Группа ИБ-221	Лабораторная работа № 2
Торгуш М.А.	Симметричное шифрование

Цель: изучить симметричное шифрование и практически реализовать алгоритмы шифровки и дешифровки.

Теоретическая часть

Блочные шифры обрабатывают кодируемое сообщение блоками из нескольких байт, при этом блок открытого текста X преобразуется в блок шифротекста Y того же размера с использованием некоторого ключа шифрования key:

Y=Encrypt(X, key)

Процедура дешифрации выполняет обратное преобразование, используя тот же самый ключ:

X=Decrypt(Y, key)

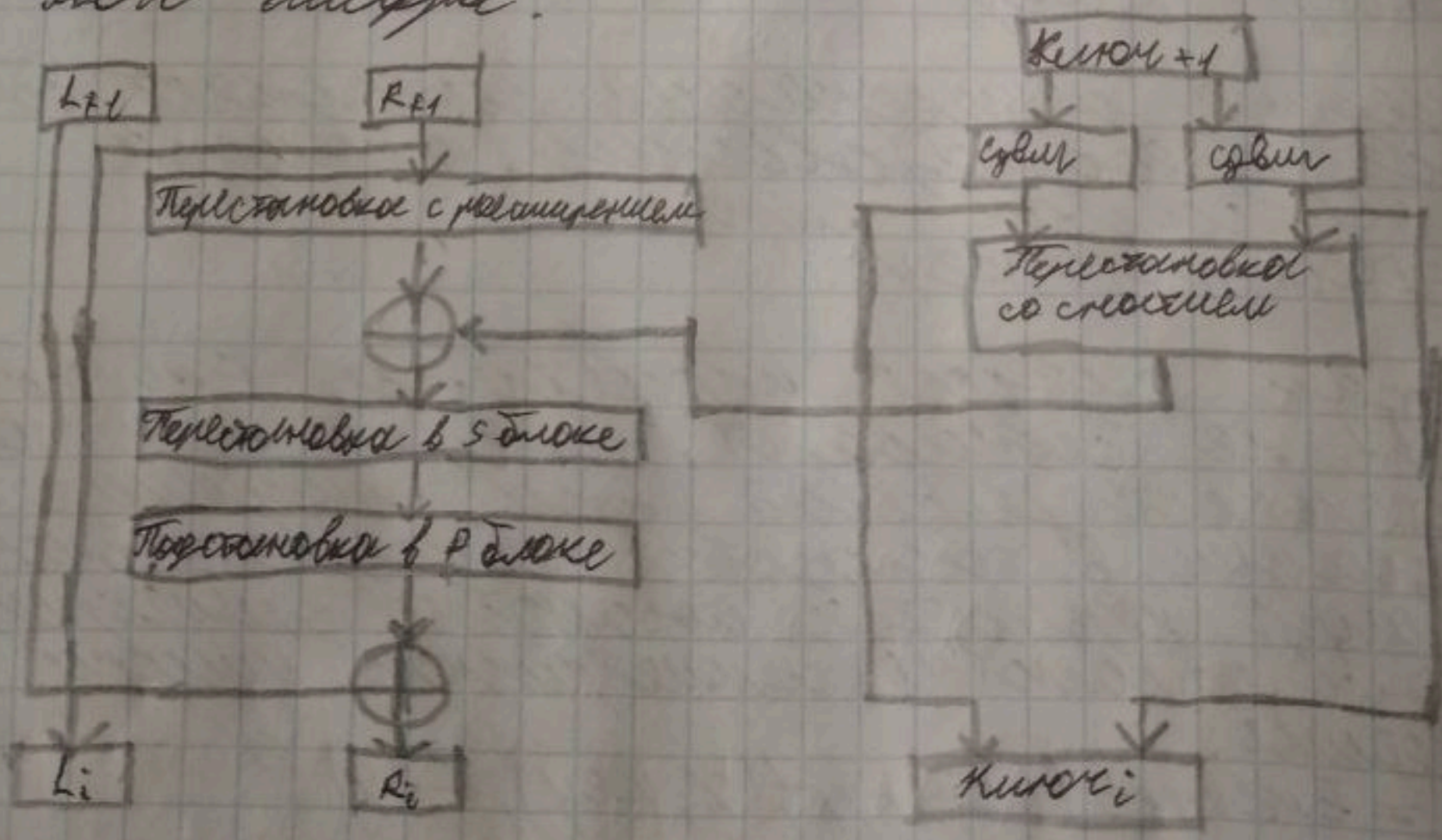
(1) В общем случае процедуры Encrypt и Decrypt не совпадают, однако если последовательность действий для шифрации и дешифрации в точности совпадает, блочный шифр называется абсолютно симметричным. Для абсолютно симметричного шифра справедливо:

X = Encrypt((Encrypt(X, Key), Key)

Одним из основных принципов структурного построения современных криптоалгоритмов является принцип итерирова-

ния. Его идея заключается в многократном (раз) повторении одного блока открытого текста с использованием на каждом шаге специального ключа полученного на основе ключа шифрования. (2) Количество циклов можно варьировать в зависимости от требований криптостойкости и производительности. Каждый цикл: увеличение количества циклов приводит к повышению стойкости шифра, но увеличивает время шифрования и потребляемые вычислительные ресурсы. Подобные циклические структуры принято называть сетями, и большинство современных блочных шифров построены с использованием сетевой архитектуры.

(3) Алгоритм шифрования DES использует в своей основе функции операции подстановки, которые существенно усложняют криптоанализ этого шифра.



Работа алгоритма DES

Каждый раунд преобразования включает перестановку правой части блока, причем на входе перестановки подается 32-разрядное число, а снимается с него 48-разрядное (некоторые биты входного числа копируются на выход дважды). После сложения с ключевым числом, осуществляется табличная подстановка, в результате которой по 8 таблицам подстановки число длиной 48-битного входа на 32-битный выход (каждая S-таблица подстановки заменяет 6-битный вход на 4-битный выход). Выход S-блока поступает на ~~выход~~ ~~подстановки~~ блок перестановки, где биты перестановляются по законам, определяемым специальной P-таблицей. Заканчивается раунд сложением ~~с~~ левой ветвей между собой.

Для повышения криптостойкости рекомендуется применять тройное DES-шифрование на трех или двух различных ключах. При двух ключах шифрование блок открытого текста сначала шифруется на ключе K_1, затем дешифруется на ключе K_2, а затем вновь шифруется на ключе K_1. Размер ключевого пространства (общее количество ключей шифрования) в этом случае возрастает до 2^{112} (у обычного DES – 2^{56}). (3) При использовании трех ключей шифрования блок открытого текста сначала шифруется на ключе K_1, затем дешифруется на ключе K_2, а затем шифруется на ключе K_3. Это обеспечивает размер ключевого пространства 2^{168}.

(3) Из-за недостаточной длины ключа шифрования, а также ориентированности шифра DES на аппаратную реализацию, в 1997 году был объявлен ~~свободный~~ конкурс на криптостандарт блочного шифрования AES. Стандартом AES стал Rijndael, основанный на KALST-сети.

Этот алгоритм представляет блок данных в виде двумерного байтового массива размером 4×4, 4×6 или 4×8

Все операции выполняются с отдельными байтами массива, а также с неповторимыми столбцами и строками. Алгоритм выполняет 4 преобразования байтов массива. BS (Byte Sub) - подстановка байтов массива. SR (ShiftRow) - сдвиг строк массива. MC (MixColumn) - операция над независимыми столбцами массива, когда каждый столбец по определенному правилу умножается на фиксированную матрицу. AK (AddRoundKey) - добавление ключа раундовой матрицы. Достоинством шифра Rijndael является то, что он обеспечивает высокую скорость шифрования на всех платформах. Требования к ресурсам для его работы минимальны.

(3) Российским стандартом блочного шифрования является алгоритм ГОСТ 28147-89, имеющий 64 битную структуру сети Фейстеля. Его особенностью стала крупная блочная длина ключа (256 бит), неудобные таблицы подстановки, большое количество раундов (32 раунда).

(4) Правило Керкгоффса - стойкость шифра должна определяться лишь секретностью ключа. То есть знание алгоритма шифрования НЕ должно повлиять никаким образом на последующее дешифрование.

(5) В своей работе Шеннон дал математическое определение криптосистемы

$S = (M, C, K, E, D)$, где M - множество открытых сообщений, C - множество шифртекстов, K - множество ключей, $E = \{E_k : M \to C; k \in K\}$ - семейство алгоритмов шифрования, $D = \{D_k : C \to M; k \in K\}$ - семейство алгоритмов дешифрования.

и выполняется условие:

$D_k(E_k(m)) = m \quad \forall m \in M, \forall k \in K$

То есть дешифрование всегда возвращает исходное сообщение для любого ключа.

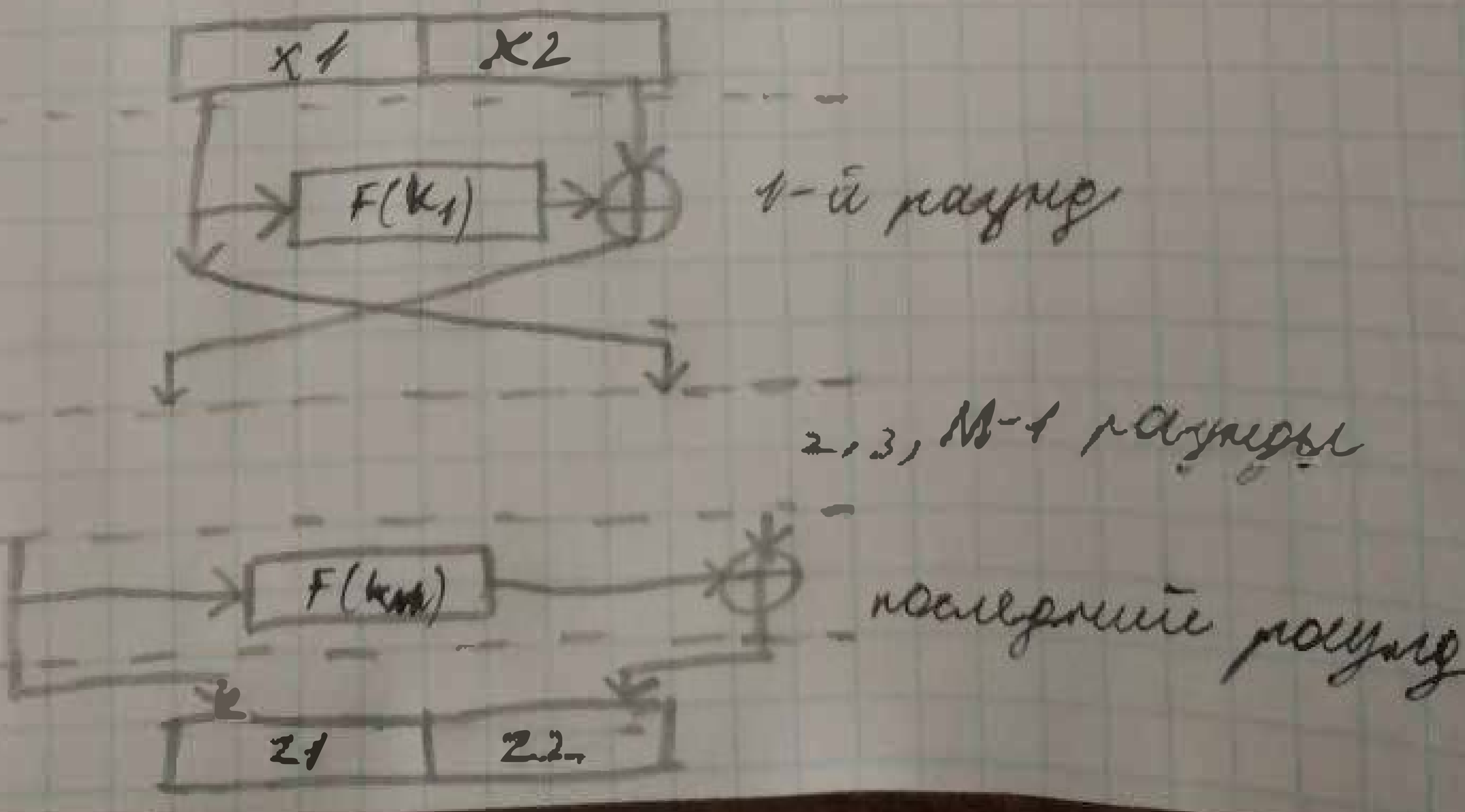
Ввел критерий идеальной стойкости: шифр считается

стая абсолютно стойким, если знание шифротекста не даёт злоумышленнику никакой информации об исходном тексте без ключа.

Доказал, что идеально стойким является только одноразовый блокнот, где ключ имеет длину не меньше сообщения и используется только один раз.

~~[illegible]~~ сформулировал 2 принципа:
confusion (запутывание) — сложная зависимость между шифротекстом и ключом.
Diffusion (рассеивание) — распределение каждого бита открытого текста ~~на~~ по всему шифротексту.

(6) В сети Фейстеля при шифровании блок открытого текста разбивается на две равные части — левую и правую. На каждой ~~[illegible]~~ цикле одна из частей подвергается преобразованию при помощи образующей функции F и вспомогательного ключа K_i, полученного из исходного ключа. Результат преобразования складывается по модулю 2 с другой частью, после чего части меняются местами. Преобразования на каждом цикле идентичны и лишь после последнего раунда выполняется перестановка частей блока.



Сеть Ретинеля реализует принципы Confusion и Difusion.

Вывод: шумы симметричное шифрование и практически реализовал отличительные шифровки и дешифровки.

Описание задания

Алгоритм шифрования 3DES
Написать програму которая шифрует или расшифровывает текстовой файл длины не менее 32 байт. Максимальный размер == 4 Мб.
Программа должна работать в 2-х режимах: шифрования и расшифрования.
Режим шифрование :
Входные параметры : имя исходного , пароль
Выходные параметры : имя   зашифрованного файла
Режим расшифрования :
Входные параметры : имя исходного файла , пароль .
Выходные параметры : имя   расшифрованного файла

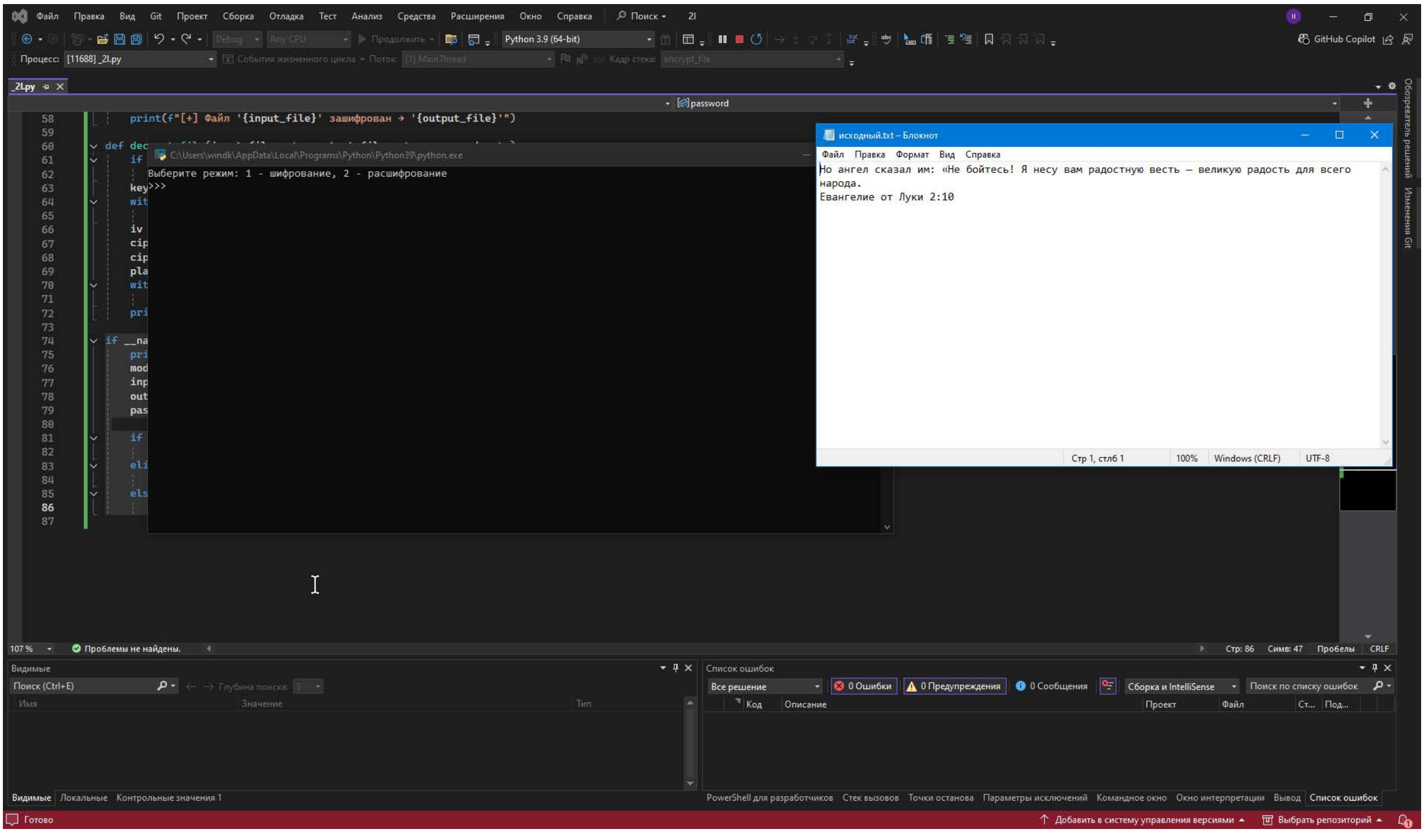
Листинг программы

```
import hashlib
import sys
import os
import msvcrt
from Crypto.Cipher import DES3

MIN_SIZE = 32
MAX_SIZE = 4 * 1024 * 1024  # 4 Мб

def get_password(prompt="Введите пароль: "):
    print(prompt, end="", flush=True)
    password = ""
    while True:
        ch = msvcrt.getch()
        if ch in {b"\r", b"\n"}:  # Enter
            print()
            break
    elif ch == b"\x08":  # Backspace
        if len(password) > 0:
            password = password[:-1]
            sys.stdout.write("\b \b")
            sys.stdout.flush()
    else:
        try:
            char = ch.decode("utf-8")
            password += char
            sys.stdout.write("*")
            sys.stdout.flush()
        except UnicodeDecodeError:
            pass
    return password

# генерация ключа 3DES из пароля
def derive_key(password: str) -> bytes:
    h = hashlib.sha256(password.encode('utf-8')).digest()
    return h[:24]
```

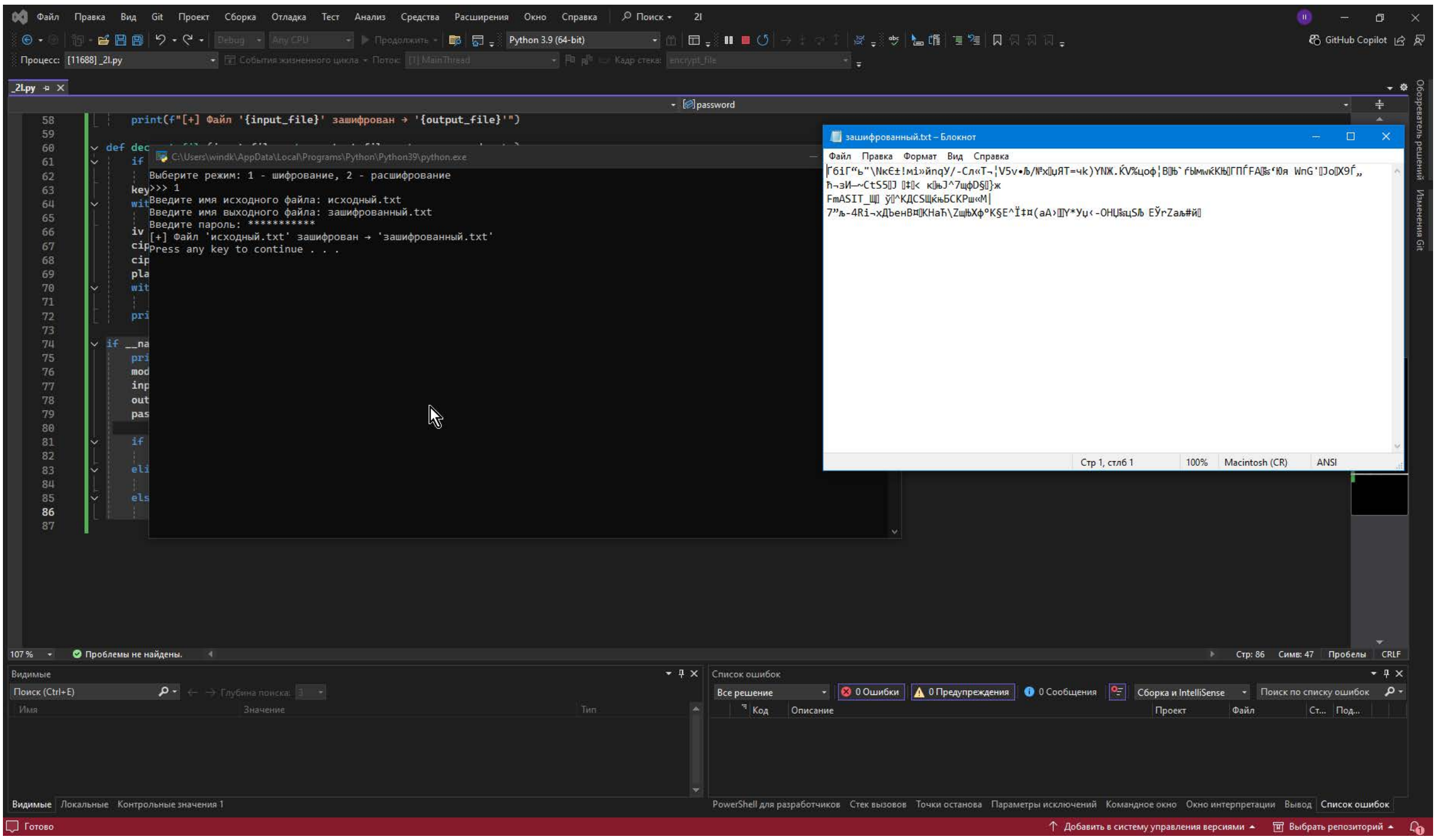
```
def check_file_size(file_path: str) -> bool:
    size = os.path.getsize(file_path)
    if size < MIN_SIZE:
        print(f"[-] Ошибка: файл слишком маленький ({size} байт, минимум
{MIN_SIZE})")
        return False
    if size > MAX_SIZE:
        print(f"[-] Ошибка: файл слишком большой ({size} байт, максимум
{MAX_SIZE})")
        return False
    return True

def encrypt_file(input_file: str, output_file: str, password: str):
    if not check_file_size(input_file):
        return
    key = derive_key(password)
    cipher = DES3.new(key, DES3.MODE_CFB)
    with open(input_file, "rb") as f_in:
        plaintext = f_in.read()
    ciphertext = cipher.iv + cipher.encrypt(plaintext)
    with open(output_file, "wb") as f_out:
        f_out.write(ciphertext)
    print(f"[+] Файл '{input_file}' зашифрован → '{output_file}'")

def decrypt_file(input_file: str, output_file: str, password: str):
    if not check_file_size(input_file):
        return
    key = derive_key(password)
    with open(input_file, "rb") as f_in:
        file_data = f_in.read()
    iv = file_data[:8]
    ciphertext = file_data[8:]
    cipher = DES3.new(key, DES3.MODE_CFB, iv=iv)
    plaintext = cipher.decrypt(ciphertext)
    with open(output_file, "wb") as f_out:
        f_out.write(plaintext)
    print(f"[+] Файл '{input_file}' расшифрован → '{output_file}'")
```

```
if __name__ == "__main__":
    print("Выберите режим: 1 - шифрование, 2 - расшифрование")
    mode = input(">>> ").strip()
    input_file = input("Введите имя исходного файла: ").strip()
    output_file = input("Введите имя выходного файла: ").strip()
    password = get_password("Введите пароль: ")

    if mode == "1":
        encrypt_file(input_file, output_file, password)
elif mode == "2":
    decrypt_file(input_file, output_file, password)
else:
    print("[-] Ошибка: неизвестный режим")
```

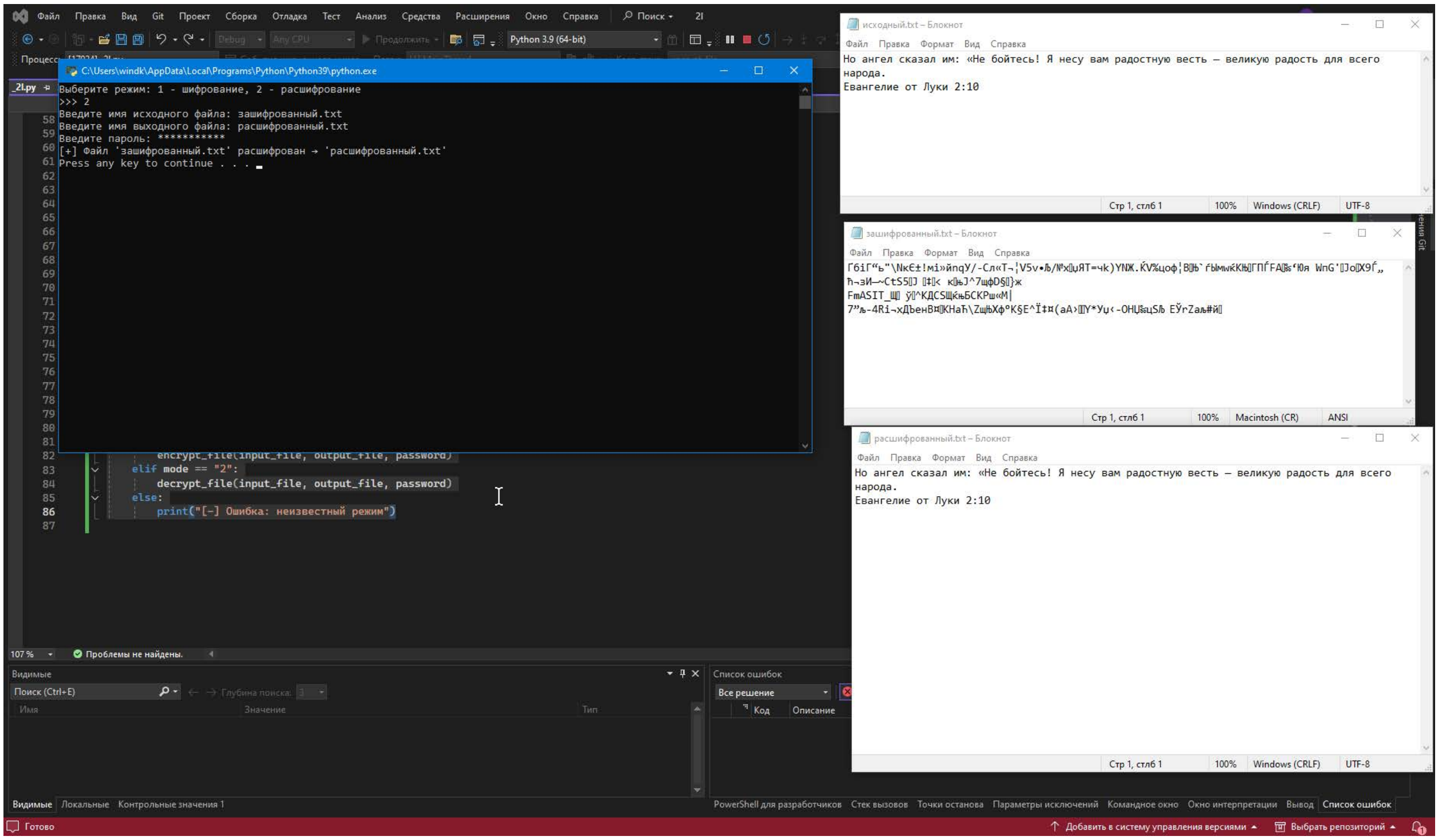
Примеры демонстрации работы программы



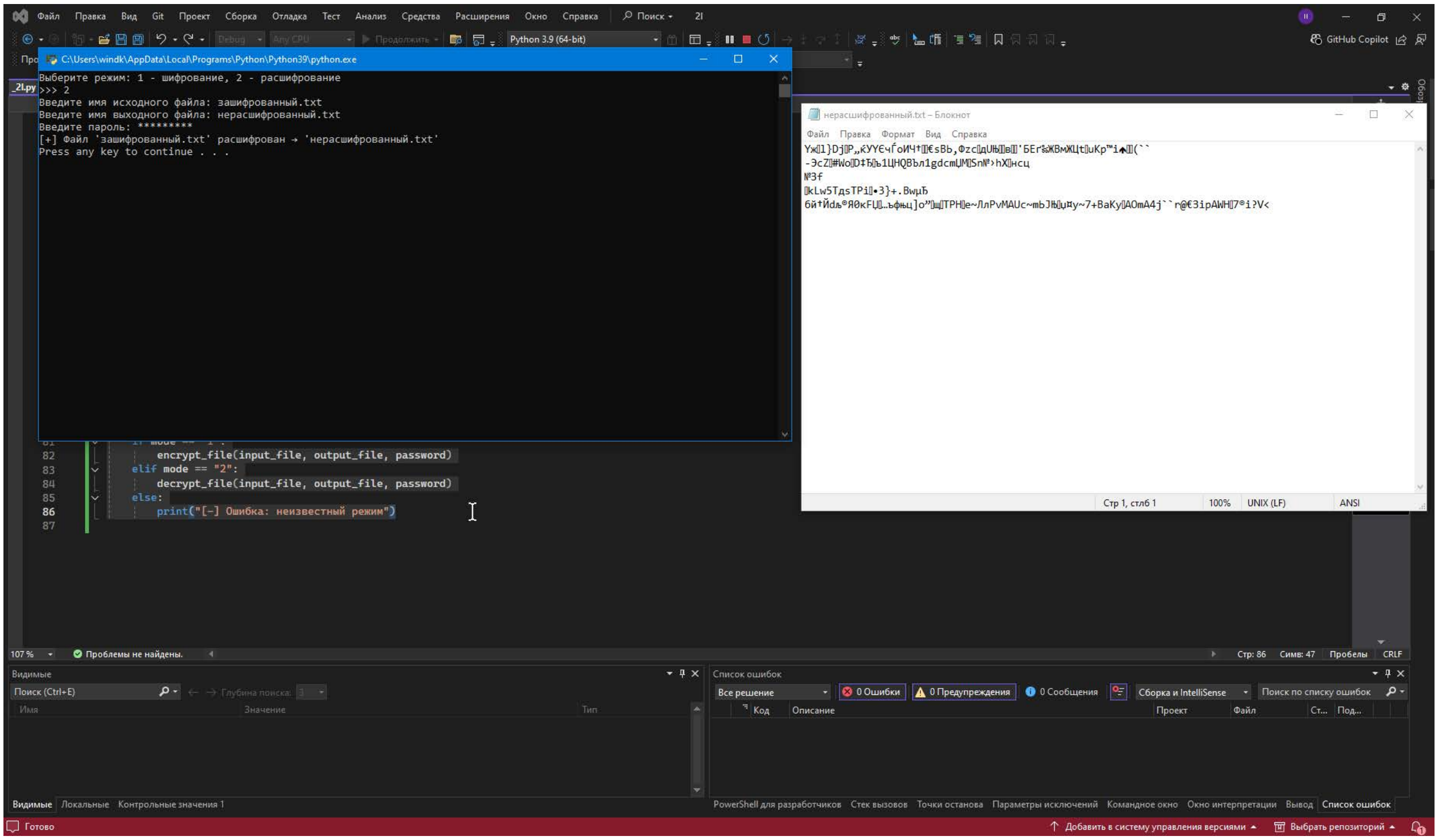
Исходный текст в файле исходный.txt



В программе выбирается режим 1 (шифрование), вводится имя файла с шифруемым текстом и вводится пароль. На выходе получается текстовый файл с зашифрованным текстом зашифрованный.txt



В программе выбирается режим 2 (дешифровка), выбирается файл с зашифрованным текстом зашифрованный.txt и указывается имя выходного файла, в который будет записан расшифрованный текст расшифрованный.txt, затем указывается тот же пароль, что и при шифровании. Результат дешифровки соответствует исходному тексту.



Если на этапе дешифровки будет указан неверный пароль, то текст не будет расшифрован.