ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Разработка систем аутентификации и криптографии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Алгоритмы криптографии и подпись приложений»

**Выполнил:**

Магистрант гр. N42514c

Балданова Юлия Батоевна

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Проверил:**

Федоров Иван Романович

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург, 2020

Оглавление

[Цель работы 3](#_Toc55680322)

[Описание выбранных средств реализации и обоснования выбора 4](#_Toc55680323)

[Описание алгоритма 5](#_Toc55680324)

[Исходный код 7](#_Toc55680325)

[Демонстрация работы программы 13](#_Toc55680326)

[Подпись файла 16](#_Toc55680327)

[Выводы 17](#_Toc55680328)

# Цель работы

**Часть 1.** Реализация алгоритма шифрования **DES**:

* необходимо реализовать процедуры генерации ключей, шифрования и дешифрования без использования криптографических библиотек;
* программа должна запускаться в среде Windows, исполняемый файл программы должен иметь расширение .EXE.

**Часть 2**. Подпись полученного в первой части файла .EXE:

* необходимо подписать полученный файл .EXE с помощью команд Windows Power Shell;
* при открытии «Свойств» файла .EXE в разделе «Цифровые подписи» должна быть подпись студента.

# Описание выбранных средств реализации и обоснования выбора

Для реализации алгоритма шифрования был выбран язык Python, а для реализации интерфейса была выбрана библиотека – tkinter.

Python – это язык программирования общего назначения, нацеленный на написание разного рода программ (веб-/десктоп приложения, игры, и т.д.) без ощутимых проблем так как он обладает огромной библиотекой (набор функций, которые доступны без дополнительной настройки). Синтаксис во многом лаконичный и читаемый, так как сами создатели стремились создать простой и понятный широкому кругу людей язык программирования.

# Описание алгоритма

DES – алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый в 1977 году как официальный стандарт. Шифрование происходит на основе ключа длиной 64 бита, из которых 56 приходятся непосредственно на шифрование, а 8 бит – это системные разряды. Шифрование строится на начальной перестановке, 6 циклов шифрования, а после шифрования производится преобразование, обратное первичному.

Схема алгоритма шифрования DES выглядит следующим образом:

1. Исходный текст (64 бит) преобразуется c помощью начальной перестановки, которая определяется таблицей IP.
2. Полученные данные участвуют в 16 циклах преобразования с помощью функции Фейстеля. Блок данных разбивается на две части по 32 бита: L(0) и R(0). В алгоритме DES используются прямое преобразование сетью Фейстеля в шифровании и обратное преобразование сетью Фейстеля в дешифрование.

Пусть *T*i−1=*L* i−1*R* i−1 результат (i-1) итерации, тогда результат i-ой итерации *T*i=*L* i*R* i определяется:

*Li=Ri−1*

*Ri=Li−1⊕f(Ri−1, ki)*

1. Ключи ki получаются из начального ключа k (56 бит) следующим образом. Добавляются биты в позиции 8, 16, 24, 32, 40, 48, 56, 64 ключа k таким образом, чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64).
2. Функция f участвующая в 16-ти циклах преобразования играет роль шифрования и вычисляется как:
   1. Функция расширения E. Правая половина из 32 битов растягивается до 48 битов и перемешивается. Это помогает рассеиванию связи между входными битами и выходными. Перестановка с расширением выбирается так, чтобы один входной бит воздействовал на две замены через S-блоки.
   2. Функция XOR с ключом k. К строке из 48 битов, полученной после перестановки с расширением, и *ki* применяется операция XOR, т. е. каждая пара соответствующих битов складывается по модулю 2.
   3. Преобразование S-блоков. Каждый 6-битовый кусок передается в один из восьми S-блоков, где он превращается в набор из 4 битов. Каждый S-блок представляет собой поисковую таблицу из четырех строк и шестнадцати столбцов. Шесть входящих в S-блок битов определяют, какую строку и какой столбец необходимо использовать для замены. Первый и шестой бит задают номер строки, а остальные – номер столбца. Выход S-блока – значение соответствующей ячейки таблицы.
   4. Перестановка P. 8 групп 4-битовых элементов, которые комбинируются в 32-битовую строку и перемешиваются, формируя выход функции F.
3. Конечная перестановка обратна начальной.

Основные достоинства симметричного алгоритма шифрования DES:

* используется только один ключ длиной 56 битов;
* относительная простота алгоритма обеспечивает высокую скорость обработки информации;
* достаточно высокая стойкость алгоритма.

Но несмотря на все достоинства данного метода шифрования, в настоящий момент DES признан ненадежным и не рекомендован к использованию.

# Исходный код

Файл des.py – реализован сам алгоритм.

subKeyList = 16 \* [[None] \* 8]  
  
# Таблица для начальной перестановки  
IP = (58, 50, 42, 34, 26, 18, 10, 2,  
 60, 52, 44, 36, 28, 20, 12, 4,  
 62, 54, 46, 38, 30, 22, 14, 6,  
 64, 56, 48, 40, 32, 24, 16, 8,  
 57, 49, 41, 33, 25, 17, 9, 1,  
 59, 51, 43, 35, 27, 19, 11, 3,  
 61, 53, 45, 37, 29, 21, 13, 5,  
 63, 55, 47, 39, 31, 23, 15, 7)  
  
# Таблица для E расширения  
E = (32, 1, 2, 3, 4, 5,  
 4, 5, 6, 7, 8, 9,  
 8, 9, 10, 11, 12, 13,  
 12, 13, 14, 15, 16, 17,  
 16, 17, 18, 19, 20, 21,  
 20, 21, 22, 23, 24, 25,  
 24, 25, 26, 27, 28, 29,  
 28, 29, 30, 31, 32, 1)  
  
# Таблица для P перестановки  
P = (16, 7, 20, 21, 29, 12, 28, 17,  
 1, 15, 23, 26, 5, 18, 31, 10,  
 2, 8, 24, 14, 32, 27, 3, 9,  
 19, 13, 30, 6, 22, 11, 4, 25)  
  
# Таблица для обратной перестановки  
F = (40, 8, 48, 16, 56, 24, 64, 32,  
 39, 7, 47, 15, 55, 23, 63, 31,  
 38, 6, 46, 14, 54, 22, 62, 30,  
 37, 5, 45, 13, 53, 21, 61, 29,  
 36, 4, 44, 12, 52, 20, 60, 28,  
 35, 3, 43, 11, 51, 19, 59, 27,  
 34, 2, 42, 10, 50, 18, 58, 26,  
 33, 1, 41, 9, 49, 17, 57, 25)  
  
sBox = 8 \* [64 \* [0]]  
  
# Таблица для S преобразований  
sBox[0] = (14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,  
 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,  
 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,  
 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13)  
  
sBox[1] = (15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,  
 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,  
 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,  
 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9)  
  
sBox[2] = (10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,  
 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,  
 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,  
 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12)  
  
sBox[3] = (7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,  
 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,  
 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,  
 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14)  
  
sBox[4] = (2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,  
 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,  
 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,  
 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3)  
  
sBox[5] = (12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,  
 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,  
 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,  
 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13)  
  
sBox[6] = (4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,  
 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,  
 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,  
 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12)  
  
sBox[7] = (13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,  
 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,  
 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,  
 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11)  
  
  
def bittoByte(bitList):  
 *"""Преобразование битового списка в байтовый"""* return [int("".join(map(str, bitList[i \* 8:i \* 8 + 8])), 2) for i in range(len(bitList) // 8)]  
  
  
def bytetoBit(byteList):  
 *"""Преобразование байтового списка в битовый"""* return [(byteList[i // 8] >> (7 - (i % 8))) & 0x01 for i in range(8 \* len(byteList))]  
  
  
def permuteBitList(inputBitList, permTable):  
 *"""Перестановка входных данных"""* return [inputBitList[e - 1] for e in permTable]  
  
  
def permByteList(inByteList, permTable):  
 *"""Перестановка входных данных"""* outByteList = (len(permTable) >> 3) \* [0]  
 for index, elem in enumerate(permTable):  
 i = index % 8  
 e = (elem - 1) % 8  
 if i >= e:  
 outByteList[index >> 3] |= \  
 (inByteList[(elem - 1) >> 3] & (128 >> e)) >> (i - e)  
 else:  
 outByteList[index >> 3] |= \  
 (inByteList[(elem - 1) >> 3] & (128 >> e)) << (e - i)  
 return outByteList  
  
  
def getIndex(inBitList):  
 *"""Перестановка битов для правильной индексации в S-блоках"""* return (inBitList[0] << 5) + (inBitList[1] << 3) + \  
 (inBitList[2] << 2) + (inBitList[3] << 1) + \  
 (inBitList[4] << 0) + (inBitList[5] << 4)  
  
  
def padData(string):  
 *"""Доведение строки до нужной длины (добавление)"""* padLength = 8 - (len(string) % 8)  
 return [ord(s) for s in string] + padLength \* [padLength]  
  
  
def unpadData(byteList):  
 *"""Доведение строки до нужной длины (удаление)"""* return "".join(chr(e) for e in byteList[:-byteList[-1]])  
  
  
def setKey(keyByteList):  
 *"""Генерация 16 подключей для циклов шифрования"""* # Перестановка для расширенного ключа  
 PC1table = (57, 49, 41, 33, 25, 17, 9,  
 1, 58, 50, 42, 34, 26, 18,  
 10, 2, 59, 51, 43, 35, 27,  
 19, 11, 3, 60, 52, 44, 36,  
 63, 55, 47, 39, 31, 23, 15,  
 7, 62, 54, 46, 38, 30, 22,  
 14, 6, 61, 53, 45, 37, 29,  
 21, 13, 5, 28, 20, 12, 4)  
  
 PC2table = (14, 17, 11, 24, 1, 5, 3, 28,  
 15, 6, 21, 10, 23, 19, 12, 4,  
 26, 8, 16, 7, 27, 20, 13, 2,  
 41, 52, 31, 37, 47, 55, 30, 40,  
 51, 45, 33, 48, 44, 49, 39, 56,  
 34, 53, 46, 42, 50, 36, 29, 32)  
  
 def leftShift(inKeyBitList, round):  
 *"""1 или 2 циклических сдвига влево"""* LStable = (1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1)  
  
 outKeyBitList = 56 \* [0]  
 if LStable[round] == 2:  
 outKeyBitList[:26] = inKeyBitList[2:28]  
 outKeyBitList[26] = inKeyBitList[0]  
 outKeyBitList[27] = inKeyBitList[1]  
 outKeyBitList[28:54] = inKeyBitList[30:]  
 outKeyBitList[54] = inKeyBitList[28]  
 outKeyBitList[55] = inKeyBitList[29]  
 else:  
 outKeyBitList[:27] = inKeyBitList[1:28]  
 outKeyBitList[27] = inKeyBitList[0]  
 outKeyBitList[28:55] = inKeyBitList[29:]  
 outKeyBitList[55] = inKeyBitList[28]  
 return outKeyBitList  
  
 permKeyBitList = permuteBitList(bytetoBit(keyByteList), PC1table)  
  
 for round in range(16):  
 auxBitList = leftShift(permKeyBitList, round)  
 subKeyList[round] = bittoByte(permuteBitList(auxBitList, PC2table))  
 permKeyBitList = auxBitList  
  
  
def encryptBlock(inputBlock):  
 *"""Шифрование 8-ми байтового блока"""* inputData = permByteList(inputBlock, IP)  
 leftPart, rightPart = inputData[:4], inputData[4:]  
 for round in range(16):  
 expRightPart = permByteList(rightPart, E)  
 key = subKeyList[round]  
 indexList = bytetoBit([i ^ j for i, j in zip(key, expRightPart)])  
 sBoxOutput = 4 \* [0]  
 for nBox in range(4):  
 nBox12 = 12 \* nBox  
 leftIndex = getIndex(indexList[nBox12:nBox12 + 6])  
 rightIndex = getIndex(indexList[nBox12 + 6:nBox12 + 12])  
 sBoxOutput[nBox] = (sBox[nBox << 1][leftIndex] << 4) + \  
 sBox[(nBox << 1) + 1][rightIndex]  
 aux = permByteList(sBoxOutput, P)  
 newRightPart = [i ^ j for i, j in zip(aux, leftPart)]  
 leftPart = rightPart  
 rightPart = newRightPart  
 return permByteList(rightPart + leftPart, F)  
  
  
def decryptBlock(inputBlock):  
 *"""Дешифрование 8-ми байтового блока"""* inputData = permByteList(inputBlock, IP)  
 leftPart, rightPart = inputData[:4], inputData[4:]  
 for round in range(16):  
 expRightPart = permByteList(rightPart, E)  
 key = subKeyList[15 - round]  
 indexList = bytetoBit([i ^ j for i, j in zip(key, expRightPart)])  
 sBoxOutput = 4 \* [0]  
 for nBox in range(4):  
 nBox12 = 12 \* nBox  
 leftIndex = getIndex(indexList[nBox12:nBox12 + 6])  
 rightIndex = getIndex(indexList[nBox12 + 6:nBox12 + 12])  
 sBoxOutput[nBox] = (sBox[nBox \* 2][leftIndex] << 4) + \  
 sBox[nBox \* 2 + 1][rightIndex]  
 aux = permByteList(sBoxOutput, P)  
 newRightPart = [i ^ j for i, j in zip(aux, leftPart)]  
 leftPart = rightPart  
 rightPart = newRightPart  
 return permByteList(rightPart + leftPart, F)  
  
  
def encrypt(key, inString):  
 *"""Шифрование исходного текста с входным ключом"""* setKey(key)  
 inByteList, outByteList = padData(inString), []  
 for i in range(0, len(inByteList), 8):  
 outByteList += encryptBlock(inByteList[i:i + 8])  
 return outByteList  
  
  
def decrypt(key, inByteList):  
 *"""Дешифрование текста с входным ключом"""* setKey(key)  
 outByteList = []  
 for i in range(0, len(inByteList), 8):  
 outByteList += decryptBlock(inByteList[i:i + 8])  
 return unpadData(outByteList)

Файл main.py – файл, в котором реализован интерфейс алгоритма

# Интерфейс  
from des import \*  
import random  
import tkinter  
from tkinter import messagebox as mb  
from tkinter import \*  
  
  
class Interface(tkinter.Frame):  
 def \_\_init\_\_(self, master):  
 tkinter.Frame.\_\_init\_\_(self, master, background="grey")  
 self.master = master  
 self.initUI()  
  
 def initUI(self):  
 self.master.title("DES")  
 self.master['bg'] = 'grey'  
 self.master.geometry("410x300+500+300")  
  
# Задает начало алгоритма  
def Start():  
 try:  
 mb.showinfo("Информация", "Сгенирируйте ключ")  
 key\_text.configure(state='disabled')  
 text.configure(state='disabled')  
 generate\_keys.configure(state='active')  
 encrypt\_text.configure(state='disabled')  
 clear.configure(state='active')  
 except OSError:  
 mb.showerror("Ошибка", "Что-то пошло не так...")  
  
# Генерирует псевно-случайный ключ  
def generate\_key():  
 try:  
 key = [int(random.getrandbits(8)) for i in range(8)]  
 key\_text.configure(state='normal')  
 key\_text.insert("1.0", key)  
 generate\_keys.configure(state='disabled')  
 text.configure(state='normal')  
 generate\_keys.configure(state='active')  
 encrypt\_text.configure(state='active')  
 clear.configure(state='active')  
 except OSError:  
 mb.showerror("Ошибка", "Что-то пошло не так...")  
  
# Шифрование  
def encryptText():  
 try:  
 key = [int(i) for i in key\_text.get("1.0", END).strip().split(' ')]  
 plaintext = text.get("1.0", END)  
 cipher\_text = encrypt(key, plaintext)  
 text.delete(1.0, END)  
 text.insert("1.0", cipher\_text)  
 text.configure(state='disabled')  
 encrypt\_text.configure(state='disabled')  
 decrypt\_text.configure(state='active')  
 except ValueError:  
 mb.showerror("Ошибка", "Пустое поле")  
 except OSError:  
 mb.showerror("Ошибка", "Что-то пошло не так...")  
  
# Дешифрование  
def decryptText():  
 try:  
 key = [int(i) for i in key\_text.get("1.0", END).strip().split(' ')]  
 message\_encrypt = [int(i) for i in text.get("1.0", END).strip().split(' ')]  
 plaintext = decrypt(key, message\_encrypt)  
 text.configure(state='normal')  
 text.delete(1.0, END)  
 text.insert("1.0", plaintext)  
 encrypt\_text.configure(state='active')  
 decrypt\_text.configure(state='disabled')  
 except ValueError:  
 mb.showerror("Ошибка", "Пустое поле")  
 except OSError:  
 mb.showerror("Ошибка", "Что-то пошло не так...")  
  
# Очистка формы  
def clearFrame():  
 start\_butt.configure(state='active')  
 key\_text.delete(1.0, END)  
 key\_text.configure(state='disabled')  
 generate\_keys.configure(state='disabled')  
 text.delete(1.0, END)  
 text.configure(state='disabled')  
 encrypt\_text.configure(state='disabled')  
 decrypt\_text.configure(state='disabled')  
 clear.configure(state='disabled')  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tkinter.Tk()  
 app = Interface(root)  
  
 position = {'ipadx': 5, 'ipady': 2, 'padx': 10, 'pady': 10, 'sticky': 'nswe'}  
  
 start\_butt = tkinter.Button(root)  
 start\_butt.configure(text='Начать', font='Arial 10', command=Start)  
 start\_butt.grid(row=1, column=1, columnspan=3, \*\*position)  
  
 key\_text = tkinter.Text(root)  
 key\_text.configure(width=25, height=1)  
 key\_text.configure(state='disabled')  
 key\_text.grid(row=2, rowspan=2, column=1, columnspan=1, \*\*position)  
  
 generate\_keys = tkinter.Button(root)  
 generate\_keys.configure(text='Сгенерировать', font='Arial 10', command=generate\_key)  
 generate\_keys.grid(row=2, column=2, columnspan=2, \*\*position)  
  
 label\_text = tkinter.Label(root, text='Сообщение', font='Arial 10 bold')  
 label\_text.grid(row=4, column=1, columnspan=3, \*\*position)  
  
 text = tkinter.Text(root)  
 text.configure(width=25, height=5)  
 text.configure(state='disabled')  
 text.grid(row=5, column=1, columnspan=3, \*\*position)  
  
 encrypt\_text = tkinter.Button(root)  
 encrypt\_text.configure(text='Зашифровать', font='Arial 10', command=encryptText)  
 encrypt\_text.configure(state='disabled')  
 encrypt\_text.grid(row=6, column=1, \*\*position)  
  
 decrypt\_text = tkinter.Button(root)  
 decrypt\_text.configure(text='Расшифровать', font='Arial 10', command=decryptText)  
 decrypt\_text.configure(state='disabled')  
 decrypt\_text.grid(row=6, column=2, \*\*position)  
  
 clear = tkinter.Button(root)  
 clear.configure(text='Очистить', font='Arial 10', command=clearFrame)  
 clear.configure(state='disabled')  
 clear.grid(row=6, column=3, \*\*position)  
  
 root.mainloop()

# Демонстрация работы программы

Для открытия программы необходимо запустить файл des.exe.

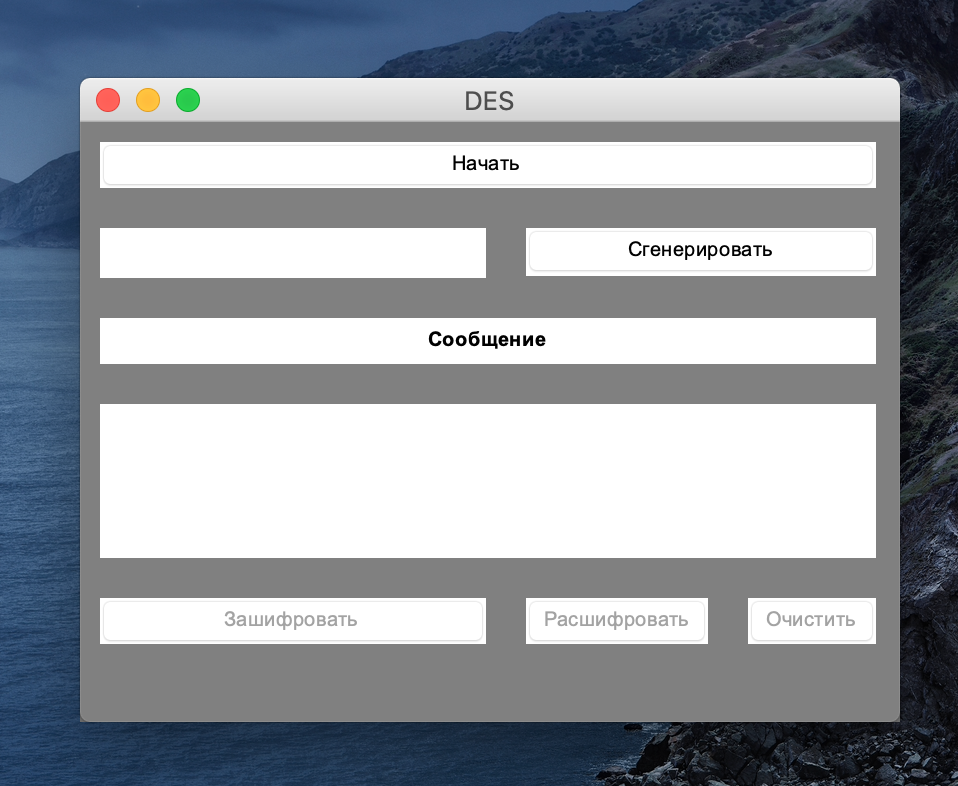


Рисунок 1 – Интерфейс программы

При открытии программы необходимо нажать на кнопку Начать (Рис. 2).

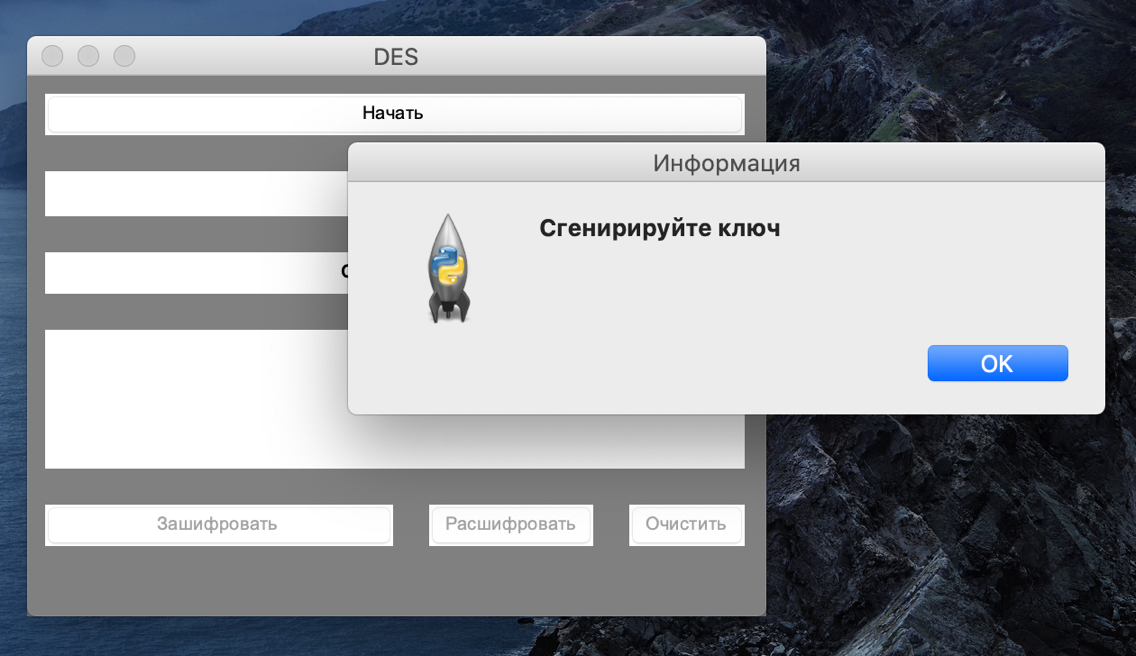


Рисунок 2 – Запуск программы

После необходимо нажать на кнопку Сгенерировать, в поле появится сгенерированный ключ (Рис. 3)

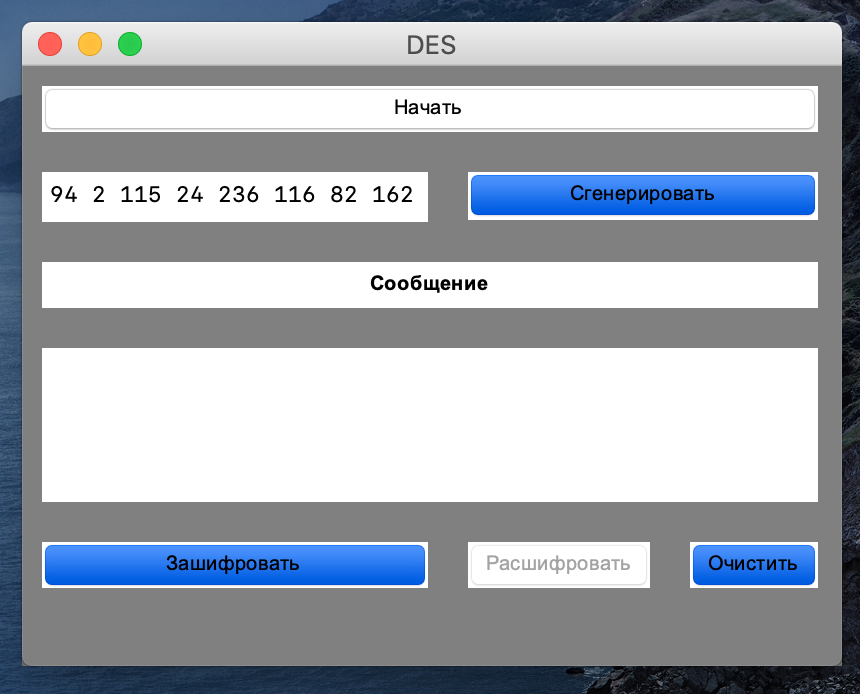


Рисунок 3 – Сгенерированный ключ

После в поле для ввода необходимо ввести исходное сообщение и нажать на кнопку Зашифровать (Рис. 4, 5).

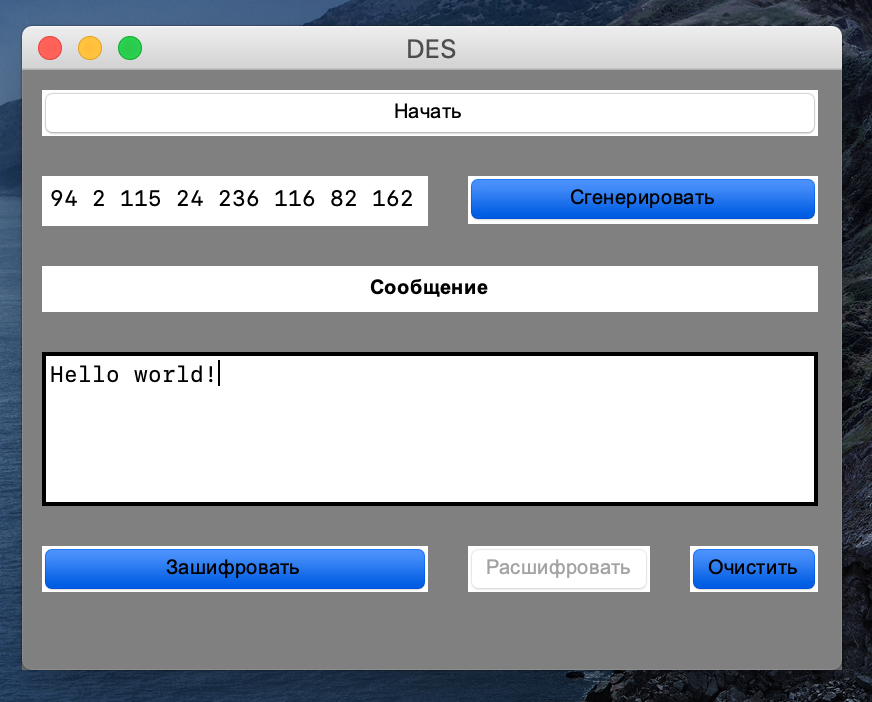


Рисунок 4 – Шифрование сообщения

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Зашифрованное сообщение

Для расшифровки сообщения необходимо нажать на кнопку Расшифровать (Рис. 6).

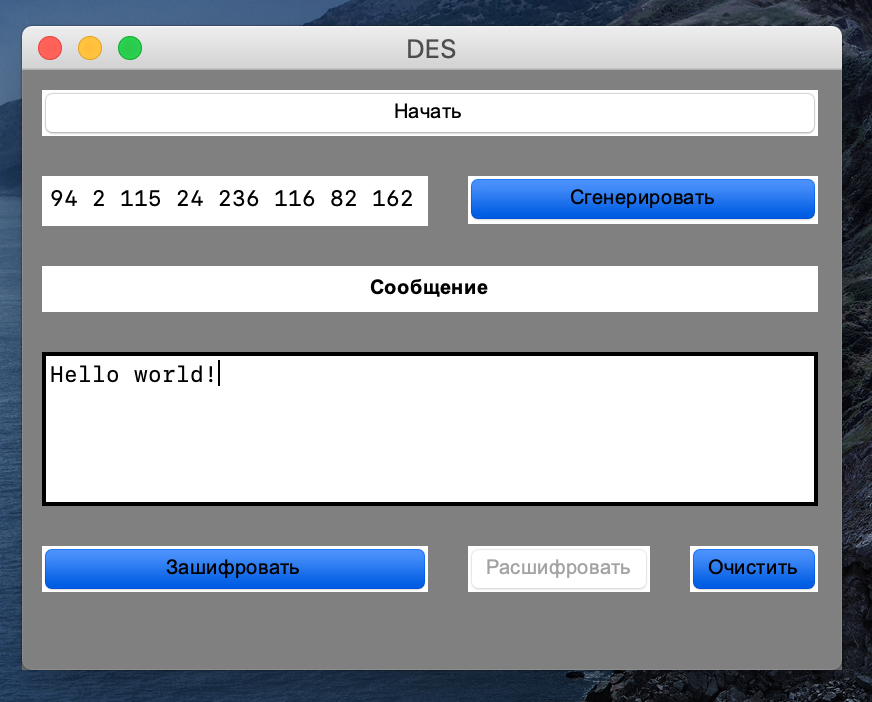


Рисунок 6 – Расшифрованное сообщение

Для очистки формы необходимо нажать на кнопку Очистить, и программа придет к первоначальному виду (Рис. 1).

# Подпись файла

Для подписания .EXE файл сертификатом необходимо:

1. Создать сертификат:

Команда: New-SelfSignedCertificate -Type Custom -Subject "CN= Baldanova, O=ITMO, C=RU" -KeyUsage DigitalSignature -FriendlyName "Baldanova" -CertStoreLocation "Cert:\CurrentUser\My"

1. Задать переменной cert только что созданный сертификат:

Команда: $cert=Get-ChildItem -Path cert:\CurrentUser\my -CodeSigningCert

1. Подписать .EXE файл этим сертификатом командой:

Команда: Set-AuthenticodeSignature des.exe $cert

1. Файл подписан.

# Выводы

В ходе лабораторной работы был изучен алгоритм симметричного шифрования DES, а также процесс подписи приложений с использованием Windows Power Shell. В результате лабораторной работы был получен исходный код алгоритма шифрования на языке программирования Python, после чего скрипт был преобразован в файл .EXE и подписан с помощью PKI Client.