Федеральное государственное бюджетное образовательное учреждение
высшего образования «Петрозаводский государственный университет»
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчёт по лабораторной работе №3
по дисциплине: «Криптографические средства»

Выполнил студент группы 22307:
Гордеев Никита Владиславович

Проверил преподаватель:
Кафедры прикладной математики и кибернетики
Института математики и информационных технологий
Ларионов Дмитрий Дмитриевич

Петрозаводск
2024

Оглавление

1 Формулировка задания

При помощи функций криптографической библиотеки .NET реализуйте гибридную криптосистему, включающую:

1. генерацию ключевой пары RSA;
2. шифрование и расшифрование документа симметричным криптоалгоритмом;

3. шифрование и расшифрование сеансового ключа симметричного алгоритма при помощи ключей RSA;
4. формирование и проверку цифровой подписи документа.

Полученный шифротекст, открытые ключи должны сохраняться и передаваться через файлы.

2 Описание метода решения

Программа представляет собой шифрование и расшифровку данных с использованием комбинации симметричного и асимметричного шифрования. Каждый этап подробно:

Шифрование (EncryptFile)

1. **Генерация и сохранение пары ключей RSA:**
    - Метод **GenerateAndSaveRSAKeys** создает пару ключей RSA.
    - Закрытый ключ сохраняется в файл **privateKey.pem**.
    - Открытый ключ сохраняется в файл **publicKey.pem**.
2. **Чтение текста из файла:**
    - Текст читается из файла **secretMessage.txt** с помощью **File.ReadAllText**.
3. **Шифрование данных с использованием AES:**
    - Метод **EncryptWithAES** создает симметричный ключ и IV (вектор инициализации).
    - Данные шифруются с использованием AES и возвращаются в виде зашифрованных данных вместе с ключом и IV.
4. **Шифрование симметричного ключа и IV с использованием RSA:**
    - Метод **EncryptDataWithRSA** шифрует симметричный ключ и IV с использованием открытого ключа RSA.
    - Открытый ключ читается из файла **publicKey.pem**.
5. **Генерация цифровой подписи:**
    - Метод **GenerateSignature** создает цифровую подпись для зашифрованных данных с использованием закрытого ключа RSA.
    - Закрытый ключ читается из файла **privateKey.pem**.
6. **Запись зашифрованных данных и ключей в файл:**
    - Метод **WriteEncryptedDataToFile** записывает зашифрованные симметричный ключ, IV, данные и цифровую подпись в файл **encryptedData.txt**.

Расшифровка (DecryptFile)

1. **Чтение зашифрованных данных из файла:**
    - Метод **ReadEncryptedFile** читает зашифрованные симметричный ключ, IV, данные и цифровую подпись из файла **encryptedData.txt**.
2. **Проверка цифровой подписи:**

- Метод **VerifySignature** проверяет цифровую подпись зашифрованных данных с использованием открытого ключа RSA.
- Открытый ключ читается из файла **publicKey.pem**.
- Если подпись не совпадает, программа завершает выполнение с сообщением об ошибке.

3. **Расшифровка симметричного ключа и IV:**
   - Метод **DecryptData** расшифровывает симметричный ключ и IV с использованием закрытого ключа RSA.
   - Закрытый ключ читается из файла **privateKey.pem**.
4. **Расшифровка данных с использованием AES:**
   - Метод **DecryptAesData** расшифровывает данные с использованием симметричного ключа и IV, полученных из предыдущего шага.
5. **Сохранение расшифрованного сообщения в файл:**
   - Метод **SaveDecryptedFile** сохраняет расшифрованное сообщение в файл **decryptedData.txt**.

Этот процесс обеспечивает безопасность передачи данных, используя сочетание симметричного шифрования (AES) для данных и асимметричного шифрования (RSA) для ключей и цифровой подписи для проверки целостности данных.

3 Примеры кода программ:

3.1 Создан класс Program:

3.1.1 Main - основной метод

```
static void Main()
{
    try
    {
        // Инициализация объектов для шифрования и дешифрования
        EncryptFile encryptFile = new EncryptFile();
        DecryptFile decryptFile = new DecryptFile();

        // Шифрование файла
        Console.WriteLine("Пользователь 1 – отправляет файл");
        encryptFile.Encrypt();

        // Дешифрование файла
        Console.WriteLine("Пользователь – 2 получает файл");
        decryptFile.Decrypt();
    }
    catch (Exception ex)
    {
        // Обработка исключений
        Console.WriteLine("Произошла ошибка: " + ex.Message);
    }
}
```

3.2 Создан класс EncryptFile

3.2.1 Encrypt - основной метод, реализующий полный процесс шифрования данных

```
public void Encrypt()
{
    Console.WriteLine($"Начинается шифрование");
    try
    {
        // Генерация и сохранение пары ключей RSA
        GenerateAndSaveRSAKeys();

        // Чтение текста из файла
        string text = File.ReadAllText(TextFilePath);

        // Генерация симметричного ключа и IV для AES и шифрование данных
        var (key, iv, encryptedData) = EncryptWithAES(text);

        // Шифрование симметричного ключа и IV с использованием открытого ключа RSA
        byte[] encryptedKey = EncryptDataWithRSA(key, PublicKeyFilePath);
        byte[] encryptedIV = EncryptDataWithRSA(iv, PublicKeyFilePath);

        // Генерация электронной подписи данных с использованием закрытого ключа RSA
        byte[] signature = GenerateSignature(encryptedData);

        // Запись зашифрованных данных и ключей в файл
        WriteEncryptedDataToFile(encryptedKey, encryptedIV, encryptedData,
signature);
        Console.WriteLine($"Файл зашифрован, данные сохранены в файле
{EncryptedFilePath}");
    }
    catch (Exception ex)
    {
        // Обработка ошибок и вывод сообщения об ошибке
        Console.WriteLine($"Ошибка при шифровании: {ex.Message}");
    }
}
```

3.2.2 GenerateAndSaveRSAKeys - метод для генерации и сохранения пары ключей RSA (открытый и закрытый ключи).

```
/// <summary>
/// Генерирует и сохраняет пару ключей RSA в файлы.
/// </summary>
private void GenerateAndSaveRSAKeys()
{
    using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
    {
        // Сохранение закрытого ключа
        string privateKey = rsa.ToXmlString(true);
        File.WriteAllText(PrivateKeyFilePath, privateKey);

        // Сохранение открытого ключа
        string publicKey = rsa.ToXmlString(false);
        File.WriteAllText(PublicKeyFilePath, publicKey);
    }
}
```

3.2.3 EncryptWithAES - метод для шифрования текста с использованием алгоритма AES, возвращающий симметричный ключ, IV и зашифрованные данные.

```
/// <summary>
/// Шифрует текст с использованием AES
/// Возвращает ключ, IV и зашифрованные данные.
///     IV (Initialization Vector, вектор инициализации) – это дополнительный ввод,
используемый с секретным ключом для повышения безопасности шифрования.
```

```
///     В алгоритме AES IV гарантирует, что одинаковые данные, зашифрованные с одним
и тем же ключом, будут иметь разные зашифрованные выходные данные.
/// </summary>
private (byte[] key, byte[] iv, byte[] encryptedData) EncryptWithAES(string text)
{
    using (Aes aes = Aes.Create())
    {
        byte[] key = aes.Key; // Генерация симметричного ключа
        byte[] iv = aes.IV; // Генерация вектора инициализации (IV)
        byte[] encryptedData;

        using (MemoryStream ms = new MemoryStream())
        using (CryptoStream cs = new CryptoStream(ms, aes.CreateEncryptor(),
CryptoStreamMode.Write))
        using (StreamWriter sw = new StreamWriter(cs))
        {
            // Запись текста в поток, зашифрованный с помощью AES
            sw.Write(text);
            sw.Flush();
            cs.FlushFinalBlock();
            encryptedData = ms.ToArray(); // Получение зашифрованных данных из
памяти
        }

        return (key, iv, encryptedData);
    }
}
```

3.2.4 EncryptDataWithRSA - метод для шифрования данных с использованием открытого ключа RSA.

```
/// <summary>
/// Шифрует данные с использованием открытого ключа RSA.
/// </summary>
private byte[] EncryptDataWithRSA(byte[] data, string publicKeyFilePath)
{
    string publicKey = File.ReadAllText(publicKeyFilePath);
    using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
    {
        rsa.FromXmlString(publicKey); // Импорт открытого ключа из строки XML
        return rsa.Encrypt(data, RSAEncryptionPadding.Pkcs1); // Шифрование данных с
использованием RSA
    }
}
```

3.2.5 GenerateSignature - метод для генерации цифровой подписи данных с использованием закрытого ключа RSA.

```
/// <summary>
/// Генерирует цифровую подпись данных с использованием закрытого ключа RSA.
/// </summary>
private byte[] GenerateSignature(byte[] data)
{
    string privateKey = File.ReadAllText(PrivateKeyFilePath);
    using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
    {
        rsa.FromXmlString(privateKey); // Импорт закрытого ключа из строки XML
        return rsa.SignData(data, HashAlgorithmName.SHA256,
RSASignaturePadding.Pkcs1); // Создание подписи данных
    }
}
```

3.2.6 WriteEncryptedDataToFile - метод для записи зашифрованных данных, ключей и подписи в файл.

```
/// <summary>
/// Записывает зашифрованные данные, ключи и подпись в файл.
```

```
/// </summary>
private void WriteEncryptedDataToFile(byte[] encryptedKey, byte[] encryptedIV,
byte[] encryptedData, byte[] signature)
{
    using (BinaryWriter writer = new BinaryWriter(File.Open(EncryptedFilePath,
FileMode.Create)))
    {
        // Запись зашифрованного ключа
        writer.Write(encryptedKey.Length);
        writer.Write(encryptedKey);

        // Запись зашифрованного IV
        writer.Write(encryptedIV.Length);
        writer.Write(encryptedIV);

        // Запись зашифрованных данных
        writer.Write(encryptedData.Length);
        writer.Write(encryptedData);

        // Запись электронной подписи
        writer.Write(signature.Length);
        writer.Write(signature);
    }
}
```

3.3 Создан класс DecryptFile

3.3.1 Decrypt - основной метод, реализующий полный процесс расшифровки данных:

```
public void Decrypt()
{
    Console.WriteLine("Начинается расшифровка");
    try
    {
        // Чтение зашифрованных данных из файла
        var (encryptedKey, encryptedIV, encryptedData, signature) =
ReadEncryptedFile(FilePath);

        // Проверка цифровой подписи
        if (!VerifySignature(encryptedData, signature, PublicKeyFilePath))
        {
            Console.WriteLine("Подпись не совпадает!");
            return;
        }

        // Расшифровка симметричного ключа и IV с помощью закрытого ключа RSA
        byte[] key = DecryptData(encryptedKey, PrivateKeyFilePath);
        byte[] iv = DecryptData(encryptedIV, PrivateKeyFilePath);

        // Расшифровка данных симметричным алгоритмом AES
        string decryptedText = DecryptAesData(encryptedData, key, iv);

        // Сохранение расшифрованного сообщения в файл
        SaveDecryptedFile(decryptedText, DecryptedFilePath);

        // Вывод расшифрованного сообщения
        Console.WriteLine("Расшифрованное сообщение сохранено в: " +
DecryptedFilePath);
    } catch (Exception ex)
    {
        // Обработка ошибок
        Console.WriteLine($"Ошибка при расшифровке: {ex.Message}");
    }
}
```

3.3.2 ReadEncryptedFile - метод для чтения зашифрованного файла и извлечения из него зашифрованного ключа, IV, данных и цифровой подписи.

```
/// <summary>
/// Читает зашифрованный файл и извлекает из него данные.
/// </summary>
private (byte[] encryptedKey, byte[] encryptedIV, byte[] encryptedData, byte[]
signature) ReadEncryptedFile(string filePath)
{
    try
    {
        using (BinaryReader reader = new BinaryReader(File.Open(filePath,
FileMode.Open)))
        {
            // Чтение зашифрованного ключа
            byte[] encryptedKey = reader.ReadBytes(reader.ReadInt32());
            // Чтение зашифрованного IV
            byte[] encryptedIV = reader.ReadBytes(reader.ReadInt32());
            // Чтение зашифрованных данных
            byte[] encryptedData = reader.ReadBytes(reader.ReadInt32());
            // Чтение цифровой подписи
            byte[] signature = reader.ReadBytes(reader.ReadInt32());

            return (encryptedKey, encryptedIV, encryptedData, signature);
        }
    } catch (IOException ex)
    {
        // Обработка ошибок при чтении файла
        throw new Exception("Ошибка при чтении зашифрованного файла", ex);
    }
}
```

3.3.3 VerifySignature - метод для проверки цифровой подписи данных с использованием открытого ключа RSA.

```
/// <summary>
/// Проверяет цифровую подпись данных.
/// </summary>
private bool VerifySignature(byte[] data, byte[] signature, string
publicKeyFilePath)
{
    try
    {
        using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
        {
            // Чтение открытого ключа из файла
            string publicKey = File.ReadAllText(publicKeyFilePath);
            rsa.FromXmlString(publicKey);

            // Проверка цифровой подписи
            return rsa.VerifyData(data, SHA256.Create(), signature);
        }
    } catch (CryptographicException ex)
    {
        // Обработка ошибок при проверке подписи
        throw new Exception("Ошибка при проверке цифровой подписи", ex);
    }
}
```

3.3.4 DecryptData - метод для расшифровки данных с использованием закрытого ключа RSA.

```
/// <summary>
/// Расшифровывает данные с использованием закрытого ключа RSA.
/// </summary>
private byte[] DecryptData(byte[] data, string privateKeyFilePath)
{
    try
```

```
    {
        // Чтение закрытого ключа из файла
        string privateKey = File.ReadAllText(privateKeyFilePath);
        using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())
        {
            rsa.FromXmlString(privateKey);

            // Расшифровка данных с использованием RSA
            return rsa.Decrypt(data, RSAEncryptionPadding.Pkcs1);
        }
    } catch (CryptographicException ex)
    {
        // Обработка ошибок при расшифровке данных
        throw new Exception("Ошибка при расшифровке данных с использованием
закрытого ключа", ex);
    }
}
```

3.3.5 DecryptAesData - метод для расшифровки зашифрованных данных с использованием алгоритма AES.

```
/// <summary>
/// Расшифровывает зашифрованные данные с использованием алгоритма AES.
/// </summary>
private string DecryptAesData(byte[] encryptedData, byte[] key, byte[] iv)
{
    try
    {
        using (Aes aes = Aes.Create())
        {
            aes.Key = key;
            aes.IV = iv;

            using (MemoryStream ms = new MemoryStream(encryptedData))
            using (CryptoStream cs = new CryptoStream(ms, aes.CreateDecryptor(),
CryptoStreamMode.Read))
            using (StreamReader sr = new StreamReader(cs))
            {
                // Чтение и расшифровка данных
                return sr.ReadToEnd();
            }
        }
    } catch (CryptographicException ex)
    {
        // Обработка ошибок при расшифровке данных AES
        throw new Exception("Ошибка при расшифровке данных с использованием AES",
ex);
    }
}
```

3.3.6 SaveDecryptedFile - метод для сохранения расшифрованного сообщения в файл.

```
/// <summary>
/// Сохраняет расшифрованное сообщение в файл.
/// </summary>
private void SaveDecryptedFile(string decryptedText, string filePath)
{
    try
    {
        File.WriteAllText(filePath, decryptedText, Encoding.UTF8);
    } catch (IOException ex)
    {
        // Обработка ошибок при сохранении файла
        throw new Exception("Ошибка при сохранении расшифрованного файла", ex);
    }
}
```

3.4 Реализован класс Main для запуска программы:

```
public void Decrypt()
{
    Console.WriteLine("Начинается расшифровка");
    try
    {
        // Чтение зашифрованных данных из файла
        var (encryptedKey, encryptedIV, encryptedData, signature) =
ReadEncryptedFile(FilePath);

        // Проверка цифровой подписи
        if (!VerifySignature(encryptedData, signature, PublicKeyFilePath))
        {
            Console.WriteLine("Подпись не совпадает!");
            return;
        }

        // Расшифровка симметричного ключа и IV с помощью закрытого ключа RSA
        byte[] key = DecryptData(encryptedKey, PrivateKeyFilePath);
        byte[] iv = DecryptData(encryptedIV, PrivateKeyFilePath);

        // Расшифровка данных симметричным алгоритмом AES
        string decryptedText = DecryptAesData(encryptedData, key, iv);

        // Сохранение расшифрованного сообщения в файл
        SaveDecryptedFile(decryptedText, DecryptedFilePath);

        // Вывод расшифрованного сообщения
        Console.WriteLine("Расшифрованное сообщение сохранено в: " +
DecryptedFilePath);
    } catch (Exception ex)
    {
        // Обработка ошибок
        Console.WriteLine($"Ошибка при расшифровке: {ex.Message}");
    }
}
```

4 Тестовые данные

Я проводил тесты с пустым сообщением, с сообщением только из цифр, только из кириллических символов, только из латинских символов, с переносами и без. Программа работает без ошибок благодаря проверкам try catch.