DOI 10.37882/2223-2966.2025.05.25

ЗАШИФРОВАННАЯ ПЕРЕПИСКА ЧЕРЕЗ ПРОТОКОЛ SMTP РЕАЛИЗОВАННЫЙ НА ЯЗЫКЕ C#

PROBLEM-ORIENTED LANGUAGE FOR WORKING WITH TEXT AS A PYTHON MODULE

A. Mukhortov
S. Koryagin

Summary. Confidentiality within the company is one of the most important factors in its work. To ensure security, it is necessary to think over a system for protecting the data transmitted inside the information system, which can entail large costs. The need for confidential correspondence between subjects in the company will reduce the risk of data leakage and their further dissemination. Any company uses an electronic mail service to communicate between departments and with customers. In this article, we implement secure correspondence based on the SMTP data transfer protocol using a strong AES encryption algorithm. The advantage of this method of communication is that the library will need to be integrated into the end devices in the information system, and the data will be stored on the server in encrypted form, thereby there is no need to alter the existing information system.

Keywords: SMTP, mail client, encryption algorithms, AES, C#, data protection.

Мухортов Артём Александрович
Российский Технологический Университет МИРЭА
Artem.mukh@gmail.com
Корягин Сергей Викторович
кандидат технических наук,
МГУПИ «Московский государственный университет приборостроения и информатики»
dongenealog2003@mail.ru

Аннотация. Конфиденциальность внутри компании является одним из важнейших факторов в её работе. Для обеспечения безопасности необходимо продумать систему защиты данных, передаваемых внутри информационной системы, что может повлечь за собой большие издержки. Необходимость конфиденциальной переписки между субъектами в компании позволит уменьшить риск утечки данных и дальнейшее их распространение. В любой фирме используется электронный почтовый сервис для общения между отделами и с клиентами. В данной статье реализуем защищённую переписку на основе протокола передачи данных SMTP используя крепкий алгоритм шифрования AES. Преимуществом данного способа общения заключается в том, что библиотеку необходимо будет интегрировать на конечные устройства в информационной системе, а данные будут храниться на сервере в зашифрованном виде, тем самым отсутствует необходимость в переделке существующей информационной системе.

Ключевые слова: SMTP, почтовый клиент, алгоритмы шифрования, AES, C#, защита данных.

Введение

Simple Mail Transfer Protocol предназначен для отправки писем из одной системы в другую.

Это могут быть почтовые клиенты, такие как Outlook, почтовые серверы, такие как Microsoft Exchange, межсетевой экран и т.д.

Связь по умолчанию осуществляется в виде открытого текста. Но в настоящее время вы, скорее всего, увидите, как почтовые серверы переключаются с обычного текста на защищенный канал с помощью SSL / TLS.

Несмотря на это сами данные, которые передаются по протоколу не зашифрованы, соответственно к ним можно получить доступ. Изучив библиотеки научных статьей:

- sciencedirect.com
- onlinelibrary.wiley.com
- Научная библиотека eLibrary.ru

Статьей на подобную тематику не было обнаружено. Суть защищённой переписки по протоколу SMTP, заключается в том, что данные на сервере будут храниться в зашифрованном виде, и для того, чтобы дешифровать их необходимо иметь ключ шифрования, который будет иметься только на персональных компьютерах конечных пользователей. Таким образом сохраниться конфиденциальность. Для шифрования будем использовать стойкий криптографический алгоритм AES.

SMTP протокол

SMTP — требующий соединения текстовый протокол, по которому отправитель сообщения связывается с получателем посредством выдачи командных строк и получения необходимых данных через надёжный канал, в роли которого обычно выступает TCP-соединение. SMTP-сессия состоит из команд, посылаемых SMTP-клиентом, и соответствующих ответов SMTP-сервера. Когда сессия открыта, сервер и клиент обмениваются её параметрами. Сессия может включать ноль и более SMTP-операций (транзакций).

SMTP-операция состоит из трёх последовательностей команда/ответ (см. пример ниже). Описание последовательностей:

- **MAIL FROM** — устанавливает обратный адрес. Это адрес для возвращённых писем.[3]
- **RCPT TO** — устанавливает получателя данного сообщения. Эта команда может быть дана несколько раз, по одной на каждого получателя. Эти адреса также являются частью оболочки.[3]
- **DATA** — для отправки текста сообщения. Это само содержимое письма, в противоположность его оболочке. Он состоит из заголовка сообщения и тела сообщения, разделённых пустой строкой. DATA, по сути, является группой команд, а сервер отвечает дважды: первый раз на саму команду DATA, для уведомления о готовности принять текст; и второй раз после конца последовательности данных, чтобы принять или отклонить всё письмо.[3]

Преимуществом использования протокола SMTP, является его простота в запуске в информационной системе, проверенная временем работоспособность и популярность в использовании. Любая компания использует почтовый сервер для общения с конечным потребителем и коммуникации внутри фирмы. Это позволяет упростить внедрение библиотеки в информационную систему предприятий.

Шифрование AES

AES — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит). Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES.[1]

На данный момент AES является одним из самых криптостойких алгоритмов шифрования.

Прежде всего отметим, что алгоритм AES оперирует байтами, которые интерпретируются как элементы конечного поля F(28). В этом поле определены операции сложения и умножения двух элементов, результатом которых, в свою очередь, также является элемент этого поля. Рассмотрим каждую из операций:

- **Сложение** выполняется с помощью операции xor. Операция выполняется над двоичными числами поразрядно, то есть для двух байт P = { p7, p6, p5, p4, p3, p2, p1, p0 } и Q = { q7, q6, q5, q4, q3, q2, q1, q0 } результатом будет R = { r7, r6, r5 , r4, r3, r2, r1, r0 }, где ri = pi xor ri .
- **Умножение.** Для этой операции используется представление байта в виде полинома: p(x) = p7x7 + p6x6 + p5x5 + p4x4 + p3x3 + p2x2 + p1x + p0 , умножение в поле F(28) в таком представлении производится по модулю неприводимого в этом поле многочлена m(x) = x8 + x4 + x3 + x + 1. Таким образом, для того, чтобы получить результат умножения двух чисел в поле F(28), мы должны представить их в виде полиномов p(x) и q(x), затем взять остаток от деления на многочлен m(x) произведения p(x) и q(x), то есть r(x) = p(x)q(x) mod (m(x)), и получить коэффициенты полинома r(x), которые являются 8-битовым числом в поле F(28).

Теперь перейдем от математики к описанию самого алгоритма шифрования AES с размером ключа 128 бит.

Предварительно входные данные разбиваются на блоки по 16 байт, если полный размер не кратен 16 байтам, то данные дополняется до размера, кратного 16 байтам. Блоки представляются в виде матрицы 4x4 — state. Далее происходит процедура расширения ключа и к каждому блоку state применяются операции 2–4. Итак, алгоритм состоит из следующих шагов:

- Расширение ключа — KeyExpansion;
- Начальный раунд — сложение state с основным ключом;
- 9 раундов шифрования, каждый из которых состоит из преобразований

— SubBytes
— ShiftRows
— MixColumns
— AddRoundKey

- Финальный раунд, состоящий из преобразований:

— SubBytes
— ShiftRows
— AddRoundKey

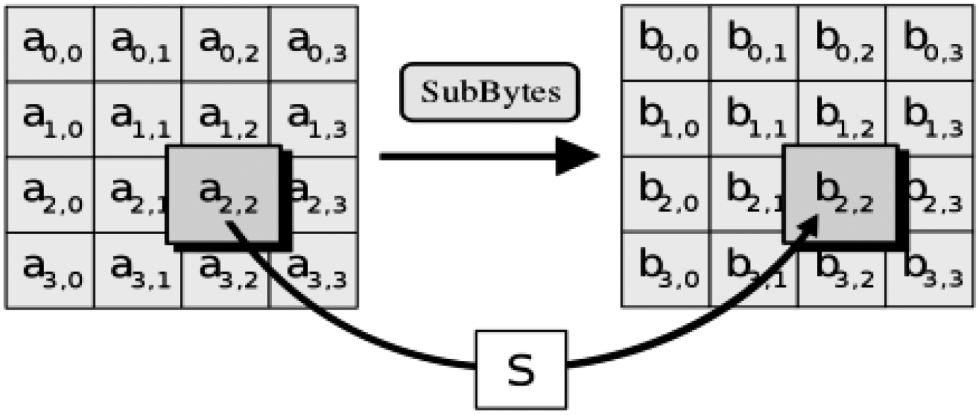
SubBytes — замена байтов state по таблице S-box. Каждый байт представляется в виде двух шестнадцатеричных чисел b = (x, y), где x определяется 4 старшими разрядами b, а y — 4 младшими. В таблице S-box размера 16x16 находятся значения для замены исходного байта: значение b' на пересечении строки x и столбца y S-box используется в качестве замены исходному байту b.[2]

- ShiftRows — циклический сдвиг строк state. Нулевая строка остается на месте, первая смещается влево на 1 байт, вторая на 2 байта и третья на 3 соответственно [2].

MixColumns — умножения каждого столбца state на фиксированную матрицу. Таким образом осуществляется линейное преобразование над столбцами state. Причем умножение и сложение производится по правилам, описанным выше.[2]

AddRoundKey — раундовый ключ поэлементно добавляется к state с помощью поразрядного XOR.[2]

KeyExpansion — процедура расширения основного ключа для создания раундовых ключей, которые затем используются в раундах шифрования. Расширенный



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Рис. 1. SubBytes [2]

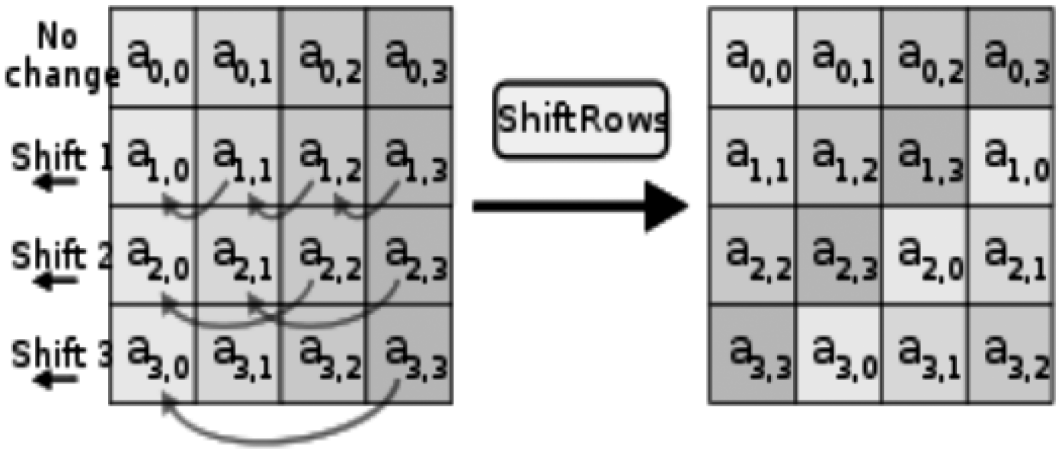


Рис. 2. ShiftRows [2]

ключ состоит из 44 четырехбайтовых слов (wi): 4 слова на основной ключ и по 4 слова на 10 раундовых ключей. Таким образом, полная длина расширенного ключа составляет 1408 бит. [2]

Так же стоит отметить другие алгоритмы шифрования.

- DES — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт. Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных и линейных преобразований. Из-за того, что данный алгоритм уже старый его защищённость не удовлетворяет существующим нормам.
- Triple DES (3DES) — симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на ос-

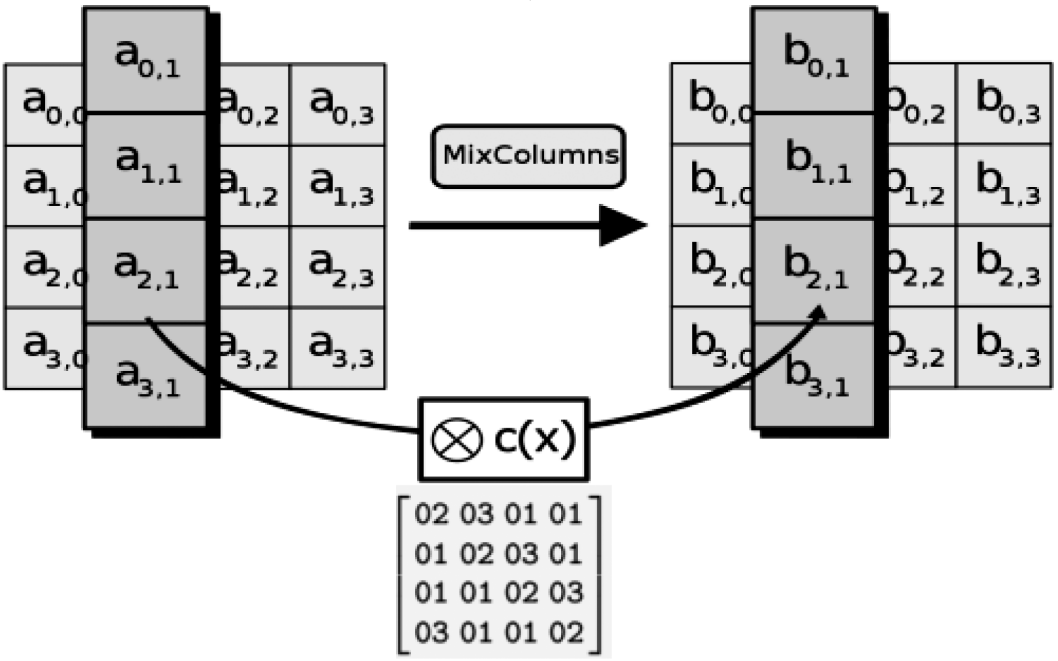


Рис. 3. MixColumns[2]

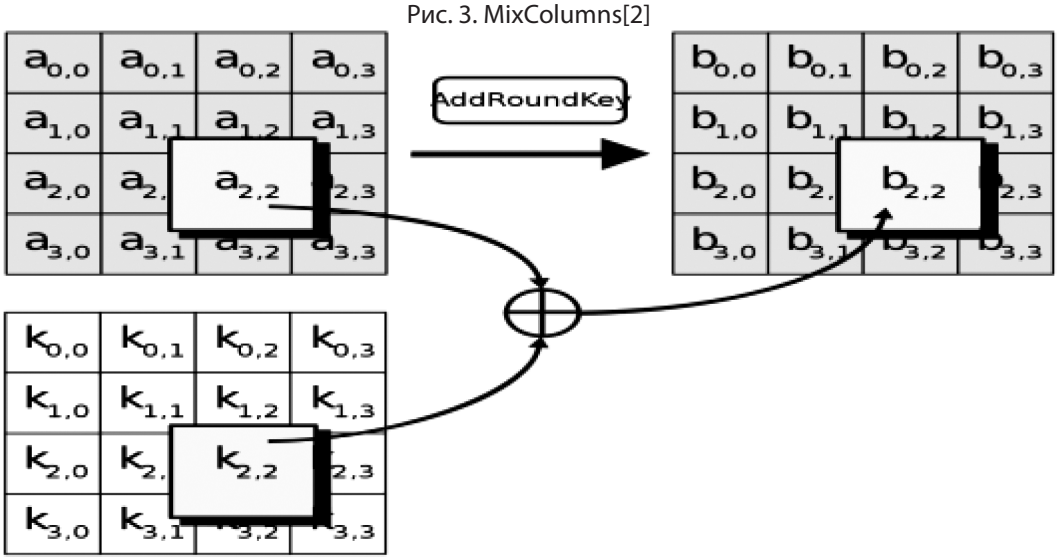


Рис. 4. AddRoundKey[2]

нове алгоритма DES с целью устранения главного недостатка последнего — малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше — время, требуемое для криптоанализа 3DES, может быть в миллиард раз больше, чем время, нужное для вскрытия DES. Данный алгоритм удовлетворяет существующим нормам защищённости, но из-за слишком долгой обработки не подходит для нашей задачи.

В среде разработке Microsoft .NET существует встроенная библиотека шифрования AES, которую мы будем использовать в процессе реализации библиотеки.

Практическая реализация алгоритма шифрования

Для реализации шифрования используем встроенную библиотеку в .NET 6 System.Security.Cryptography. На основе этого разработаем класс, который будет зашифровывать и дешифровывать наши данные — AESCrypto, который будет шифровать данные на базе 256-битного ключа.

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

class AESCrypto
{
  public string Message;
```

```
        public byte[] Cypher;
        private byte[] Key;

        public AESCrypto(string key, string message)//Конструктор для шифрования сообщения
        {
            Key = getBytesKey(key);
           //Обязательным условием является 32 разрядный ключ иначе шифрование не пройдёт а искомые значения раны NULL
            if (Key!=null)
            {
                Cypher = EncryptStringToBytes_Aes(message);
            }

        }
         public AESCrypto(string key, byte[] cypher)//Конструктор для дешифрования сообщения
        {


            Key = getBytesKey(key);
           //Обязательным условием является 32 разрядный ключ иначе дешифрование не пройдёт а искомые значения раны NULL
            if (Key!=null)
            {
                Message = DecryptStringFromBytes_Aes(cypher);
            }

        }

          private byte[] EncryptStringToBytes_Aes(string plainText)//Метод шифрующий текст по ключу
            {
                byte[] encrypted;
                byte[] IV;

                using (Aes aesAlg = Aes.Create())
                {
                    aesAlg.Key = Key;

                    aesAlg.GenerateIV();
                    IV = aesAlg.IV;

                    aesAlg.Mode = CipherMode.CBC;

                     var encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

                      using (var msEncrypt = new MemoryStream())
                    {
                using (var csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))

                        {
                                using (var swEncrypt = new StreamWriter(csEncrypt))
                                {

                                    swEncrypt.Write(plainText);
                                }
                                encrypted = msEncrypt.ToArray();
                            }
                        }
                    }

                var combinedIvCt = new byte[IV.Length + encrypted.Length];
                    Array.Copy(IV, 0, combinedIvCt, 0, IV.Length);
                    Array.Copy(encrypted, 0, combinedIvCt, IV.Length, encrypted.Length);

                    return combinedIvCt;

                }

                 private string DecryptStringFromBytes_Aes(byte[] cipherTextCombined) //Метод дешифрующий текст по ключу

                {

                    string plaintext = null;

                    using (Aes aesAlg = Aes.Create())
                    {
                        aesAlg.Key = Key;

                        byte[] IV = new byte[aesAlg.BlockSize/8];
                    byte[] cipherText = new byte[cipherTextCombined.Length — IV.Length];

                        Array.Copy(cipherTextCombined, IV, IV.Length);
                            Array.Copy(cipherTextCombined, IV.Length, cipherText, 0, cipherText.Length);

                        aesAlg.IV = IV;

                        aesAlg.Mode = CipherMode.CBC;

                                ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

                                    using (var msDecrypt = new MemoryStream(cipherText))
                        {
                    using (var csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
                            {
                                        using (var srDecrypt = new StreamReader(csDecrypt))
```

```
                {
                    plaintext = srDecrypt.ReadToEnd();
                }
            }
        }

    }

    return plaintext;

}
 private byte[] getBytesKey(string key) //Преобразование строки
{
    if(key.Length==32)
    {
        return Encoding.ASCII.GetBytes(key);
    }
    else
    {
        return null;
    }
}

}
```

В данной библиотеке присутствуют методы:

- getBytesKey() — преобразует ключ, полученный в String в массив Byte
- DecryptStringFromBytes_Aes() — метод дешифрующий текст по заданному ключу
- EncryptStringToBytes_Aes() — метод шифрующий текст по заданному ключу

В качестве сервера для отправки сообщений используем сервера Microsoft Outlook. Класс MessageSender создаёт образ письма и в качестве арументов принимает данные логина, пароля, порта, по которому он будет обращаться к серверу и адрес сервера. Для отправки Email воспользуемся встроенный в библиотеку Microsoft.NET System.Mail.

Для повышения защиты SMTP для отправки писем будет использовать SSL шифрование данных, для того чтобы повысить защищённость передачи.

```
using System.Net;
using System.Net.Mail;
using System.Security;
using mail;

namespace unit
{
   class MessageSender
  {
     MessageBuilder build;

     public MessageSender(MessageBuilder builder)
     {
       this.build = builder;
     }

        public void CreateMessage(string fromAddr, string toAddr, string message)
     {
       build.BuildAddress(fromAddr, toAddr);
       build.BuildTheme();
       build.BuildText(message);

     }

          public void SendMessage(string server,int port,string username, string pass)
     {
       SmtpClient smtp = new SmtpClient(server, port);
                  SecureString theSecureString = new NetworkCredential(«», pass).SecurePassword;
      smtp.Credentials = new NetworkCredential(username, theSecureString);
       smtp.EnableSsl = true;
       smtp.Send(build.GetResult());
     }
   }
}
```

Для дешифрования на конечном устройстве, необходимо использовать класс AESCrypto, но в качестве входных параметров в конструктор используем ключ и битовое представление зашифрованных данных. Для преобразования String байтового представления данных, разработаем метод stringToByteAES() в библиотеке AESCrypto(), который возвращает массив байтов для дальнейшего дешифрования.

```
static byte[] stringToByteAES(string mess)
  {
    int l=0;
    string[] array_bytes = mess.Split(«#»);
    byte[] array = new byte[array_bytes.Length-1];
    for(int j=0;j<array_bytes.Length-1;j++)
    {
      array[j]=Convert.ToByte(array_bytes[j]);
    }
    return array;
  }
```

В качестве конструктора письма используем класс MessageBuilder.

```
using System.Net.Mail;
namespace mail
{
  public class MessageBuilder
```

```
    {
        MailMessage mail;

            public void BuildAddress(string fromAddr, string
toAddr)
        {
            MailAddress from = new MailAddress(fromAddr);
            MailAddress to = new MailAddress(toAddr);
            mail = new MailMessage(from, to);
        }

        public void BuildText(string cryptoMessage)
        {
            mail.IsBodyHtml = false;
            mail.Body = cryptoMessage;
        }

        public void BuildTheme(string text)
        {
            mail.Subject = text;
        }

        public MailMessage GetResult()
        {
            return mail;
        }
    }
}
```

В нём содержатся методы:

- BuildTheme — метод добавляющий заголовок в письмо
- BuildText — метод вставляющий текст в письмо
- BuildAddress — метод вставляющий адреса отправителя и получателя
- GetResult — метод возвращающий в качестве аргумента письмо по заданным параметрам

Примеры работы программного обеспечения

Рассмотрим пример с использованием разработанного программного обеспечения. Напишем программу реализующую данную библиотеку.

```
using System;
using unit;
using mail;

class program
{
    public static void Main(string[] args)
    {
        MessageSender mess1 = new MessageSender(new
MessageBuilder());
        Console.WriteLine(«Write Key»);
        string key = Console.ReadLine();
        Console.WriteLine(«Input new Message»);
        string message = Console.ReadLine();
        AESCrypto crypto = new AESCrypto(key,message);

        message = «»;

        foreach(byte i in crypto.Cypher)
        {
            message+= i.ToString() + «#»;
        }
                    mess1.CreateMessage(«basta548@gmail.
com»,»artem.mukh@gmail.com»,message,»Hello guys»);
        while(true)
        {
            Console.WriteLine(«Input password»);
            string pass = Console.ReadLine();
            try{
                        mess1.SendMessage(«smtp.outlook.
com»,587,»basta548@gmail.com»,pass);//Проверка данных по паролю
                break;
            }
            catch
            {
                Console.WriteLine(«Error, try again»);
            }
        }
        Console.WriteLine(«Email Send»);

        Console.Write(«Decrypt: «);
        AESCrypto decrypt = new AESCrypto(key,AESCrypto.
stringToByteAES(message));
        Console.WriteLine(«Message: {0}»,decrypt.Message);

        Console.ReadKey(true);
    }
}
```

Пример выполнения программы

Результат выполнения представлен на рис. 5.

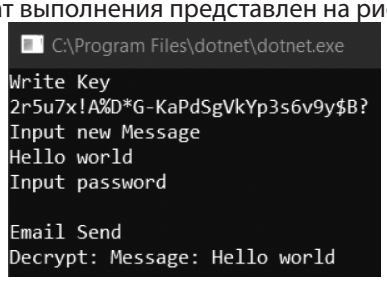


Рис. 5. Результат выполнения программы

Из приведенного примера видно, что программа выполнила отправку Email.

При этом если мы зайдём на почтовый клиент, в нашем примере мы отправляли сообщение на Gmail, оно

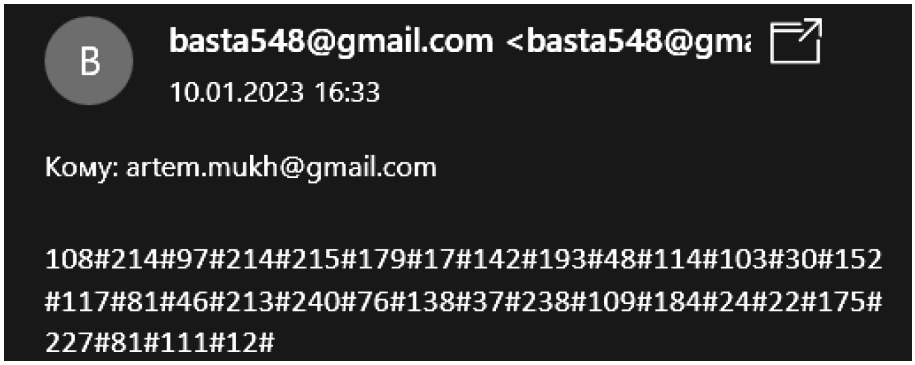


Рис. 6. Вид хранения данных на сервере

будет храниться на сервере в зашифрованном виде Рис. 6.

Выводы

В результате проведённой работы были создана библиотека, используя, которую без установки дополнительных надстроек на существующую информационную систему предприятий, можно реализовать зашифрованную переписку, между двумя и более устройствами. Данная библиотека предназначена для установки на конечные персональные устройства пользователей. Благодаря алгоритму шифрования AES256 дешифрование данных без ключа, практически невозможна, что позволит обеспечит необходимый уровень конфиденциальности.

ЛИТЕРАТУРА

1. Описание алгоритма AES // studbooks URL: https://studbooks.net/2088709/informatika/opisanie_algoritma. (дата обращения: 04.01.2023).
2. Алгоритм выбора раундового ключа // lektsii URL: https://lektsii.org/3-32204.html (дата обращения: 04.01.2023).
3. Что такое SMTP-протокол и как он устроен? // selectel URL: https://selectel.ru/blog/smtp-protocol/ (дата обращения: 04.01.2023).
4. Таненбаум Э.С., Уэзеролл Д. Компьютерные сети. — 5-е изд. изд. — СПБ: Издательский Дом ПИТЕР, 2019. — 960 с.
5. Таненбаум Э.С., Уэзеролл Д. Архитектура компьютера. — 6-е изд. изд. — СПБ: Издательский Дом ПИТЕР, 2018. — 816 с.
6. .NET // Microsoft-Learn URL: https://learn.microsoft.com/ru-ru/dotnet/ (дата обращения: 04.01.2023).
7. AES шифрование и Android клиент // Habr URL: https://habr.com/ru/company/rambler_and_co/blog/279835/ (дата обращения: 04.01.2023).
8. Криптоалгоритмы. Классификация с точки зрения количества ключей // habr URL: https://habr.com/ru/post/336578/ (дата обращения: 04.01.2023).
9. SmtpClient Класс // Learn Microsoft URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.net.mail.smtpclient?view=net-7.0 (дата обращения: 04.01.2023).
10. System.Security.Cryptography Пространство имен // Learn Microsoft URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.security.cryptography?view=net-7.0 (дата обращения: 04.01.2023).

Журнал «Современная наука: актуальные проблемы теории и практики»