Лабораторная работа

Тема: Реализация алгоритма передачи информации с использованием современных симметричных криптосистем

Теоретическая часть

Симметричные криптосистемы (также симметричное шифрование, симметричные шифры) (англ. symmetric-key algorithm) — способ шифрования, в котором для шифрования и расшифрования применяется один и тот же криптографический ключ. До изобретения схемы асимметричного шифрования единственным существовавшим способом являлось симметричное шифрование. Ключ алгоритма должен сохраняться в тайне обеими сторонами, должны осуществляться меры по защите доступа к каналу, на всем пути следования криптограммы, или сторонами взаимодействия посредством криптообъектов, сообщений, если данный канал взаимодействия под грифом «Не для использования третьими лицами». Алгоритм шифрования выбирается сторонами до начала обмена сообщениями.

AES (англ. Advanced Encryption Standard; также Rijndael, [rɛinda:l] — рейндал) — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES.

Практическая часть

Реализация алгоритма шифрования RSA

Для использования алгоритма AES при разработке приложений на языке программирования C# необходимо использовать пространства имен System.Security.Cryptography.

```
using System.Security.Cryptography;
```

Из данного пространства имен потребуются классы UnicodeEncoding и Aes.

UnicodeEncoding - Представляет кодировку символов Юникода в формате UTF-16. Кодирование - это процесс преобразования набора символов Юникода в последовательность байтов. Декодирование — это процесс преобразования последовательности закодированных байтов в набор символов Юникода. Ссылка на документацию от Microsoft: https://learn.microsoft.com/ru-ru/dotnet/api/system.text.unicodeencoding?view=net-6.0

Класс Aes Представляет абстрактный базовый класс, от которого должны наследоваться все реализации стандарта AES. Для создания экземпляра класса Aes необходимо выполнить следующий код:

```
Aes myAes = Aes.Create();  // Экземпляр класса Aes
                           // отвечает за шифрование и ключи
              // Здесь также генерируется ключ: сам ключ и вектор инициализации (IV)
```

Реализация алгоритма шифрования, предложенная компанией Microsoft:

```
// Алгоритм шифрования от Microsoft
        static byte[] EncryptStringToBytes_Aes(string plainText, byte[] Key, byte[] IV)
        {
            // Пооверка входящих переменных на валидность
            if (plainText == null || plainText.Length <= 0)
                throw new ArgumentNullException("plainText");
            if (Key == null || Key.Length <= 0)
                throw new ArgumentNullException("Key");
            if (IV == null || IV.Length <= 0)
                throw new ArgumentNullException("IV");
            byte[] encrypted;

            // Создается экземпляр класса Aes для использования ключа и IV
            using (Aes aesAlg = Aes.Create())
            {
                aesAlg.Key = Key; //
                aesAlg.IV = IV;

                // Работа производится с использованием потоков
                // Create an encryptor to perform the stream transform.
                ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key,
aesAlg.IV);

                // Create the streams used for encryption.
                using (MemoryStream msEncrypt = new MemoryStream())
                {
                    using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
encryptor, CryptoStreamMode.Write))
                    {
                        using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                        {
                            //Write all data to the stream.
                            swEncrypt.Write(plainText);
                        }
                        encrypted = msEncrypt.ToArray();
                    }
                }
            }

            // Возвращаем зашифрованные данные.
            return encrypted;
      }
```

Алгоритм дешифрования, предложенный компанией Microsoft:

```
// Алгоритм расшифровки от Microsoft
```

```
        static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key, byte[]
IV)
        {
            // Пооверка входящих переменных на валидность
            if (cipherText == null || cipherText.Length <= 0)
                throw new ArgumentNullException("cipherText");
            if (Key == null || Key.Length <= 0)
                throw new ArgumentNullException("Key");
            if (IV == null || IV.Length <= 0)
                throw new ArgumentNullException("IV");

            // Строка для формирования расшифрованного текста
            string plaintext = null;

            // Создается экземпляр класса Aes для использования ключа и IV
            using (Aes aesAlg = Aes.Create())
            {
                aesAlg.Key = Key;
                aesAlg.IV = IV;

                // Работа производится с использованием потоков
                // Create a decryptor to perform the stream transform.
                ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key,
aesAlg.IV);

                // Create the streams used for decryption.
                using (MemoryStream msDecrypt = new MemoryStream(cipherText))
                {
                    using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
                    {
                        using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                        {

                            // Read the decrypted bytes from the decrypting stream
                            // and place them in a string.
                            plaintext = srDecrypt.ReadToEnd();
                        }
                    }
                }
            }
            // Возвращаем расшифрованные данные.
            return plaintext;
        }
```

Реализуем простое приложение, для проверки работы алгоритма AES. Для этого реализуем форму, представленную на рисунке 1.

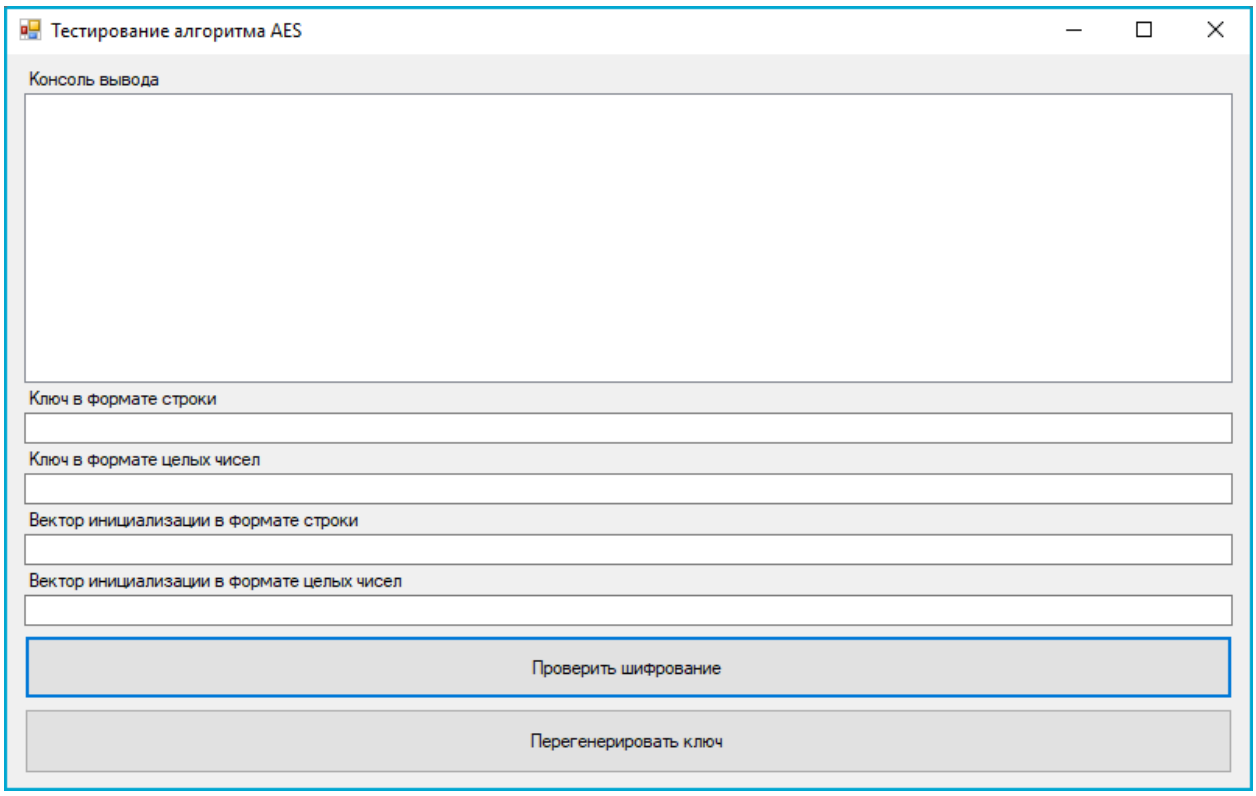

Рис. 1. Форма тестового приложения

Для проверки корректности работы программно реализуем шифрование и вывод всех промежуточных значений с использованием следующего кода:

```
// Строка для шифрования
            string original = "Привет от Астафьева Александра!";

            // Создаём новый экземпляк класса Aes

            //myAes = Aes.Create();

            // Выводим ключ в компоненты textBox
            textBox1.Text = Encoding.Unicode.GetString(myAes.Key);
            string keyInInt = "";
            for (int i = 0; i < myAes.Key.Length; i++)
            {
                keyInInt += myAes.Key[i].ToString()+", ";
            }
            textBox2.Text = keyInInt.Substring(0, keyInInt.Length - 2); // Убираем лишнюю
запятую, а то меня раздражает

            // Выводим вектор инициализации в компоненты textBox
            textBox4.Text = Encoding.Unicode.GetString(myAes.IV);
            string IVInInt = "";
            for (int i = 0; i < myAes.IV.Length; i++)
            {
                IVInInt += myAes.IV[i].ToString() + ", ";
            }
            textBox3.Text = IVInInt.Substring(0, IVInInt.Length - 2); // Убираем лишнюю
запятую, а то меня раздражает


            // Шифрование данных в массив байт.
            byte[] encrypted = EncryptStringToBytes_Aes(original, myAes.Key, myAes.IV);
```

```
            // Расшифрование массива байт в строку.
            string roundtrip = DecryptStringFromBytes_Aes(encrypted, myAes.Key,
myAes.IV);

            // Вывод данных.
            listBox1.Items.Add("Исходный текст:       " + original);
            listBox1.Items.Add("Зашифрованный текст:  " +
Encoding.Unicode.GetString(encrypted, 0, encrypted.Length));
            listBox1.Items.Add("Расшифрованный текст: " + roundtrip);
```

При выполнении приведенного кода должен получиться результат, приведенный на рисунке 2.

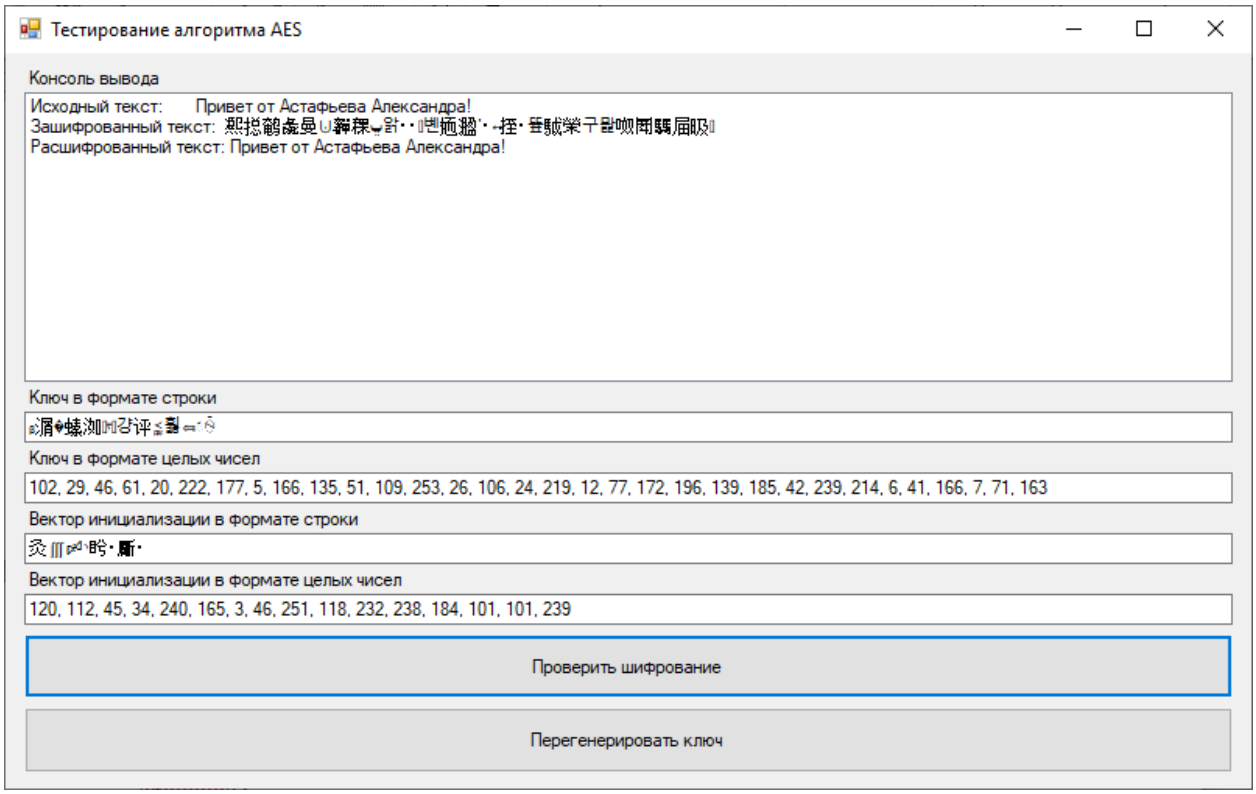

Рис. 2. Результат работы тестового приложения

Генерация ключа шифрования производится при создании экземпляра класса AES. Однако, во время работы можно вручную перегенерировать ключи шифрования. Для этого используется следующий код:

```
            myAes.GenerateIV();
            myAes.GenerateKey();
            // Выводим ключ в компоненты textBox
            textBox1.Text = Encoding.Unicode.GetString(myAes.Key);
            string keyInInt = "";
            for (int i = 0; i < myAes.Key.Length; i++)
            {
                keyInInt += myAes.Key[i].ToString() + ", ";
            }
            textBox2.Text = keyInInt.Substring(0, keyInInt.Length - 2); // Убираем лишнюю
запятую, а то меня раздражает

            // Выводим вектор инициализации в компоненты textBox
```

```
            textBox4.Text = Encoding.Unicode.GetString(myAes.IV);
            string IVInInt = "";
            for (int i = 0; i < myAes.IV.Length; i++)
            {
                IVInInt += myAes.IV[i].ToString() + ", ";
            }
                textBox3.Text = IVInInt.Substring(0, IVInInt.Length - 2); // Убираем
лишнюю запятую, а то меня раздражает
```

Общую схему работы можно изобразить в виде рисунка 3.

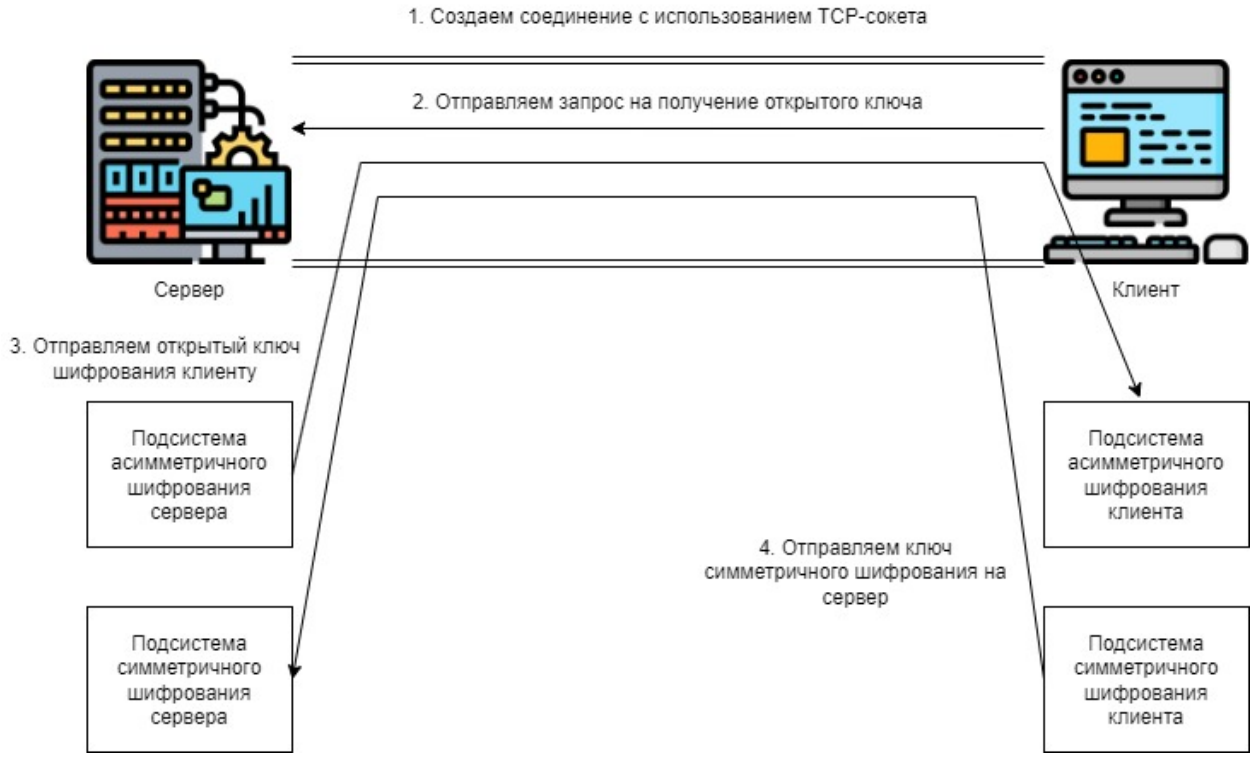

Рис. 3. Общая схема работы клиент-серверного приложения с симметричным шифрованием

Типовой порядок работы

В рамках лабораторной работы предлагается следующий порядок работы:

1. После соединения клиента и сервера посредствам сокетов (программа разработанная ранее) клиент отправляет запрос на открытый ключ сервера.
2. Сервер, получив запрос (например, сообщение "OpenKeyRequest"), выгружает открытый ключ и отправляет его клиенту.
3. Клиент, получая очередное сообщение, должен проверить является ли оно ключом. Об этом можно судить по наличию открывающего тега <RSAKeyValue>.
4. Клиент должен проверить целостность ключа. Это можно определить по закрывающему тегу </RSAKeyValue>.
5. После получения открытого ключа клиент должен его сохранить

и внести в параметры крипто провайдера, например, с использованием метода RSA.FromXmlString.

6. Реализовать генерацию ключа шифрования симметричного алгоритма AES.
7. Передать ключ шифрования от клиента к серверу, используя шифрование RSA.
8. Весь дальнейший трафик должен быть зашифрован с использованием симметричного шифрования AES.

Задание на лабораторную работу

Используя в качестве механизма сетевого взаимодействия приложение, реализованное в прошлой работе, произвести реализацию возможности передачи информации по открытому каналу связи с использованием криптографического алгоритма AES:

1. Реализовать работу алгоритма AES на клиенте и сервере.
2. Реализовать механизм передачи ключа симметричного шифрования с использованием алгоритма RSA по открытому каналу связи.
3. Весь дальнейший трафик должен быть зашифрован с использованием алгоритма AES.
4. Произвести захват сетевого трафика с использованием программы WireShark по локальному интерфейсу «Adapter for loopback traffic capture».
5. Создать дисплейный фильтр для фильтрации сообщений, передаваемых приложениями на базе сокетов.
6. Визуализировать содержание пакетов с открытым ключом и зашифрованной информацией.
7. Оформить отчёт по проделанной работе с фиксацией основных моментов работы.