов и применяет соответствующее правило. Сдвиг с учетом границ матрицы достигается с помощью оператора %.

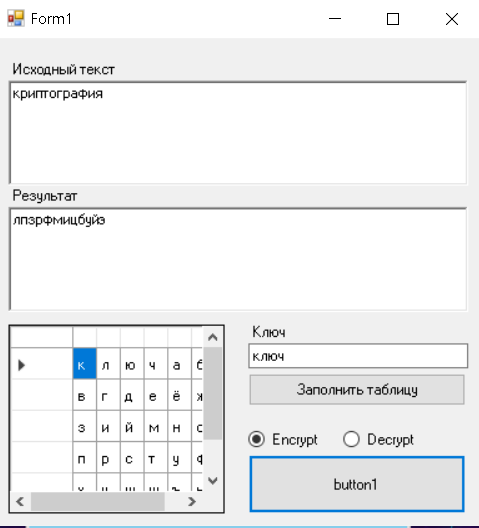


Рисунок 5. Метод Плейфера.

* 1. **Пример использования AES для работы с файлами.**

**Инициализация шифра AES**

Перед началом шифрования или дешифрования необходимо создать объект Aes и задать ключ и вектор инициализации (IV). Эти параметры обеспечивают безопасность шифрования и должны быть одинаковыми при шифровании и дешифровании данных.

using System.Security.Cryptography;

Aes aes = Aes.Create();

aes.Key = Encoding.UTF8.GetBytes(key); // 16, 24 или 32 байта для ключа

aes.IV=Encoding.UTF8.GetBytes(iv); //16 байт для IV

**Шифрование данных**

Метод шифрования принимает входные данные и преобразует их в зашифрованный формат. Для этого используется CryptoStream, который позволяет записывать зашифрованные данные в поток.

public byte[] Encrypt(byte[] data, string key, string iv)

{

using (Aes aes = Aes.Create())

{

aes.Key = Encoding.UTF8.GetBytes(key);

aes.IV = Encoding.UTF8.GetBytes(iv);

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, aes.CreateEncryptor(), CryptoStreamMode.Write))

{

cs.Write(data, 0, data.Length);

}

return ms.ToArray();

}

}

}

**Объяснение:**

* aes.CreateEncryptor() создает объект шифрования, используя заданные ключ и IV.
* CryptoStream используется для записи зашифрованных данных в поток памяти.

**Дешифрование данных**

Метод дешифрования выполняет обратную операцию — преобразует зашифрованные данные обратно в исходный формат.

public byte[] Decrypt(byte[] data, string key, string iv)

{

using (Aes aes = Aes.Create())

{

aes.Key = Encoding.UTF8.GetBytes(key);

aes.IV = Encoding.UTF8.GetBytes(iv);

using (MemoryStream ms = new MemoryStream(data))

{

using (CryptoStream cs = new CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Read))

{

using (MemoryStream result = new MemoryStream())

{

cs.CopyTo(result);

return result.ToArray();

}

}

}

}

}

Объяснение:

* aes.CreateDecryptor() создает объект для дешифрования.
* CryptoStream используется для чтения и расшифровки данных из входного потока.

Изображение выглядит как текст, снимок экрана, Цвет электрик, Графика

Автоматически созданное описание

Рисунок 6. Шифр AES для работы с файлами.

**Работа с файлами**

Чтобы работать с файлами, нужно использовать методы чтения и записи файлов в комбинации с шифрованием и дешифрованием:

Изображение выглядит как текст, снимок экрана, Цвет электрик, Шрифт

Автоматически созданное описание

Рисунок 7. Выбрать тип файла.

**Шифрование файла:**

public void EncryptFile(string inputFilePath, string outputFilePath, string key, string iv)

{

byte[] fileBytes = File.ReadAllBytes(inputFilePath);

byte[] encryptedBytes = Encrypt(fileBytes, key, iv);

File.WriteAllBytes(outputFilePath, encryptedBytes);

}

**Дешифрование файла:**

public void DecryptFile(string inputFilePath, string outputFilePath, string key, string iv)

{

byte[] encryptedBytes = File.ReadAllBytes(inputFilePath);

byte[] decryptedBytes = Decrypt(encryptedBytes, key, iv);

File.WriteAllBytes(outputFilePath, decryptedBytes);

}

Объяснение**:**

* File.ReadAllBytes() читает все байты из файла, создавая массив байтов.
* File.WriteAllBytes() записывает байты обратно в файл.
* Методы EncryptFile и DecryptFile используют функции шифрования и дешифрования для обработки файлов.

**Важные замечания:**

* *Длина ключа и IV:* Ключ и IV должны быть определенной длины. Для AES ключ может быть длиной 16, 24 или 32 байта, а IV всегда должен быть 16 байт.
* *Безопасность ключа и IV:* Никогда не храните ключ и IV в коде в открытом виде. Используйте безопасные методы хранения, такие как защищенные хранилища ключей.
* *Протокол безопасности:* AES должен использоваться с безопасным режимом, например, CBC (Cipher Block Chaining) или GCM (Galois/Counter Mode).

Эти функции и методы обеспечивают базовые операции шифрования и дешифрования данных в формате AES для работы с файлами и могут быть расширены для более сложных применений, включая защиту ключей и оптимизацию производительности.

# **ЗАКЛЮЧЕНИЕ**

Разработка и тестирование системы шифрования на основе алгоритма AES с использованием интерфейса Windows Forms позволила создать эффективное и надежное решение для защиты данных. В современном цифровом мире обеспечение конфиденциальности и безопасности данных становится необходимостью, и разработанное приложение удовлетворяет эти потребности, предоставляя пользователям удобный инструмент для защиты их информации.

С увеличением объемов передаваемой и хранимой информации защита данных становится одной из ключевых задач. Данные могут содержать чувствительную информацию, которая может быть использована с целью шантажа, кражи личности или других преступных действий. Приложение для шифрования данных, основанное на алгоритме AES, обеспечивает высокий уровень безопасности благодаря своей криптографической стойкости и широкому признанию в разных областях.

Приложение обладает рядом значительных преимуществ, которые делают его востребованным и полезным для разных категорий пользователей. Оно обеспечит высокий уровень защиты данных благодаря использованию алгоритма AES, признанного одним из наиболее надежных. Это особенно важно для защиты конфиденциальной информации в бизнесе, где утечка данных может привести к финансовым потерям и репутационным рискам. В частных целях приложение поможет пользователям шифровать личные данные, такие как фотографии, документы и переписку, что обеспечит их конфиденциальность и безопасность.

Государственные учреждения и образовательные организации также могут извлечь пользу из приложения, так как оно поможет защищать данные студентов и преподавателей, а также облегчить безопасный обмен документами между структурами. Для разработчиков программного обеспечения приложение станет важным инструментом, позволяющим интегрировать базовые функции защиты данных в их проекты.

Будущее развитие приложения направлено на расширение его функциональности и повышение удобства использования. Планируется добавление поддержки дополнительных алгоритмов шифрования, таких как RSA и ECC, что обеспечит пользователям возможность выбора оптимальных методов защиты данных. Внедрение цифровых подписей и сертификатов повысит безопасность, предоставляя возможность подтверждения подлинности файлов. Автоматизация процессов через планировщик задач улучшит удобство работы, позволяя автоматическое шифрование и дешифрование по расписанию. Разработка мобильной версии расширит доступность приложения на смартфонах и планшетах, а интеграция с облачными сервисами обеспечит безопасность данных, передаваемых через интернет. Введение функций аналитики и отчетов позволит пользователям отслеживать безопасность данных и повышать прозрачность работы системы.

В заключение, данное приложение является универсальным инструментом для обеспечения безопасности данных и защиты конфиденциальной информации. Оно подходит как для профессионального использования в бизнесе, так и для индивидуальных пользователей, стремящихся защитить свои личные данные. Планы по дальнейшему расширению функционала, включая поддержку мобильных устройств и интеграцию с облачными сервисами, только укрепляют его значимость в современном цифровом мире.

# 

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Rajaboyevich G. S. et al. Characterizing honeypot-captured cyber-attacks: statistical framework and case study. – 2023.
2. Иванов В. В., Трубникова Т. В., Чурилов А. Ю. Правовая защита частных интересов в цифровом мире: позитивное и/или «мягкое право»? Вестник Томского государственного университета. – 2024. – №. 502. – С. 188-203.
3. Tuo Z. A comparative Analysis of AES and RSA algorithms and their integrated application //Theoretical and Natural Science. – 2023. – Т. 25. – С. 28-35
4. Аннеков Р., Аширов Б., Азадов Ю. ШИФРОВАНИЕ ДАННЫХ: НЕОБХОДИМОСТЬ И МЕТОДЫ РЕАЛИЗАЦИИ //Вестник науки. – 2024. – Т. 3. – №. 10 (79). – С. 807-810.
5. Utomo K. B., Hakim A. R., Cahyono B. Development of an encryption algorithm based on the Caesar Cipher algorithm //BIS Information Technology and Computer Science. – 2024. – Т. 1. – С. V124001-V124001
6. Чеченец В. А. Шифрование как способ защиты корпоративной информации. – 2023.
7. Miroshnichenko A. S., Mikhelev V. M. Segmentation of MRI images in classification tasks //Марчуковские научные чтения. – 2020. – №. 2020. – С. 154-154.
8. Фараносов Д. М. Шифратор на основе AES. – 2022
9. Зацепа А. И. Шифрование данных //StudNet. – 2022. – Т. 5. – №. 1. – С. 344-351.
10. Sharp J. Microsoft Visual C# step by step. – Microsoft Press, 2022.