

Документация

Цель проекта: Разработать программное средство для безопасной передачи информации между отправителем и получателем, используя криптографические алгоритмы RSA и AES для шифрования данных. Проект направлен на обеспечение конфиденциальности информации при её передаче, гарантируя защиту от несанкционированного доступа и поддерживая целостность данных этапах обмена.

Оглавление

- **Страница 1**
 - Основные задачи 2
- **Страницы 3–6**
 - Техническое описание 3–7
- **Страницы 7–16**
 - Описание кода 8–16
- **Страницы 17–21**
 - Инструкция по использованию 17–22
- **Страница 23**
 - Заключение 23

Основные задачи:

- Двухэтапное шифрование данных :
 - Реализация последовательного шифрования текста: сначала с использованием алгоритма RSA, затем с применением алгоритма AES.
 - Обеспечение многоуровневой защиты данных за счёт комбинирования асимметричного (RSA) и симметричного (AES) шифрования.
- Шифрование текста алгоритмом RSA :
 - Преобразование исходного текста в зашифрованный формат с использованием публичного ключа RSA.
 - Гарантия безопасности данных на этапе первичного шифрования.
- Дополнительное шифрование RSA-данных алгоритмом AES :
 - Шифрование результата работы RSA с помощью симметричного алгоритма AES.
 - Повышение безопасности передаваемых данных за счёт добавления второго уровня шифрования.
- Двухэтапное дешифрование данных :
 - Реализация обратного процесса: сначала дешифрование данных алгоритмом AES, затем дешифрование результата алгоритмом RSA.
 - Восстановление исходного текста из зашифрованного формата.
- Генерация и управление ключами :

- Создание пары ключей RSA (публичный и приватный) для шифрования и дешифрования данных.
- Генерация ключа AES для шифрования и дешифрования данных.
- Удобное управление ключами :
 - Создание профилей для сохранения множества ключей, предназначенных для разных собеседников или пользователей.
 - Упрощение работы с ключами за счёт их организации и хранения в профилях.
- Сохранение зашифрованной информации :
 - Сохранение зашифрованных данных в текстовый файл для их последующей передачи собеседнику.
 - Обеспечение удобства обмена зашифрованной информацией между пользователями.

Техническое описание

- **Используемые технологии**
 - Проект разработан с использованием следующих технологий:
- Язык программирования : C#
- Фреймворк : .NET MAUI (Multi-platform App UI), который позволяет создавать кроссплатформенные приложения для Windows, macOS, iOS и Android.
- Библиотеки :
 - **System.Security.Cryptography**: Используется для реализации алгоритмов шифрования AES.
 - **System.Numerics**: Предоставляет поддержку работы с большими числами (например, для операций с ключами RSA).
 - **System.IO**: Используется для работы с файлами (например, сохранение зашифрованных данных в текстовый файл).
 - **System.Text**: Используется для обработки строк и преобразования данных между различными форматами (например, текст → байты).
 - **Microsoft.Maui.Controls** и **Microsoft.Maui.Storage**: Используются для создания пользовательского интерфейса и управления локальным хранилищем данных (например, профилей с ключами).

Архитектура программы

Программа построена с использованием объектно-ориентированного программирования (ООП) и соответствует принципам SOLID, что обеспечивает гибкость, масштабируемость и удобство поддержки кода.

1. Применение принципов ООП

- В проекте активно используются следующие концепции ООП:
- Инкапсуляция :
 - Все данные и методы для работы с шифрованием, дешифрованием и ключами инкапсулированы в соответствующих классах (например, RSA_Encrypt, AES_Decrypt).
 - Это позволяет скрыть сложную логику от пользователя и предоставить простой интерфейс для взаимодействия.
- Наследование :

Кассы на странице в интерфейсе хaml наследуют заданный размер шрифта

- **Полиморфизм :**
 - Используется для создания универсального интерфейса шифрования. Например, метод `OnEncryptClicked` может быть переопределён в классах `RSA_Encrypt` и `Decrypt`, чтобы реализовать разные алгоритмы шифрования.
- **Абстракция :**
 - Взаимодействие пользователя с интерфейсом, скрывая под капотом сложную логику приложения и нюансы приложения

2. Принципы SOLID

Проект соответствует пяти принципам SOLID, что делает его более гибким и удобным для поддержки:

- **Single Responsibility Principle (Принцип единственной ответственности) :**
 - Каждый класс выполняет одну задачу. Например:
 - `AES_Encrypt` отвечает только за шифрование данных.
 - `GenerateTwoLargePrimes` управляет генерацией, хранением и загрузкой ключей.Это упрощает тестирование и модификацию кода.
- **Open/Closed Principle (Принцип открытости/закрытости) :**
 - Классы открыты для расширения, но закрыты для изменения. Например:
 - Если нужно добавить новый алгоритм шифрования (например, DES), достаточно создать новый класс, реализующий интерфейс **ICryptoService**, без изменения существующего кода.
- **Liskov Substitution Principle (Принцип подстановки Лисков) :**
 - Любой класс, реализующий интерфейс может быть заменён другим классом без нарушения работы программы. Например, `RSA_Encrypt` и `AES_Encrypt` могут использоваться взаимозаменяемо.
- **Interface Segregation Principle (Принцип разделения интерфейса) :**
 - Интерфейсы разделены на небольшие, специализированные части. Например:
 - `SaveFileButtonEncrypt` отвечает за работу с файлами.
 - `SaveKeysForProfile` управляет хранением ключей.
 - Это предотвращает создание "раздутых" интерфейсов.
- **Dependency Inversion Principle (Принцип инверсии зависимостей) :**
 - Высокоуровневые модули зависят от абстракций, а не от конкретных реализаций. Например:
 - Класс `OnEncryptClicked` зависит от интерфейса `InputTextEntry`, а не от конкретного класса `AES_Encrypt`.

- Это позволяет легко заменять реализации (например, использовать моки для тестирования).

Программа построена по модульному принципу, что обеспечивает удобство разработки, тестирования и поддержки. Основные компоненты архитектуры:

- Пользовательский интерфейс (UI) :
 - Создан с использованием XAML и Microsoft.Maui.Controls .
 - Интерфейс включает поля для ввода текста, отображения результатов шифрования/дешифрования, кнопки для выполнения операций и элементы управления профилями ключей, кнопки для сохранения зашифрованного текста, генерации ключей RSA, AES, удаление профилей
- Логика шифрования и дешифрования :
 - Реализована в отдельных классах или методах для обеспечения чистоты кода и повторного использования.
 - Включает два уровня шифрования:
 - Первый уровень: Шифрование текста с использованием алгоритма RSA .
 - Второй уровень: Шифрование результата RSA с использованием алгоритма AES .
- Управление ключами :
 - Ключи RSA генерируются с помощью классов из пространства имён **System.Security.Cryptography**.
 - Ключ AES генерируется случайным образом
 - Профили ключей сохраняются в локальном хранилище с использованием **Microsoft.Maui.Storage**.
- Работа с файлами :
 - Зашифрованные данные сохраняются в текстовые файлы с использованием **System.IO**.
 - Файлы могут быть переданы другим пользователям для последующего дешифрования.

Описание работы программы

- Программа реализует двухэтапный процесс шифрования и дешифрования текста с использованием алгоритмов RSA и AES. Ниже приведено пошаговое описание логики работы программы.

1. Создание и управление ключами

- Генерация ключей RSA :
 - Пользователь может создать новую пару ключей RSA (публичный и приватный) через интерфейс приложения.

- Ключи генерируются автоматически с использованием библиотеки **System.Security.Cryptography**.
- Публичный ключ используется для шифрования, а приватный — для дешифрования.
- Генерация ключа AES :
 - Программа позволяет генерировать случайный ключ AES для каждого сеанса шифрования.
 - Ключ AES является симметричным и используется для шифрования текста.
- Создание профиля :
 - Пользователь может создать отдельный профиль для хранения ключей (например, для разных собеседников или пользователей).
 - В профиле сохраняются:
 - Пара ключей RSA (публичный и приватный).
 - Ключ AES (если он был сгенерирован вручную).
- Управление профилями :
 - Пользователь может создавать, удалять и выбирать профили через интерфейс.
 - Каждый профиль хранится отдельно, что позволяет разграничивать ключи для разных пользователей.

2. Шифрование данных

- Ввод текста :
 - Пользователь вводит текст, который нужно зашифровать, в соответствующее поле интерфейса.
- Выбор профиля :
 - Пользователь выбирает профиль, содержащий ключи RSA и AES.
- Генерация или использование ключа AES :
 - Если ключ AES еще не был создан, программа генерирует новый случайный ключ AES.
 - Если ключ AES уже существует (например, был сохранен в профиле), он используется для шифрования.
- Шифрование текста с помощью AES :
 - Введенный текст шифруется с использованием ключа AES.
 - Результатом является зашифрованный текст.
- Шифрование ключа AES с помощью RSA :
 - Сгенерированный ключ AES шифруется с использованием публичного ключа RSA из выбранного профиля.
 - Это обеспечивает безопасную передачу ключа AES вместе с зашифрованным текстом.
- Сохранение результатов :
 - Зашифрованный текст и зашифрованный ключ AES могут быть сохранены в текстовый файл для последующей передачи получателю.

3. Передача данных

- Зашифрованный текст и зашифрованный ключ AES передаются получателю через выбранный канал (например, электронная почта, мессенджер или обмен файлами).

4. Дешифрование данных

- Получение данных :
 - Получатель загружает зашифрованный текст и зашифрованный ключ AES из файла или другого источника.
- Выбор профиля :
 - Получатель выбирает профиль, содержащий приватный ключ RSA.
- Дешифрование ключа AES :
 - Зашифрованный ключ AES расшифровывается с использованием приватного ключа RSA из выбранного профиля.
- Дешифрование текста :
 - Расшифрованный ключ AES используется для дешифрования зашифрованного текста.
 - Результатом является исходный текст.
- Отображение результата :
 - Расшифрованный текст отображается в интерфейсе для просмотра пользователем.

5. Сохранение результатов

- Пользователь может сохранить результаты (зашифрованный или расшифрованный текст) в текстовый файл для дальнейшего использования.

Описание кода

Функция принимает публичный, приватные ключи и текст, который нужно зашифровать алгоритмом RSA

```
public static string RSA_Encrypt(string PublicKeyNEntry, string PrivateKeyDEntry, string
inputText)//принимает аргументы

{

    try

    {

        Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

        Encoding win1251 = Encoding.GetEncoding(1251); //Реистрируем кодировку 1251, чтобы
можно было перевести из текста в неё

        // Проверка ключей RSA

        if (!BigInteger.TryParse(PublicKeyNEntry, out BigInteger publicKeyN) ||

            !BigInteger.TryParse(PrivateKeyDEntry, out BigInteger privateKeyE))

        {

            return "Ошибка, некорректный ключ RSA";

        }

        string inputtext = inputText;

        // Проверка входного текста

        if (string.IsNullOrEmpty(inputText))

        {

            return "Введите текст для шифрования.";

        }

        List<BigInteger> encryptedValues = new List<BigInteger>();// создание нового списка
BigInteger(числа с произвольной длиной)
```

```
byte[] bytes = win1251.GetBytes(inputText);
```

```
foreach (byte b in bytes)
```

```
{
```

```
    BigInteger asciiValue = b;
```

```
    if (asciiValue >= publicKeyN)// проверка что символ не выходит за пределы кодировки
```

```
    {
```

```
        return $"Символ '{(char)b}' слишком большой для шифрования.";
```

```
    }
```

```
    BigInteger encryptedValue = BigInteger.ModPow(asciiValue, privateKeyE, publicKeyN);//
```

возведение переведенное числа из символа в число в степень приватного ключа и сразу деление его по модулю на публичный ключ

```
    encryptedValues.Add(encryptedValue);// Добавление рассчитанного набора цифр в список
```

```
}
```

```
return string.Join(" ", encryptedValues);
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
    return $"Ошибка RSA шифрования: {ex.Message}";// обработка ошибок
```

```
}
```

```
}
```

Функция принимает публичный, приватные ключи и текст, который нужно расшифровать алгоритмом RSA

```
public static string RSA_Decrypt(string PublicKeyNEntry, string PrivateKeyDEntry, string inputText)
```

```
{
```



```

try
{
    Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

    // Проверка ключей RSA
    if (!BigInteger.TryParse(PublicKeyNEntry, out BigInteger publicKeyN) ||
        !BigInteger.TryParse(PrivateKeyDEntry, out BigInteger privateD))
    {
        return "Ошибка, некорректный ключ RSA";
    }
    StringBuilder finalMessage = new StringBuilder();// создаем изменяемую строку, куда будем
сохранять расшифровку текста
    string words = inputText;

    if (string.IsNullOrEmpty(words))
    {
        return "Зашифрованный текст пуст.";
    }

    Encoding win1251 = Encoding.GetEncoding(1251);// Регистрируем кодировку 1251
    string[] numbers = words.Trim().Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);//
удаляем пробелы в начале и в конце, разделяем строку по пробелам

    foreach (string number in numbers)
    {
        // Проверка корректности числа
        if (!BigInteger.TryParse(number, out BigInteger num))
        {
            return $"Некорректное число: {number}";
        }

        BigInteger decryptedValue = BigInteger.ModPow(num, privateD, publicKeyN);// возведение
переведенное числа из символа в число в степень приватного ключа и сразу деление его по модулю
на публичный ключ

        if (decryptedValue >= 0 && decryptedValue <= 255)// Проверка на то, чтобы число не
выходило за пределы чисел кодировка 1251
        {
            byte byteValue = (byte)decryptedValue;// Переводим из числового значение обратно в
символы, тобиш текст
            finalMessage.Append(win1251.GetString(new byte[] { byteValue }));// добавляем в
изменяемую строку результат
        }
        else
        {
            return $"Некорректное значение ключа";
        }
    }
}

```

```

    }

    return finalMessage.ToString();
}
catch (Exception ex)
{
    return $"Ошибка RSA расшифрования: {ex.Message}";
}
}

```

Функция принимает вектор инициализации, набор байтов и текст, который нужно зашифровать алгоритмом AES

```

public static string AES_Encrypt(string keyString, string ivString, string inputText)
{
    try
    {
        string Inputtext = inputText;

        // Проверка входного текста на пустоту
        if (string.IsNullOrEmpty(Inputtext))
        {
            return "Введите текст для шифрования."; // Если текст пустой, возвращаем сообщение об
ошибке
        }

        string KeyStringAES = keyString;

        string IVString = ivString;

        // Проверка ключа и вектора инициализации (IV) на пустоту
        if (string.IsNullOrEmpty(keyString) || string.IsNullOrEmpty(ivString))

```

```

    {

        return "Введите ключ и IV для AES."; // Если ключ или IV пустые, возвращаем сообщение
        об ошибке

    }


    // Преобразование строки ключа из шестнадцатеричного формата в массив байтов

    byte[] key = Convert.FromHexString(KeyStringAES.Replace("-", "")); // Убираем дефисы и
    преобразуем в байты

    byte[] iv = Convert.FromHexString(IVString.Replace("-", "")); // То же самое для IV


    using (Aes aes = Aes.Create()) // Создаем экземпляр AES

    {

        aes.Key = key; // Устанавливаем ключ

        aes.IV = iv; // Устанавливаем вектор инициализации (IV)


        using (MemoryStream msEncrypt = new MemoryStream()) // Создаем поток для записи
        зашифрованных данных

        {

            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, aes.CreateEncryptor(),
            CryptoStreamMode.Write))

            {

                byte[] dataToEncrypt = Encoding.UTF8.GetBytes(Inputtext); // Преобразуем входной
                текст в массив байтов

                csEncrypt.Write(dataToEncrypt, 0, dataToEncrypt.Length); // Записываем данные в
                поток для шифрования

                csEncrypt.FlushFinalBlock(); // Завершаем процесс шифрования

                return Convert.ToBase64String(msEncrypt.ToArray()); // Возвращаем зашифрованный
                текст в формате Base64

            }

        }

    }

```

```

    }

}

catch (Exception ex)

{

    return $"Ошибка AES шифрования: {ex.Message}"; // Обработка ошибок

}

}

```

Функция принимает вектор инициализации, набор байтов и текст, который нужно расшифровать алгоритмом AES

```

public static string AES_Decrypt(string keyString, string ivString, string inputText)
{
    try
    {
        string Inputtext = inputText;

        // Проверка зашифрованного текста на пустоту
        if (string.IsNullOrEmpty(Inputtext))
        {
            return "Введите текст для расшифрования."; // Если текст пустой, возвращаем сообщение
об ошибке
        }

        string keyStringAES = keyString;
        string IVString = ivString;

        // Проверка ключа и IV на пустоту
        if (string.IsNullOrEmpty(keyString) || string.IsNullOrEmpty(ivString))
        {
            return "Введите ключ и IV для AES."; // Если ключ или IV пустые, возвращаем сообщение
об ошибке
        }
    }
}

```

```

    }

    // Преобразование строки ключа из шестнадцатеричного формата в массив байтов
    byte[] key = Convert.FromHexString(keyString.Replace("-", "")); // Убираем дефисы и
    преобразуем в байты

    byte[] iv = Convert.FromHexString(ivString.Replace("-", "")); // То же самое для IV

    using (Aes aes = Aes.Create()) // Создаем экземпляр AES
    {
        aes.Key = key; // Устанавливаем ключ

        aes.IV = iv; // Устанавливаем вектор инициализации (IV)

        using (MemoryStream msDecrypt = new
        MemoryStream(Convert.FromBase64String(Inputtext))) // Преобразуем Base64-строку в байты
        {
            using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, aes.CreateDecryptor(),
            CryptoStreamMode.Read)) // Создаем поток для расшифровки
            {
                using (StreamReader srDecrypt = new StreamReader(csDecrypt)) // Читаем расшифрованные
                данные
                {
                    return srDecrypt.ReadToEnd(); // Возвращаем расшифрованный текст
                }
            }
        }
    }
    catch (Exception ex)
    {
        return $"Ошибка AES расшифрования: {ex.Message}"; // Обработка ошибок
    }
}

```

Функции, которые исходя из количество битов, которое выбирает пользователь, создают ключи RSA

```

public static BigInteger GenerateLargeRandomNumber(int length)
{
    // Преобразуем длину в количество байт
    int byteLength = (length + 7) / 8; // Длина в байтах (округляется вверх)

    byte[] bytes = new byte[byteLength];
}

```

```

using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
{
    rng.GetBytes(bytes); // Генерация случайных байтов
}
return BigInteger.Abs(new BigInteger(bytes)); // Возвращаем положительное число
}

public static (BigInteger, BigInteger, BigInteger) GenerateTwoLargePrimes(int bits)
{
    BigInteger prime1 = GenerateLargePrime(bits); // Генерация первого простого числа
    BigInteger prime2 = GenerateLargePrime(bits); // Генерация второго простого числа

    // Public key: N = prime1 * prime2
    BigInteger PublicKey = prime1 * prime2; // Вычисление публичного ключа

    // F(N) = (prime1 - 1) * (prime2 - 1)
    BigInteger F_n_ = (prime1 - 1) * (prime2 - 1); // Функция Эйлера

    // Mutually Prime Number (E)
    BigInteger PrivatKey1;

    // Генерация взаимно простого числа с F(N)
    do
    {
        PrivatKey1 = GenerateLargeRandomNumber(1000); // Генерация случайного числа длиной 1000
        бит
    } while (NOD(F_n_, PrivatKey1) != 1); // Проверка, что число взаимно просто с F(N)

    // Private key (D)
    BigInteger PrivatKey2;
    if (TryModInverse(PrivatKey1, F_n_, out PrivatKey2))
    {
        // Возвращаем публичный и приватные ключи
        return (PublicKey, PrivatKey1, PrivatKey2);
    }

    throw new Exception("Не удалось найти обратный элемент для приватного ключа.");
}

public static BigInteger NOD(BigInteger a, BigInteger b)
{
    while (b != 0) // Алгоритм Евклида для нахождения НОД
    {
        BigInteger temp = b;
        b = a % b;
        a = temp;
    }
    return a; // Возвращаем наибольший общий делитель
}

```

```

// Алгоритм нахождения обратного числа по модулю (расчет приватного ключа)
private static bool TryModInverse(BigInteger a, BigInteger m, out BigInteger result)
{
    BigInteger m0 = m, t, q;
    BigInteger x0 = 0, x1 = 1;

    if (m == 1)
    {
        result = 0;
        return false;
    }

    while (a > 1) // Расширенный алгоритм Евклида
    {
        q = a / m;
        t = m;

        m = a % m;
        a = t;
        t = x0;

        x0 = x1 - q * x0;
        x1 = t;
    }

    if (x1 < 0)
        x1 += m0;

    result = x1; // Возвращаем обратный элемент
    return true;
}

public static BigInteger GenerateLargePrime(int bits)
{
    Random random = new Random();
    while (true)
    {
        byte[] bytes = new byte[bits / 8];
        random.NextBytes(bytes); // Генерация случайных байтов
        bytes[bytes.Length - 1] |= 0x01; // Устанавливаем старший бит в 1 (для нечетности)
        BigInteger candidate = new BigInteger(bytes);
        if (IsProbablePrime(candidate, 10)) // Проверка на вероятность простоты
        {
            return candidate; // Возвращаем простое число
        }
    }
}

```

```

static bool IsProbablePrime(BigInteger number, int certainty)
{
    if (number <= 1) return false; // Число должно быть больше 1
    if (number <= 3) return true; // 2 и 3 — простые числа
    if (number % 2 == 0) return false; // Четные числа не являются простыми

    BigInteger d = number - 1;
    int r = 0;
    while (d % 2 == 0) // Находим d и r для теста Миллера-Рабина
    {
        d /= 2;
        r++;
    }

    Random random = new Random();

    for (int i = 0; i < certainty; i++) // Проводим тест Миллера-Рабина
    {
        BigInteger a;
        do
        {
            byte[] bytes = new byte[number.GetByteCount()];
            random.NextBytes(bytes);
            bytes[bytes.Length - 1] |= 0x01; // Число нечетное
            bytes[0] &= 0x7F; // Старший бит равен 0 для положительных чисел
            a = new BigInteger(bytes);
        } while (a < 2 || a >= number - 1);

        BigInteger x = BigInteger.ModPow(a, d, number); // Вычисляем a^d mod number
        if (x == 1 || x == number - 1) continue;

        bool continueOuter = false;
        for (int j = 0; j < r - 1; j++)
        {
            x = BigInteger.ModPow(x, 2, number); // Повторяем возведение в квадрат
            if (x == number - 1)
            {
                continueOuter = true;
                break;
            }
        }
        if (continueOuter) continue;

        return false; // Если тест не пройден, число не является простым
    }
    return true; // Если все тесты пройдены, число вероятно простое
}

```


Функция которая создает AES ключ

```
public static (string Key, string IV) CreateKeyAESandIV()
{
    using (Aes aes = Aes.Create())
    {
        aes.KeySize = 256;
        aes.GenerateKey();
        aes.GenerateIV();
        string keyString = Convert.ToBase64String(aes.Key);
        string ivString = Convert.ToBase64String(aes.IV);
        string keyHex = BitConverter.ToString(aes.Key);
        string ivHex = BitConverter.ToString(aes.IV);
        return (keyHex, ivHex);
    }
}

//create key AES
```

Инструкция по использованию

При запуске приложения вы попадаете в главное меню

Encrypt

Шифрование

Введите текст для шифрования

Публичный ключ RSA

Приватный ключ RSA

Ключ AES (16/24/32 байта)

ключ IV (16 байт)

Зашифровать

Результат шифрования

Введите текст для расшифрования

Расшифровать

Результат расшифрования

Введите название нового профиля:

Имя профиля

Создать профиль

Удалить профиль

Выберите профиль для смены ключей

Введите текст для расшифрования

Расшифровать

Результат расшифрования

Введите название нового профиля:

Имя профиля

Создать профиль

Удалить профиль

Выберите профиль для смены ключей

1

Выберите профили для удаления

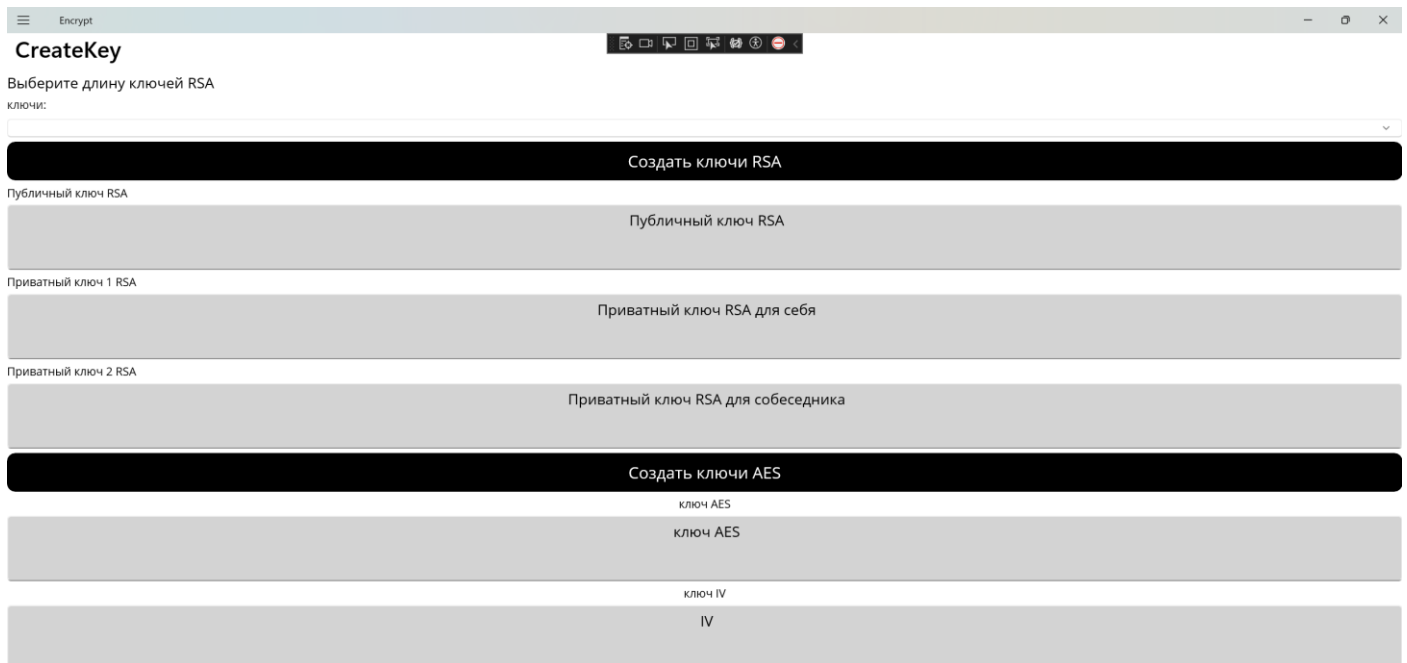
3

4

aboba

Чтобы создать ключи RSA и AES, выполните следующие действия:

1. Нажмите на значок меню (три полосы), расположенный в верхнем левом углу экрана.
2. В открывшемся меню выберите пункт "Создание ключей" , чтобы перейти на соответствующую страницу.



Encrypt

CreateKey

Выберите длину ключей RSA
ключи:

Создать ключи RSA

Публичный ключ RSA

Публичный ключ RSA

Приватный ключ 1 RSA

Приватный ключ RSA для себя

Приватный ключ 2 RSA

Приватный ключ RSA для собеседника

Создать ключи AES

ключ AES

ключ AES

ключ IV

IV

Выберите длину ключа RSA и нажмите создать ключи

512 бит, минимальная длина, используется только для тестов, не рекомендуется для реальных задач

1024 бит, устаревший стандарт, слабая защита, подходит только для ограниченных сценариев

2048 бит, рекомендуемый стандарт для современных приложений, сбалансированная безопасность и производительность

4096 бит, высокая безопасность, подходит для долгосрочной защиты данных. Требуется много вычислительной мощности

Публичный ключ RSA

Публичный ключ RSA

Приватный ключ 1 RSA

Приватный ключ RSA для себя

Приватный ключ 2 RSA

Приватный ключ RSA для собеседника

Создать ключи AES

Ключи созданы, передаем собеседнику ключи AES, публичный RSA и один приватный, а второй остается у нас.

Выберите длину ключей RSA

ключи:

1024 бит, устаревший стандарт, слабая защита, подходит только для ограниченных сценариев

Создать ключи RSA

Публичный ключ RSA

168963871319042360609421122453184156169593805894149719935954355195787212137046147560754857514291347443190650786693896991482066290944975146664196253986707795913054887930063928699339409700652769543620630406941510603802621610680530516918514494559585471594982208134959712137420747218817717329098815512522433734172518984731278233587960428005149963641237079377194543591581298084654776549830431733600702271986131340259025800614087261037460745199603956410743617540470182473372939863401496216490049394

Приватный ключ 1 RSA

3163217069810457187140157175432921840522886588321019151195973615654280820093063470993272291362649694599649685347276530778411943117464412640126125461332408947912946455052932566667731622036446023303299371875167540496003436124246630897192279162564733607974255153509168080667037402547579720239283362475835

Приватный ключ 2 RSA

10301039300479026886954317935934743129815669466616847735893036296555484243467981247042028191994533247098710056469666057784440790676918646974285917563197166140881189641582257948339666893620443050860284805829858835468143678356751406480916412546921975918669851352893735256774115508782459307720150377085312782863364079529346212026289834990485078904583792074468151758755943049365222739548624890988866551544369986611058481208605809809365206900257593498507352471686696282727736068294885985285799867

Создать ключи AES

Теперь создадим ключ AES, нажатием кнопки создать ключ AES

Создать ключи AES

ключ AES

DB-3E-0C-B3-AD-83-0B-74-B1-D8-3A-92-E0-46-04-60-91-B3-C8-28-B4-D9-89-D4-C8-2E-9B-2C-AC-47-A1-88

ключ IV

E6-36-01-7A-0A-A2-C7-A3-C9-2E-73-D1-3C-F3-42-2B

После того как вы передали собеседнику ключи(второй приватный ключ RSA и публичный, ключи AES) и перенесли ключи в созданный профиль зашифруем сообщение.

Вводим текст, который хотите зашифровать, нажимаем кнопку зашифровать и нас перекидывает на страницу где нам предлагает сохранить зашифрованное сообщение в текстовый файл.

привет мир !

168963871319042360609421122453184156169593805894149719935954355195787212137046147560754857514291347443190650786693896991482066290944975146664196253986707795913054887930063928699339409700652769543620630406941510603802621610680530516918514494559585471594982208134959712137420747218817717329098815512522433734172518984731278233587960428005149963641237079377194543591581298084654776549830431733600702271986131340259025800614087261037460745199603956410743617540470182473372939863401496216490049394263233476019804469863129872137936788976904548662023616263705264751212746695924420329576639375268098215324653603579554630441

3163217069810457187140157175432921840522886588321019151195973615654280820093063470993272291362649694599649685347276530778411943117464412640126125461332408947912946455052932566667731622036446023303299371875167540496003436124246630897192279162564733607974255153509168080667037402547579720239283362475835

DB-3E-0C-B3-AD-83-0B-74-B1-D8-3A-92-E0-46-04-60-91-B3-C8-28-B4-D9-89-D4-C8-2E-9B-2C-AC-47-A1-88

E6-36-01-7A-0A-A2-C7-A3-C9-2E-73-D1-3C-F3-42-2B

Зашифровать

FileSettings

Сохранить текст шифрования

Прочитать файл, для дальнейшей расшифровки

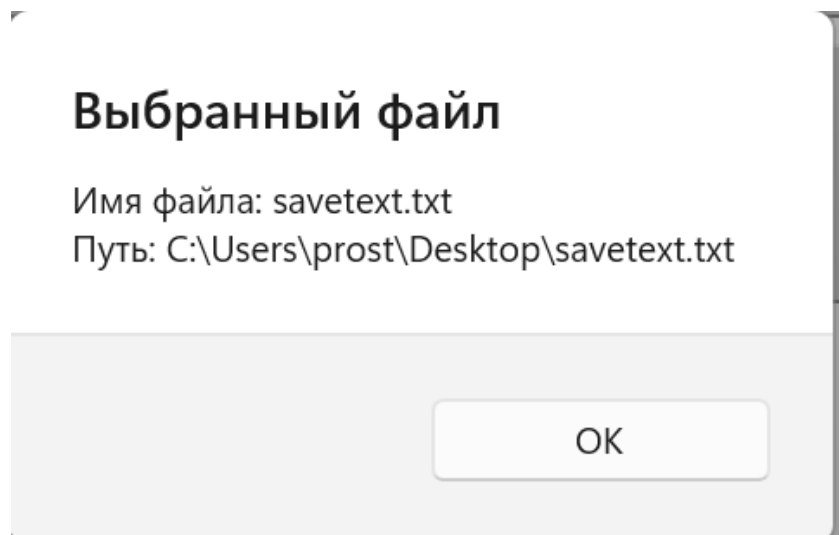
Выбрать файл (.txt) для сохранений

Очистить выбранный файл

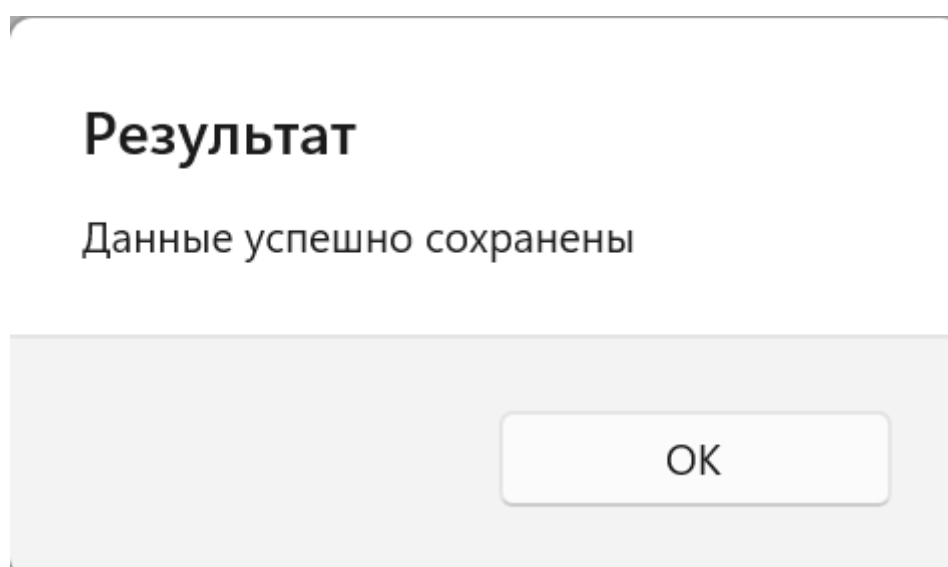
Путь к файлу

Имя файла

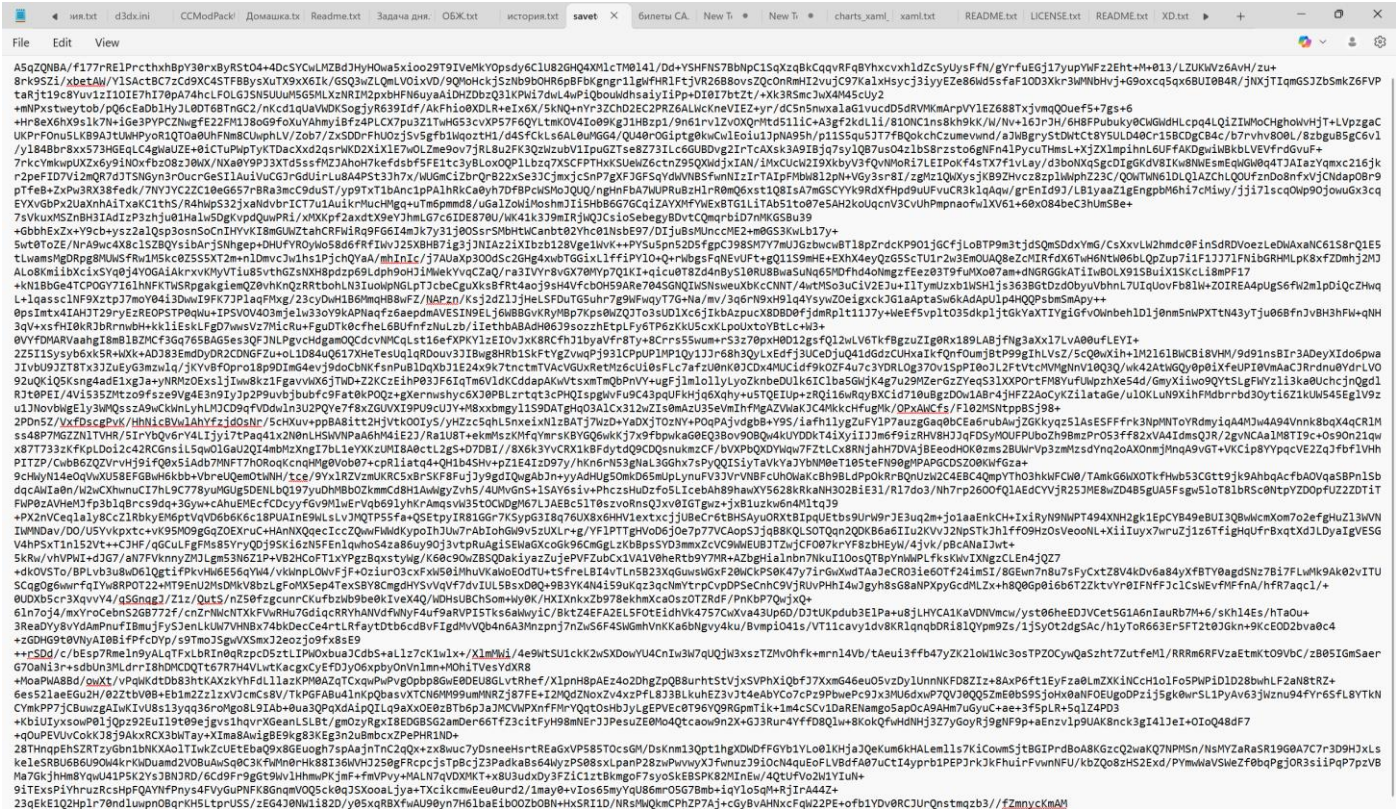
Выбираем файл



Когда выбрали файл нажимаем кнопку “сохранить текст шифрования” и видим результат



Если перейти в расположение текстового файла и открыть его, то вы увидите что текст зашифрован



Чтобы расшифровать текст мы заходим сначала на страницу “Настройки файла” и выбираем текстовый файл, с зашифрованным текстом.

Выбранный файл

Имя файла: savetext.txt

Путь: C:\Users\prost\Desktop\savetext.txt

OK

Нажимаем кнопку “Прочитать файл, для дальнейшей расшифровки”, и зашифрованный текст появляется в поле

4DcSYCwLMZBdJHyHOwa5xioo29T9IveMkYOpsdy6CIU82GHQ4XMicTM0i4I/Dd+YSHFN57BbNpC1SqXzqBkCqqvRFqBYhxcvxhldZcSyUysFfN/gYrfuEGj17yupYWFz2Eht+M+013/LZUKWVz6AvH/zu+
6JdC7i6L+AMM6A+D67+d4Vd6TDD+u+TV6+Y6H+6CQ3+7I+Q+U6+V6+Q6M+U+d6+N6+Q6+R6K+==1+M6+R6+V6+Q6+7Q+Q+R+U6+6CQ7+6H+6H+67+Q6+R6+64+Q6+V6+3

Нажимаем кнопку расшифровать и получаем расшифрованное сообщение

4DcSYCwLMZBdJHyHOwa5xioo29T9IveMkYOpsdy6CIU82GHQ4XMicTM0i4I/Dd+YSHFN57BbNpC1SqXzqBkCqqvRFqBYhxcvxhldZcSyUysFfN/gYrfuEGj17yupYWFz2Eht+M+013/LZUKWVz6AvH/zu+
6JdC7i6L+AMM6A+D67+d4Vd6TDD+u+TV6+Y6H+6CQ3+7I+Q+U6+V6+Q6M+U+d6+N6+Q6+R6K+==1+M6+R6+V6+Q6+7Q+Q+R+U6+6CQ7+6H+6H+67+Q6+R6+64+Q6+V6+3

Расшифровать

Привет, Мир!

Вывод

Разработанная система двухэтапного шифрования и дешифрования информации с использованием алгоритмов RSA и AES продемонстрировала высокую эффективность и надежность в защите

данных. Проект полностью выполнил поставленные задачи, реализовав комплексный подход к безопасности, сочетающий преимущества асимметричного и симметричного шифрования.

Основные результаты проекта:

1. Двухэтапная защита данных :
 - a. Текст сначала шифруется алгоритмом RSA с использованием публичного ключа, что обеспечивает безопасность на начальном этапе.
 - b. Результат RSA-шифрования подвергается дополнительному шифрованию AES, усиливая общую защиту и делая атаки на данные практически невозможными.
2. Управление ключами :
 - a. Реализована генерация пар ключей RSA (публичный и приватный) и ключа AES.
 - b. Создание профилей для хранения ключей упростило работу с разными пользователями или собеседниками, минимизировав риски ошибок при их использовании.
3. Дешифрование в обратном порядке :
 - a. Система успешно восстанавливает исходный текст через двухэтапный процесс: сначала дешифрование AES, затем RSA с использованием приватного ключа.
4. Практическая применимость :
 - a. Возможность сохранять зашифрованные данные в текстовые файлы обеспечила удобство их передачи и хранения, что актуально для реальных сценариев обмена конфиденциальной информацией.

Полезность и значимость проекта:

Разработанная система показала себя как надежный инструмент для защиты данных, сочетающий безопасность асимметричного шифрования (RSA) и эффективность симметричного (AES).

Комбинация алгоритмов позволила минимизировать уязвимости, характерные для каждого из них в отдельности. Управление ключами через профили и сохранение данных в файлах делает систему удобной для пользователей, что расширяет ее применение в областях, где требуется конфиденциальность, например, в корпоративной коммуникации или личной переписке.

Перспективы развития:

- Улучшение производительности за счет оптимизации алгоритмов.
- Добавление функции проверки целостности данных (например, с помощью хеш-функций).
- Реализация графического интерфейса для повышения удобства использования.

Проект подтвердил, что сочетание RSA и AES является оптимальным решением для обеспечения многоуровневой защиты информации в современных условиях, где безопасность данных остается ключевым приоритетом.

Github для скачивания приложения: https://github.com/AmigoFox/Encrypt_maui