

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Санкт-Петербургский государственный технологический институт (технический университет)»**

**(СПбГТИ(ТУ))**

Факультет **информационных технологий и управления**

Кафедра **систем автоматизированного проектирования и управления**

**ОТЧЕТ**

**по дисциплине**

**«Информационная безопасность»**

**Лабораторная работа №4**

**«Разработка программного продукта»**

Выполнил студент 475 группы:

Овчинников Р.С.

Преподаватели: Кузнецова Г.В.

Макарук Р.В.

Санкт-Петербург

2020

**Цель работы**

Целью данной лабораторной работы является разработка программного продукта с элементами шифрования.

**Задача работы:**

Программа, реализующая алгоритм симметричного шифрования ГОСТ 28147-89 для сообщения. Необходимо зашифровать и дешифровать текст.

**Теория**

Алгоритм шифрования ГОСТ 28147-89 является симметричным, блочным алгоритмом. Преобразование осуществляется над блоком размером 64 бита, размер секретного ключа 256 бит, в алгоритме 32 раунда преобразований.

Необходимые определения и обозначения:

X – блок открытого текста размером 64 бита;

Y – блок зашифрованного текста размером 64 бита;

K – секретный ключ (256 бит);

W – раундовый ключ.

В алгоритме ГОСТ используются следующие операции:

S-блок или S-box – табличная подстановка, при которой группа бит отображается в другую группу бит;

⊞ – операция сложения по модулю 232;

⊕ (или XOR) – операция сложения по модулю 2 (или побитовое «исключающее или»);

11 – циклический сдвиг влево на 11 бит.

Эти операции циклически повторяются в алгоритме, образуя так называемые раунды. Входом каждого раунда является выход предыдущего раунда и раундовый подключ *W*, который получен из секретного ключа шифрования *K* следующим образом. Рассмотрим секретный ключ *K* (256 бит), состоящий из восьми слов по 32 бита: 

На их основе строим раундовый ключ *W*:



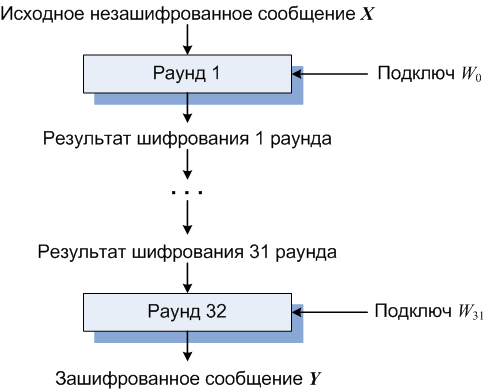


Рисунок 1 – Основная схема алгоритма ГОСТ

Для шифрования блок открытого текста сначала разбивается на две одинаковые части, правую *R* (младшее слово) и левую *L* (старшее слово).

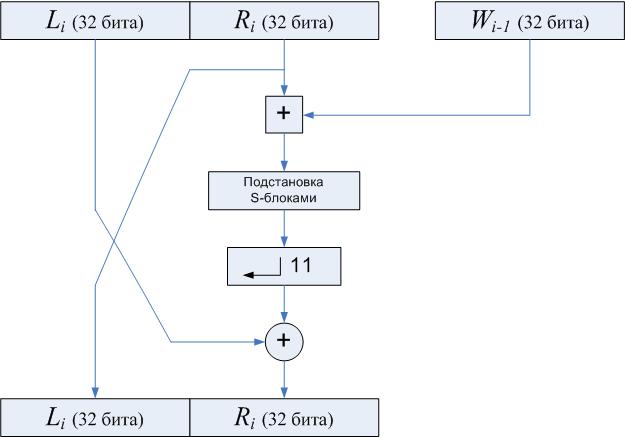


Рисунок 2 – Схема одного раунда алгоритма ГОСТ

На i-м раунде используется подключ . Правая часть  складывается по модулю 232 с раундовым подключом . Над получившимся результатом выполняется операция табличной подстановки.

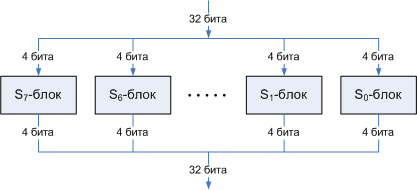


Рисунок 3 – Подстановка S-блоками

Для этого результат разбивается на восемь 4-битовых кусочков, каждый из которых подается на вход своего S-блока: первые четыре бита в S0-блок, вторые – в S1-блок и так далее. Каждый S-блок содержит 16 четырехбитовых элементов, нумеруемых с 0 по 15. ГОСТ рекомендует заполнять каждую из восьми таблиц различными числами множества {0, 1, 2, …, 15}, переставленными случайным образом.

По входным четырем битам определяется номер элемента в S-блоке, который поступает на выход. Выходы всех восьми S-блоков объединяются в 32-битовое слово, затем все слово циклически сдвигается влево на 11 бит. Наконец, результат объединяется с помощью XOR с левой половиной, и получается новая правая половина, а правая половина становится левой половиной. Эти операции выполняются 32 раза. После этого левая и правая части меняются местами.

Для шифрования и дешифрования сообщения используется один алгоритм. Единственным различием является генерация раундового ключа. Чтобы дешифровать блок, строим раундовый ключ



подаем на вход *Y* и на выходе получаем *X*.

**Блок-схема алгоритма**

Блок-схемы алгоритма симметричного шифрования ГОСТ 28147-89 представлены на рисунках 4, 5 и 6.



Рисунок 4 – Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89



Рисунок 5 – Схема цикла зашифрования



Рисунок 6 – Схема цикла расшифрования

Министерство образования и науки РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

"Санкт-Петербургский государственный технологический институт

(технический университет)"

Утверждаю

Зав. кафедрой САПР и У

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_Чистякова Т.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« » 2020

Программа, реализующая алгоритм симметричного шифрования

ГОСТ 28147-89

Описание применения

Лист утверждения

460.2.031.0007-01 31 01-1-ЛУ

Руководитель Кузнецова Г.В.

гр. 475

Овчинников Р.С.

Санкт - Петербург

2020

Утвержден

460.2.031.0007-01 31 01-1-ЛУ

Программа, реализующая алгоритм симметричного шифрования

ГОСТ 28147-89

Описание применения

460.2.031.0007-01 31 01-1

Листов: 10

Санкт-Петербург

2020

2

460.2.031.0007-01 31 01-1

#### АННОТАЦИЯ

Данная программа предназначена для шифрования и расшифрования теста алгоритмом симметричного шифрования ГОСТ 28147-89.

В данном программном документе, в разделе «Назначение программы» приведено описание назначения программы, возможности данной программы, а также ее основные характеристики и ограничения, накладываемые на область применения программы.

В разделе «Условия применения» указаны условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам, и другим программам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера).

В данном программном документе, в разделе «Описание задачи» указаны определения задачи и методы ее решения.

В разделе «Входные и выходные данные» указаны сведения о входных и выходных данных.

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД ГОСТ 19.502-781.

Продукт реализован в среде Visual Studio 2017.

3

460.2.031.0007-01 31 01-1

СОДЕРЖАНИЕ

[1 НАЗНАЧЕНИЕ 11](#_Toc59059469)

[1.1 Назначение программы 11](#_Toc59059470)

[1.2 Возможности программы 11](#_Toc59059471)

[1.3 Ограничения, накладываемые на область применения программы 11](#_Toc59059472)

[2 УСЛОВИЯ ПРИМЕНЕНИЯ 12](#_Toc59059473)

[2.1 Требования к техническим (аппаратным) средствам 12](#_Toc59059474)

[2.2 Требования к программным средствам (другим программам) 12](#_Toc59059475)

[3 ОПИСАНИЕ ЗАДАЧИ 13](#_Toc59059476)

[3.1 Определения задачи программы 13](#_Toc59059477)

[4 ВХОДНЫЕ И ВЫХОДНЫЕ ПЕРЕМЕННЫЕ 18](#_Toc59059478)

[4.1 Сведения о входных данных 18](#_Toc59059479)

[4.2 Сведения о выходных данных 18](#_Toc59059480)

[ПРИЛОЖЕНИЕ 19](#_Toc59059481)

# **НАЗНАЧЕНИЕ**

# **Назначение программы**

Программа, реализующая алгоритм симметричного шифрования ГОСТ 28147-89

# **Возможности программы**

Программа позволяет:

* вводить 32 байтовый ключ в текстовое поле;
* загрузить 32 байтовый ключ из файла;
* сохранить 32 байтовый ключ в файл;
* вводить текста для шифрования;
* сохранить текст для шифрования в файл;
* сохранить шифрованный текст в файл;
* вводить шифрованный текс в текстовое поле;
* загрузить шифрованный текст;
* сохранить результат расшифровки в файл;
* просмотреть информацию о разработчике и программе;

# **Ограничения, накладываемые на область применения программы**

1. Программа предназначена для работы на операционной системе семейства Windows (Windows 8, Windows 10). Работа под управлением других версий ОС не предусмотрена.

2. Программа не позволяет зашифровать/расшифровать текст длиной больше 10.000 символов. Иначе ожидание превышало бы 5 минут.

3. Программа не позволяет зашифровать/расшифровать пустой текст.

4. Программа не сможет зашифровать/расшифровать пустой текст, без секретного ключа длиной 32 символа.

5. Программа не позволяет зашифровать/расшифровать текст, длина которого не делится на 8.

5

460.2.031.0007-01 31 01-1

# **УСЛОВИЯ ПРИМЕНЕНИЯ**

# **Требования к техническим (аппаратным) средствам**

В состав минимальных технических средств должен входить IBM PC-совместимый компьютер на базе микропроцессора Intel® Core(ТМ) i7-4702MQ (2,2 ГГц), ОЗУ 16 Гб, монитор ЖК (17”), CD-ROM дисковод, клавиатура, мышь.

# **Требования к программным средствам (другим программам)**

В состав минимльных программных средств должна входить операционная система Windows 8.1, среда разработки Visual Studio 2017, текстовый процессор MS Office Word 2017.

6

460.2.031.0007-01 31 01-1

# **ОПИСАНИЕ ЗАДАЧИ**

# **Определения задачи программы**

Основная задача, решаемая программой – шифрование текста.

Вводится 32 символьный ключ, который будет использоваться при шифровании или расшифровании. Затем вводится текст, который будет зашифрован или расшифрован и нажимаем на кнопку “Шифровать” или “Расшифровать”

Выполнение программы:

* + - 1. При запуске программы появляется окно на рисунке 7.

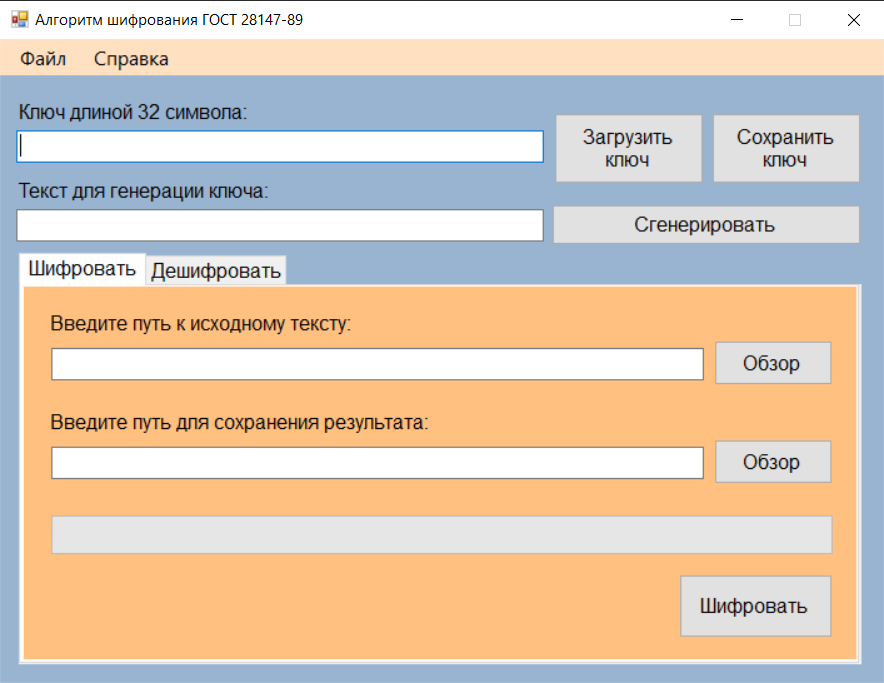


Рисунок 7 – Окно шифрования текста

7

460.2.031.0007-01 31 01-1

2. Вводим вручную, загружаем из файла или генерируем по годовому слову 32 символьный ключ, представлено на рисунке 8.

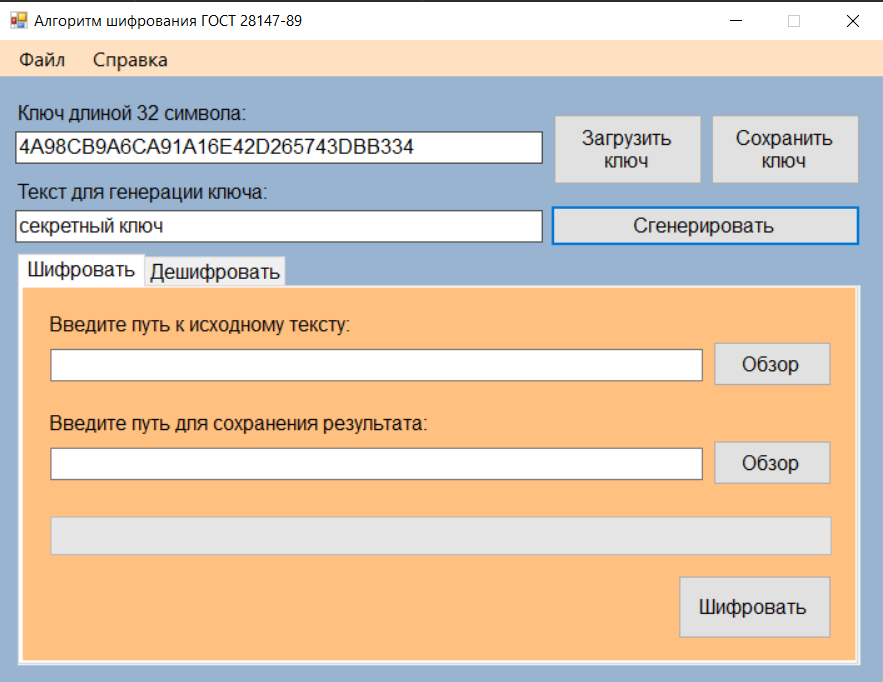


Рисунок 8 – Ввод 32 символьного ключа

3. Вводим путь к файлу с исходным текстом (рисунок 9).

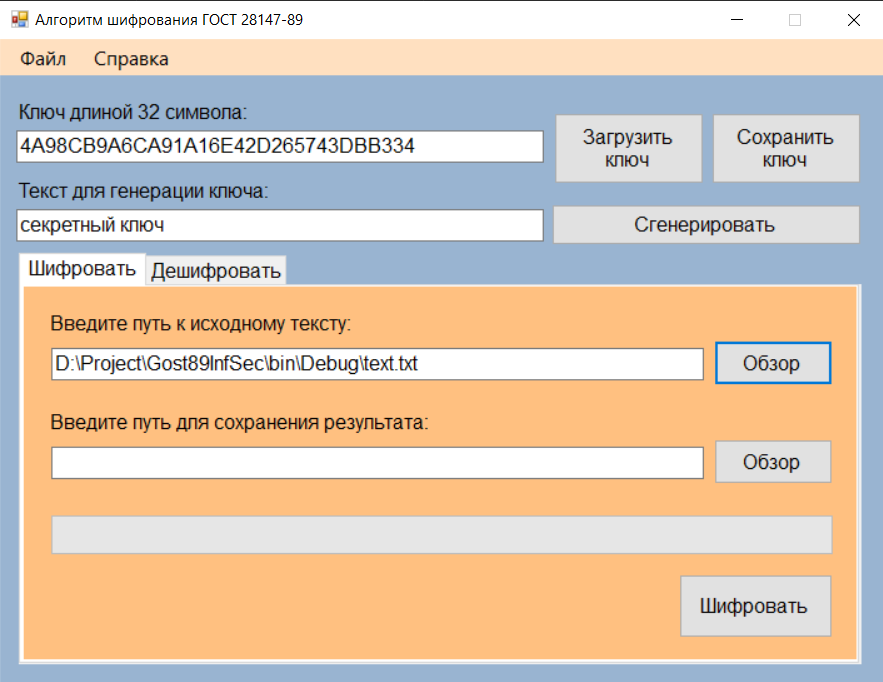


Рисунок 9 – Введён путь к исходному тексту

8

460.2.031.0007-01 31 01-1

4. Вводим путь для сохранения результата шифрования

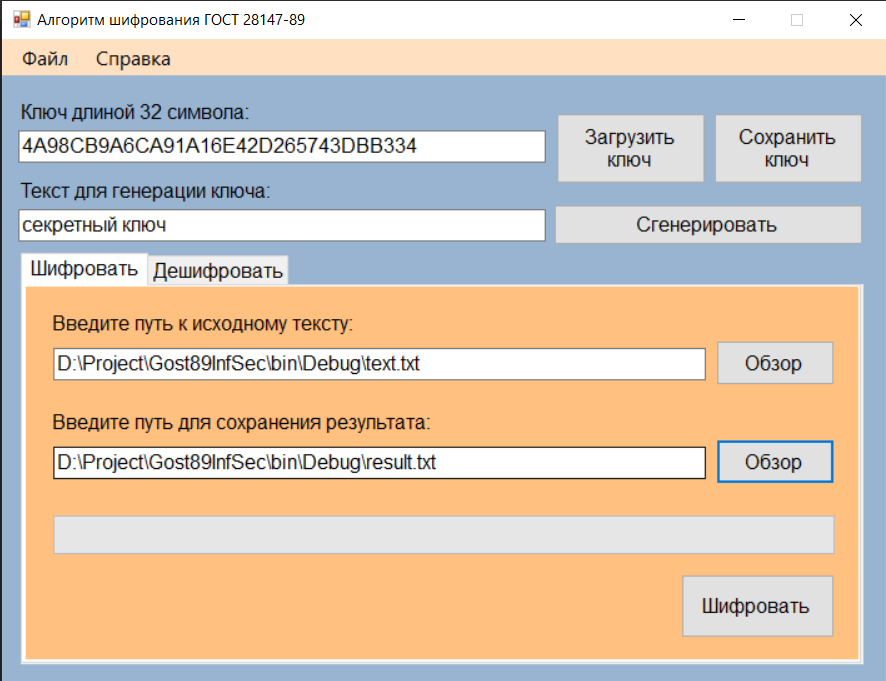


Рисунок 10 – Путь для сохранения результата

5. Нажимаем кнопку “Шифровать” и ожидаем завершения загрузки.

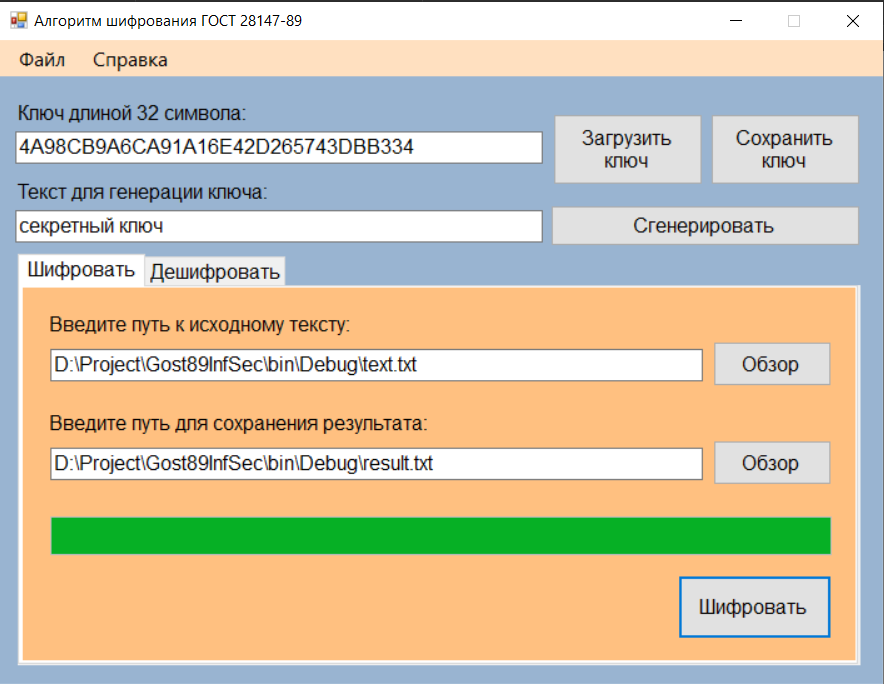


Рисунок 12 – Успешное шифрование и сохранение в файл

9

460.2.031.0007-01 31 01-1

7. Переходим на вкладку “Дешифровать”, представлено на рисунке 13.

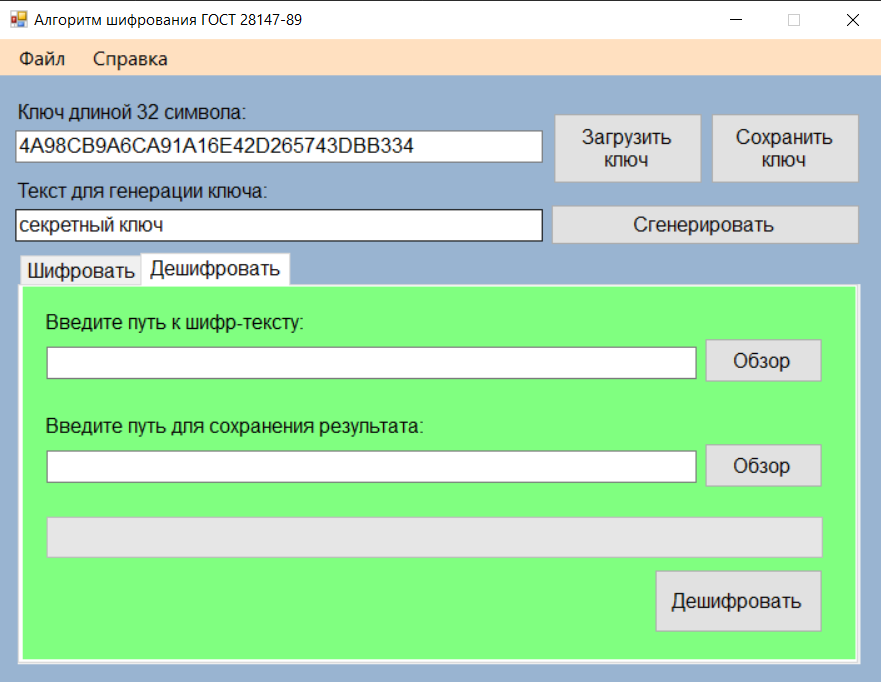


Рисунок 13 – Вкладка для дешифрования текста

8. Вводим путь к шифрованному тексту и файлу, куда запишется результат дешифрования.

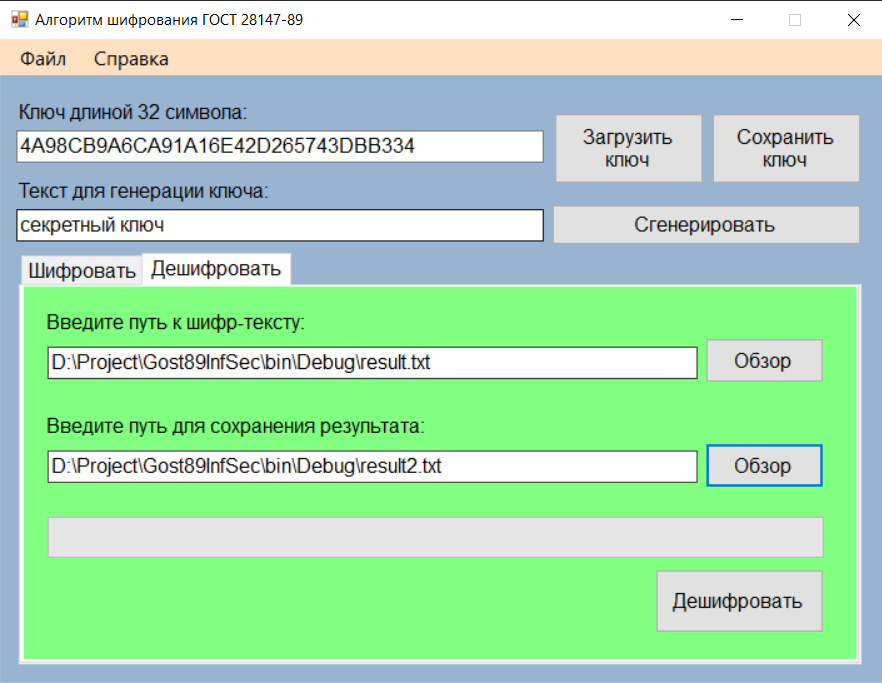


Рисунок 14 – Ввод данных

9. Нажимаем на кнопку “Дешифровать” и ждём окончания загрузки.

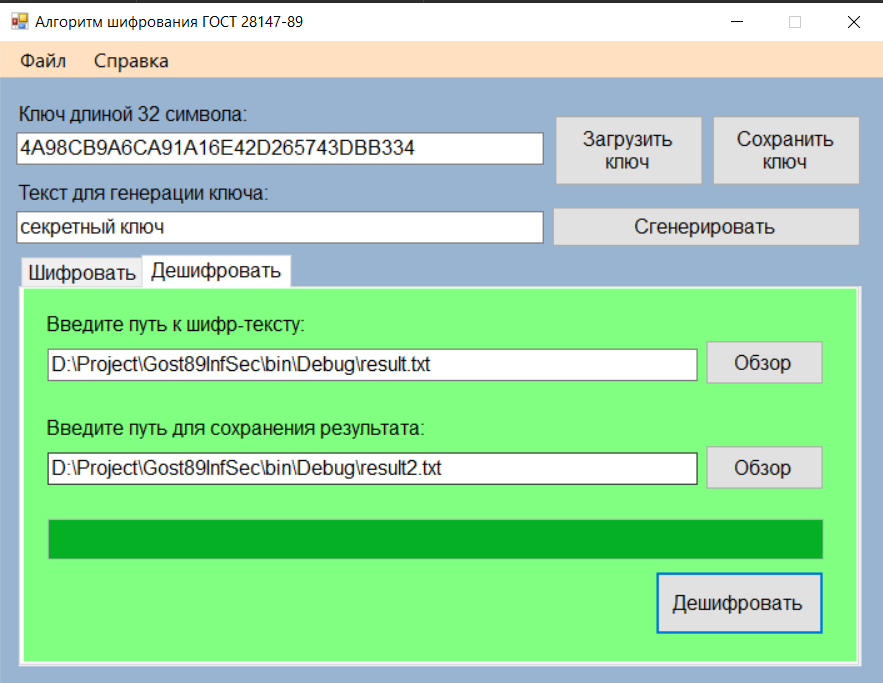


Рисунок 15 – Нажатие на кнопку “Дешифровать”

10

460.2.031.0007-01 31 01-1

# **ВХОДНЫЕ И ВЫХОДНЫЕ ПЕРЕМЕННЫЕ**

# **Сведения о входных данных**

При входе мы вводим 32 символьный ключ, который будет использоваться для работы с текстом, путь к исходному тексту, а также путь для сохранения результата шифрования

# **Сведения о выходных данных**

На выходе мы получаем текст, который был зашифрован. Для проверки текст можно расшифровать.

# **ПРИЛОЖЕНИЕ**

[---Блок E32.cs---]

using System.Collections.Generic;

namespace WinGost

{

class E32 : Converter

{

readonly byte[] \_encrByteFile;

readonly uint[] \_uintKey;

ulong[] \_ulongFile;

public List<string> otv = new List<string>();

private E32() { }

public E32(byte[] file, byte[] key)

{

\_uintKey = GetUIntKeyArray(key);

\_ulongFile = GetULongDataArray(file);

\_encrByteFile = ConvertToByte(EncryptFile());

}

public byte[] GetEncryptFile

{

get { return \_encrByteFile; }

}

private ulong[] EncryptFile()

{

BasicStep[] K = new BasicStep[8];

ulong[] ulongEncrFile = new ulong[\_ulongFile.Length];

//цикла зашифрования

for (int k = 0; k < \_ulongFile.Length; k++)

{

ulongEncrFile[k] = \_ulongFile[k];

for (int j = 0; j < 3; j++)

{

for (int i = 0; i <= 7; i++)

{

K[i] = new BasicStep(ulongEncrFile[k], \_uintKey[i]);

ulongEncrFile[k] = K[i].BasicEncrypt(false);

}

}

for (int i = 7; i >= 0; i--)

{

K[i] = new BasicStep(ulongEncrFile[k], \_uintKey[i]);

if (i != 0)

ulongEncrFile[k] = K[i].BasicEncrypt(false);

else

ulongEncrFile[k] = K[i].BasicEncrypt(true);

}

}

return ulongEncrFile;

}

}

}

[---Конец блока E32.cs---]

[---Блок D32.cs---]

using System;

namespace WinGost

{

/// <summary>

/// Дешефратор

/// </summary>

class D32 : Converter

{

byte[] decrByteFile;

uint[] uintKey;

ulong[] ulongFile;

private D32() { }

public D32(byte[] file, byte[] key)

{

uintKey = GetUIntKeyArray(key);

ulongFile = GetULongDataArray(file);

decrByteFile = ConvertToByte(DecryptFile());

}

public byte[] GetDecryptFile

{

get { return decrByteFile; }

}

private ulong[] DecryptFile()

{

BasicStep[] K = new BasicStep[8];

ulong[] ulongDecrFile = new ulong[ulongFile.Length];

//цикла дешифрования

for (int k = 0; k < ulongFile.Length; k++)

{

ulongDecrFile[k] = ulongFile[k];

for (int i = 0; i <= 7; i++)

{

K[i] = new BasicStep(ulongDecrFile[k], uintKey[i]);

ulongDecrFile[k] = K[i].BasicEncrypt(false);

}

for (int j = 0; j < 3; j++)

{

for (int i = 7; i >= 0; i--)

{

K[i] = new BasicStep(ulongDecrFile[k], uintKey[i]);

if ((j == 2) && (i == 0))

ulongDecrFile[k] = K[i].BasicEncrypt(true);

else

ulongDecrFile[k] = K[i].BasicEncrypt(false);

}

}

}

return ulongDecrFile;

}

}

}

[---Конец блока D32.cs---]

[---Блок BasicStep.cs ---]

using System;

namespace WinGost

{

struct BasicStep

{

uint N1, N2, X;

public BasicStep(ulong dateFragment, uint keyFragment)

{

N1 = (uint)(dateFragment >> 32);

N2 = (uint)((dateFragment << 32) >> 32);

X = keyFragment;

}

public ulong BasicEncrypt(bool IsLastStep)

{

return (FourthAndFifthStep(IsLastStep, ThirdStep(SecondStep(FirstStep()))));

}

// Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^32

//с используемым на шаге элементом ключа, результат передается на следующий шаг

private uint FirstStep()

{

return (uint)((X + N1) % (Convert.ToUInt64(Math.Pow(2, 32))));

}

//Поблочная замена. 32-битовое значение, полученное на предыдущем шаге,

//интерпретируется как массив из восьми 4-битовых блоков кода S=(S0,S1,S2,S3,S4,S5,S6,S7).

//Производится замена по таблице

private uint SecondStep(uint S)

{

uint newS, S0, S1, S2, S3, S4, S5, S6, S7;

S0 = S >> 28;

S1 = (S << 4) >> 28;

S2 = (S << 8) >> 28;

S3 = (S << 12) >> 28;

S4 = (S << 16) >> 28;

S5 = (S << 20) >> 28;

S6 = (S << 24) >> 28;

S7 = (S << 28) >> 28;

S0 = ReplacementTable.Table0[S0];

S1 = ReplacementTable.Table0[0x10 + S1];

S2 = ReplacementTable.Table0[0x20 + S2];

S3 = ReplacementTable.Table0[0x30 + S3];

S4 = ReplacementTable.Table0[0x40 + S4];

S5 = ReplacementTable.Table0[0x50 + S5];

S6 = ReplacementTable.Table0[0x60 + S6];

S7 = ReplacementTable.Table0[0x70 + S7];

newS = S7 + (S6 << 4) + (S5 << 8) + (S4 << 12) + (S3 << 16) +

(S2 << 20) + (S1 << 24) + (S0 << 28);

return newS;

}

//Циклический сдвиг на 11 бит влево.

private uint ThirdStep(uint S)

{

return (uint)(S << 11) | (S >> 21);

}

//Побитовое сложение и Сдвиг по цепочке

//Полученное значение преобразуемого блока возвращается

//как результат выполнения ал-горитма основного шага криптопреобразования.

private ulong FourthAndFifthStep(bool IsLastStep, uint S)

{

ulong N;

S = (S ^ N2);

if (!IsLastStep)

{

N2 = N1;

N1 = S;

}

else

N2 = S;

N = ((ulong)N2) | (((ulong)N1) << 32); // Побитовое ИЛИ (or)

return N;

}

}

}

[---Конец блока BasicStep.cs ---]

[---Блок ReplacementTable.cs ---]

using System;

namespace WinGost

{

// Структура с таблицами подстановок

struct ReplacementTable

{

internal static byte[] Table0

{

get { return table0; }

}

private static readonly byte[] table0 = {

0x4,0x2,0xF,0x5,0x9,0x1,0x0,0x8,0xE,0x3,0xB,0xC,0xD,0x7,0xA,0x6,

0xC,0x9,0xF,0xE,0x8,0x1,0x3,0xA,0x2,0x7,0x4,0xD,0x6,0x0,0xB,0x5,

0xD,0x8,0xE,0xC,0x7,0x3,0x9,0xA,0x1,0x5,0x2,0x4,0x6,0xF,0x0,0xB,

0xE,0x9,0xB,0x2,0x5,0xF,0x7,0x1,0x0,0xD,0xC,0x6,0xA,0x4,0x3,0x8,

0x3,0xE,0x5,0x9,0x6,0x8,0x0,0xD,0xA,0xB,0x7,0xC,0x2,0x1,0xF,0x4,

0x8,0xF,0x6,0xB,0x1,0x9,0xC,0x5,0xD,0x3,0x7,0xA,0x0,0xE,0x2,0x4,

0x9,0xB,0xC,0x0,0x3,0x6,0x7,0x5,0x4,0x8,0xE,0xF,0x1,0xA,0x2,0xD,

0xC,0x6,0x5,0x2,0xB,0x0,0x9,0xD,0x3,0xE,0x7,0xA,0xF,0x4,0x1,0x8

};

}

}

[---Конец блока ReplacementTable.cs ---]