Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

«ИССЛЕДОВАНИЕ АСИММЕТРИЧНЫХ ШИФРОВ»

Студент: Белицкий В.Д.

ФИТ 3 курс 5 группа

Преподаватель:

Савельева Маргарита Геннадьевна

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования C# и позволяет выполнить 2 задачи:

* зашифрование сообщения, состоящего из ФИО;
* расшифрование сообщения;
* оценка времени выполнения операций зашифрования и расшифрования.
* генерация сверхвозрастающей последовательности (тайного ключа);
* вычисление нормальной последовательности (открытого ключа);

1. **Методика выполнения поставленных задач**

Для реализации генерации сверхвозрастающей последовательности была разработана функция, представленная на рисунке 2.1.

public int[] Generate(int z)

{

Random rnd = new Random();

int[] k = new int[z];

int sum = 0;

for (int i = 0; i < z; i++)

{

k[i] = rnd.Next(sum, sum + 23);

sum += k[i];

}

return

Рис. 2.1 – Реализация генерации тайного ключа

Внутри метода создается экземпляр класса **Random** для генерации случайных чисел. Затем создается пустой массив **k** размером **z**, который будет хранить сгенерированные числа. Также создается переменная **sum**, которая будет использоваться для накопления суммы предыдущих чисел.

Для вычисления нормальной последовательности была разработана функция, представленная на рисунке 2.2.

public int[] getNorm(int[] d, int a, int n, int z)

{

int[] e = new int[z];

for (int i = 0; i < z; i++)

{

e[i] = (d[i] \* a) % n;

}

return e;

}

Рис. 2.2 – Реализация вычисления открытого ключа

Данная функция вычисляет элементы последовательности *e*={} по формуле: *=* *×a* (mod *n*), где *di* – члены тайного ключа, n больше суммы всех чисел последовательности, причем НОД (*a*,*n*) = 1.

Для зашифрования сообщения *M* каждый его символ *mi* был представлен в двоичной форме. Функция для шифрования, представленная на рисунке 2.3

public int[] getcipher(int[] e, string M, int z)

{

int j = 0;

int[] result = new int[M.Length];

int total = 0;

Console.Write("Исходное сооб M: ");

foreach (char Mi in M)

{

total = 0;

string Mi2 = '0' + GetBytes(Mi.ToString()); //110010

Console.Write($"{Mi2} ");

for (int i = 0; i < Mi2.Length; i++)

{

if (Mi2[i] == '1') total += e[i];

}

result[j] = total;

j++;

}

return result;

}

Рис. 2.3 – Реализация шифрования сообщения

Для расшифрования сообщения был использован сгенерированный тайный ключ.

Реализация данной функции представлена на рисунке 2.4.

public string decipher(int[] d, int Si, int z)

{

string res = "";

string res2 = "";

for (int i = z; i > 0; i--)

{

if (Si >= d[i - 1])

{

res += '1';

Si = Si - d[i - 1];

}

else

{

res += '0';

}

}

for (int i = res.Length - 1; i > -1; i--)

{res2 += res[i];}

return res2;

Рис. 2.4 – Реализация функции расшифрования

Результат выполнения данного консольного приложения представлен на рисунке 2.5.

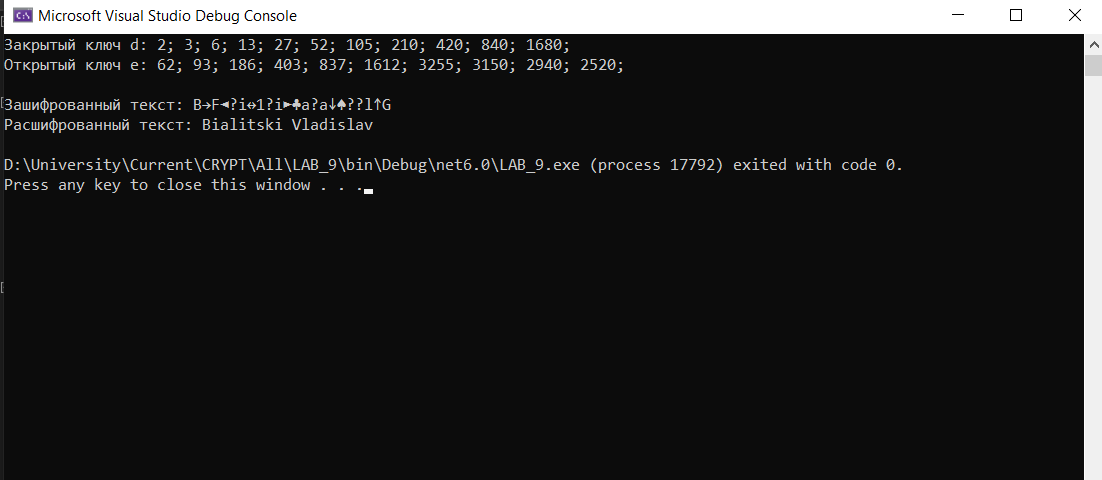


Рис. 2.5 – Результат работы приложения

Также, нами была оценена скорость выполнения зашифрования/расшифрования сообщения с помощью встроенной возможности C# – Stopwatch. Вычисленное время составило 8 мс и 3мс соответственно, что является неплохим результатом.

Сравним результаты работы при увеличении символов и использовании других кодировок. Посмотрим на рисунок 2.6 и 2.7.

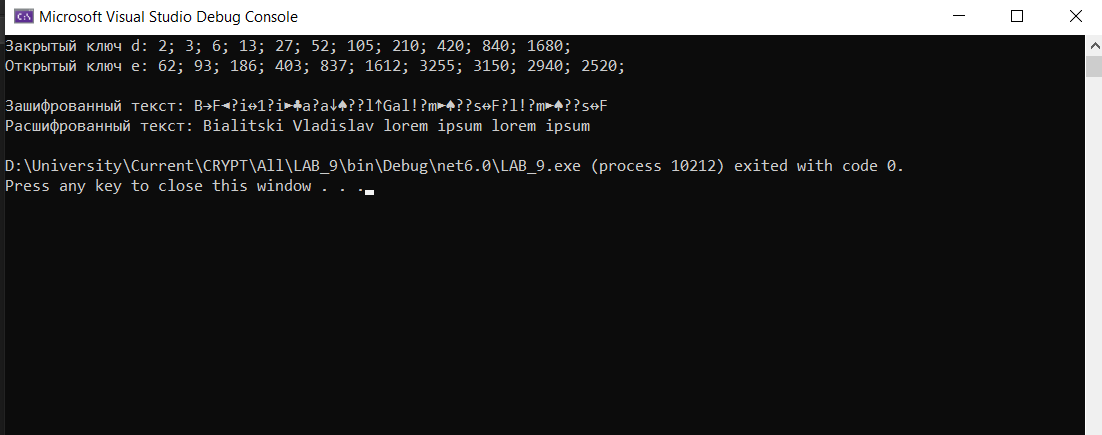


Рис. 2.6 – Результат работы приложения с разными данными

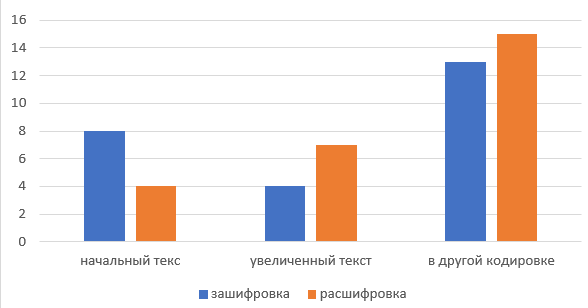


Рис. 2.7 – График зависимости времени по разным критериям

По результатам анализа графика можно заметить, что при изменении кодировки происходит значительный скачок во времени выполнения. Однако, при увеличении размера текста с использованием той же кодировки, время выполнения обычно не увеличивается. Это явление встречается редко, но возможно.

В моем случае, поскольку текст содержит множество повторяющихся символов, процесс кодирования был выполнен быстрее. Это объясняется тем, что ранее зашифрованные символы уже хранятся в кэше и повторно используются.

Выводы

В результате выполнения лабораторной работы были получены практические навыки разработки и использования приложений для реализации асимметричных шифров. Было разработано приложение, которое реализует методы генерации ключевой информации и ее использования. Также была оценена скорость выполнения операций шифрования и расшифрования.