Ковкель Никита, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № 7.

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров.

**1.Практическая часть**

Основной код представляет последовательное выполнение зашифрования и расшифрования сообщения в соответствии с введёнными значениями, а также подсчёт затраченного времени на выполнение этих функций.

|  |
| --- |
| def timestart():  def encrypt\_ede2():  start\_time = time.time()  global previous\_ciphertext  plaintext = plaintext\_entry.get("1.0", "end-1c")  if plaintext:  cipher = pyDes.triple\_des(key1 + key2 + key1, pyDes.ECB, pad=None, padmode=pyDes.PAD\_PKCS5)  ciphertext = cipher.encrypt(plaintext.encode())  ciphertext\_display.delete("1.0", "end")  ciphertext\_display.insert("end", ciphertext.hex())  changed\_bits = get\_avalanche\_effect(plaintext.encode(), ciphertext)  avalanche\_label.config(text=f"Подсчёт количества изменяющихся символов: {changed\_bits}")    end\_time = time.time()  execution\_time = end\_time - start\_time  return execution\_time  execution\_time = encrypt\_ede2()  time\_label.config(text=f"Подсчёт времени: {execution\_time:.6f} секунд")  def timeend():  def decrypt\_ede2():  start\_time = time.time()  ciphertext = ciphertext\_display.get("1.0", "end-1c")  if ciphertext:  cipher = pyDes.triple\_des(key1 + key2 + key1, pyDes.ECB, pad=None, padmode=pyDes.PAD\_PKCS5)  plaintext = cipher.decrypt(bytes.fromhex(ciphertext))  plaintext\_entry.delete("1.0", "end")  plaintext\_entry.insert("end", plaintext.decode())  end\_time = time.time()  execution\_time = end\_time - start\_time  return execution\_time  execution\_time = decrypt\_ede2()  time\_label.config(text=f"Время на расшифровку: {execution\_time:.6f} секунд") |

Листинг 1.1 – Основной код приложения

Данный код представляет собой две функции: timestart и timeend. Они выполняют операции шифрования и дешифрования с использованием алгоритма Triple DES (3DES) и замеряют время выполнения этих операций.

Функция timestart отвечает за шифрование текста. В ней определена внутренняя функция encrypt\_ede2, которая выполняет шифрование текста. В начале функции извлекается введенный текст из элемента интерфейса plaintext\_entry. Затем создается объект шифрования cipher с использованием ключа key1 + key2 + key1 и режима ECB (электронная кодовая книга) с указанием режима заполнения PKCS5. Далее, текст шифруется с помощью метода encrypt и вставляется в элемент интерфейса ciphertext\_display. Затем вызывается функция get\_avalanche\_effect, которая подсчитывает количество изменяющихся символов (локализацию лавинного эффекта) между исходным текстом и зашифрованным текстом. Результат выводится на элемент интерфейса avalanche\_label.

Функция timeend отвечает за дешифрование текста. В ней определена внутренняя функция decrypt\_ede2, которая выполняет дешифрование текста. В начале функции извлекается зашифрованный текст из элемента интерфейса ciphertext\_display. Затем создается объект шифрования cipher с использованием ключа key1 + key2 + key1 и режима ECB с указанием режима заполнения PKCS5. Далее, текст дешифруется с помощью метода decrypt и вставляется в элемент интерфейса plaintext\_entry.

**Слабые ключи:**

Слабые ключи — это одни из тех, которые после операции удаления проверочных бит состоят из всех нулей или всех единиц, или половины нулей и половины единиц.

Опасность использования слабых ключей заключается в том, что, если мы зашифровали блок слабым ключом и впоследствии расшифровали результат тем же самым слабым ключом, мы получаем первоначальный блок. Процесс создает один и тот же первоначальный блок, если мы расшифровываем блок дважды.

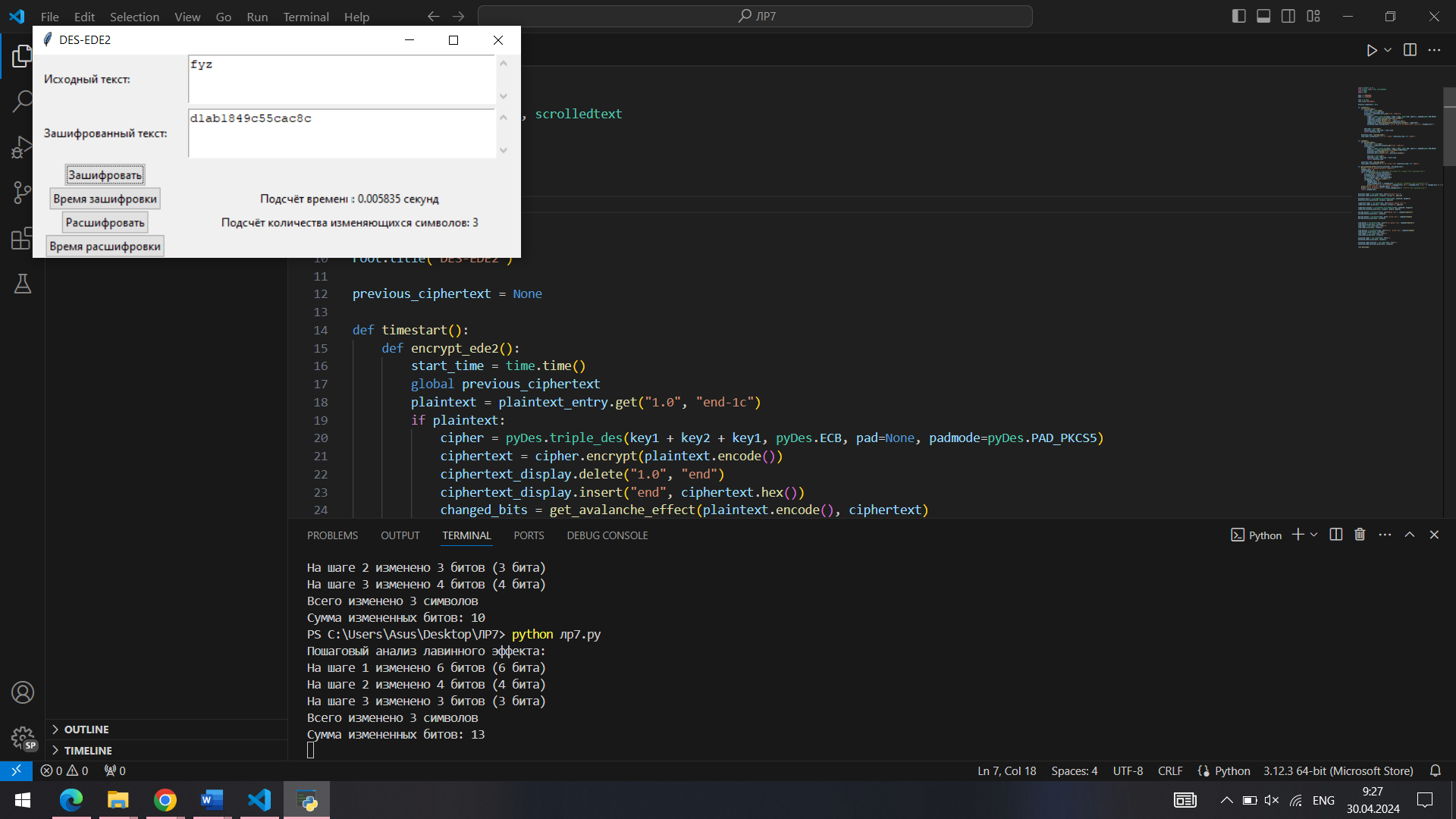


Рисунок 1.1 – Результат использования слабых ключей

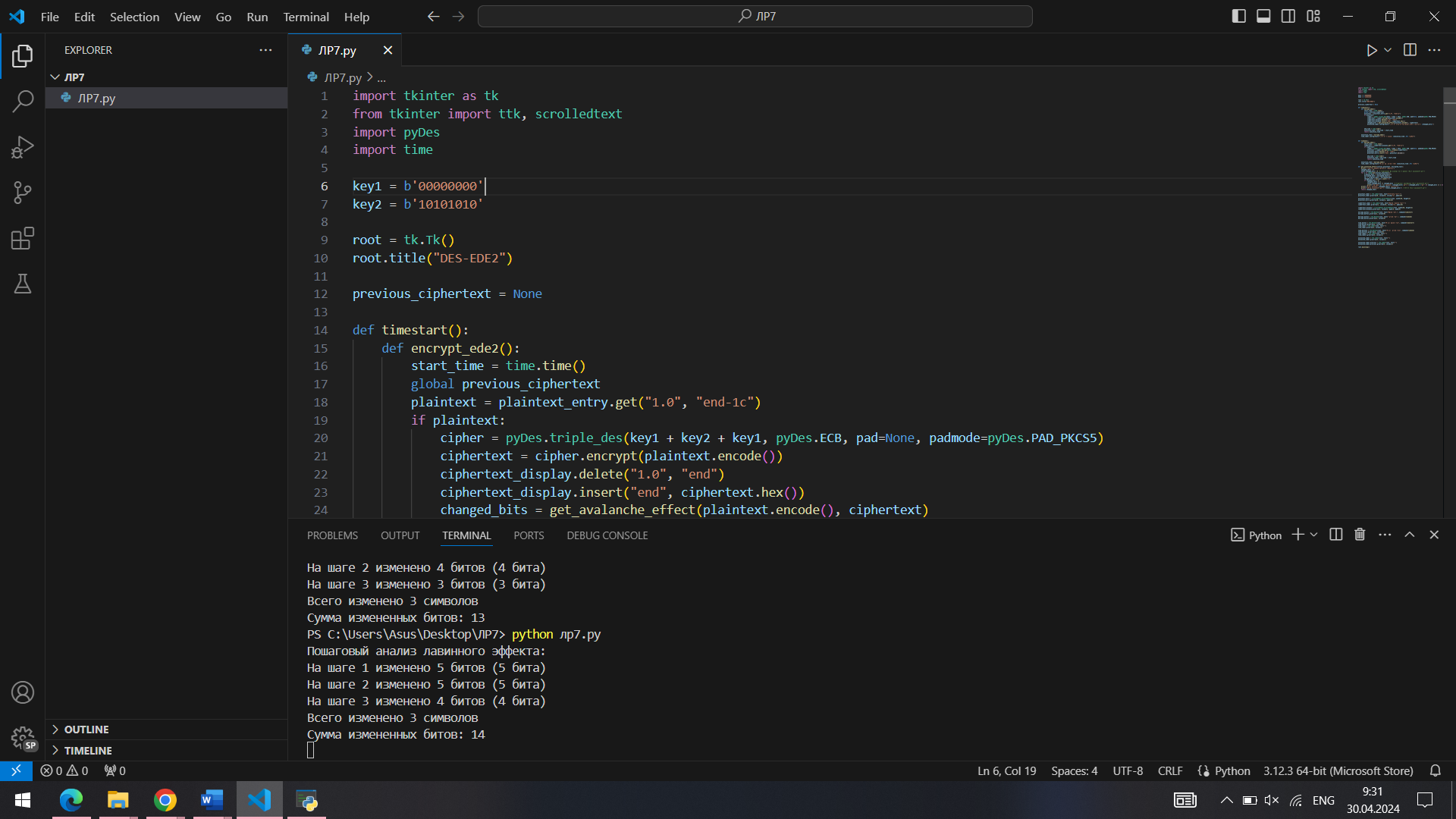


Рисунок 1.2 – Слабые ключи

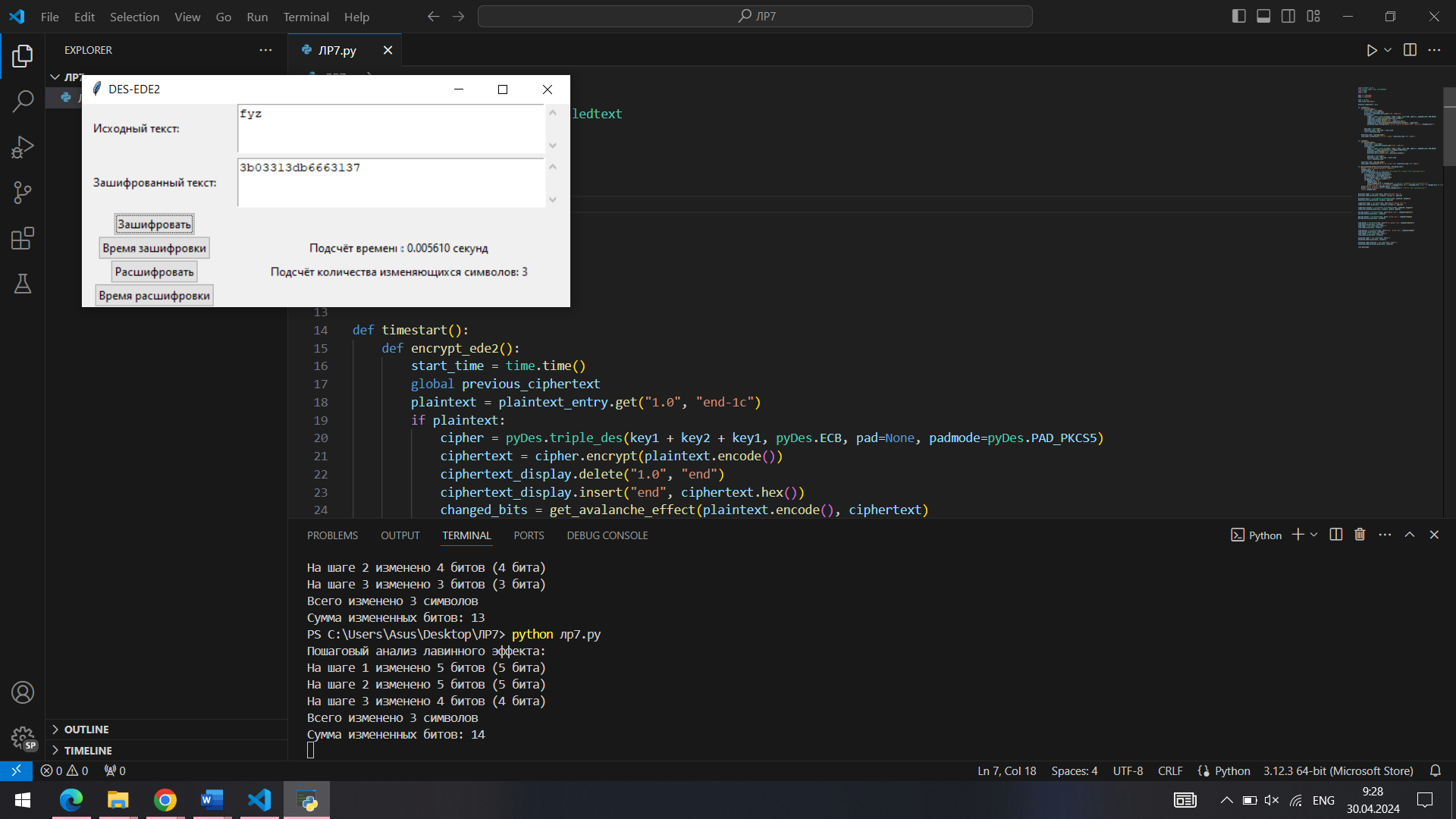


Рисунок 1.3 – Результат использования обычных ключей

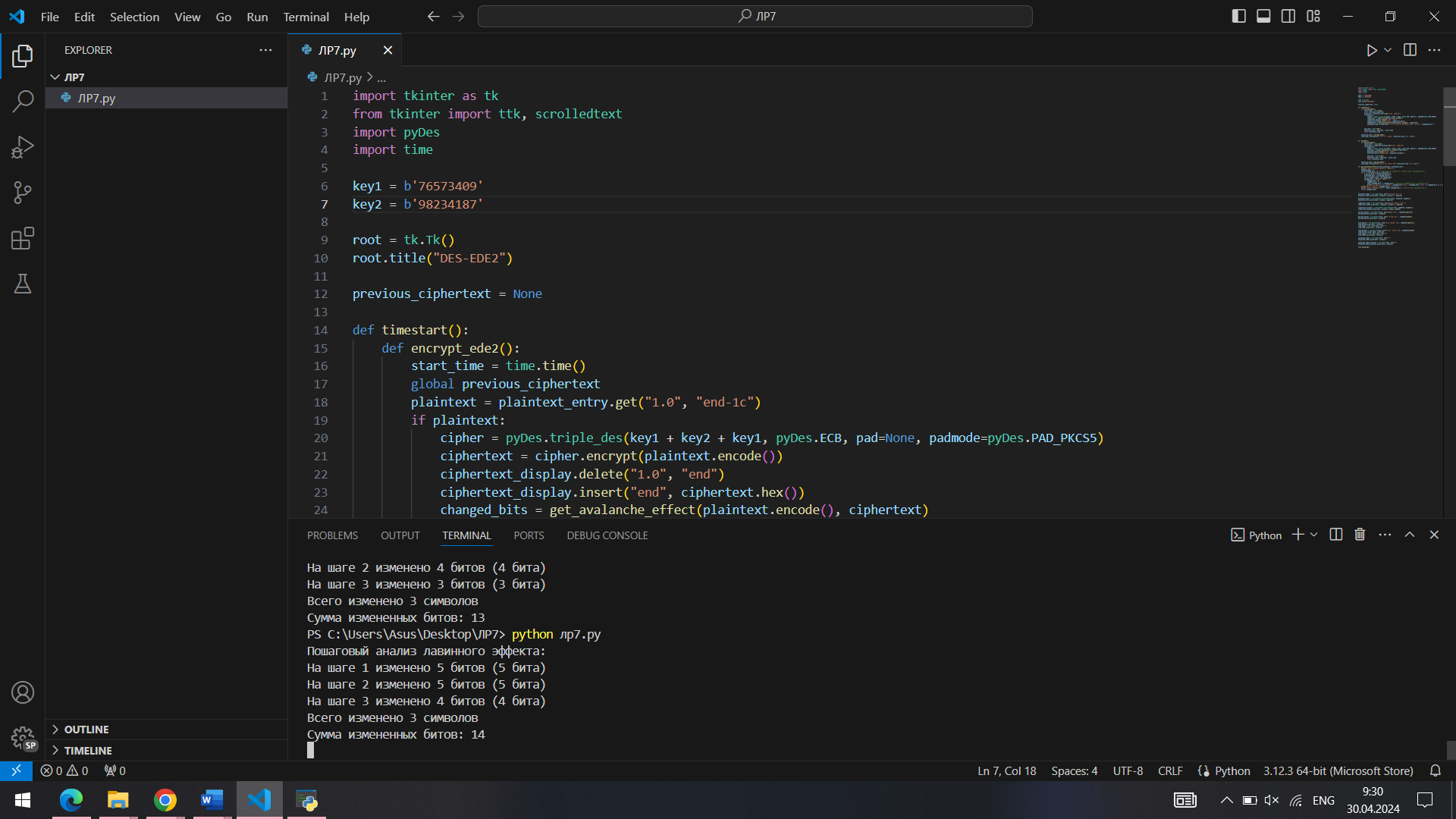


Рисунок 1.4 – Используемые ключи

**Время выполнения:**

За измерение времени выполнения отвечает блок кода, начинающийся с start\_time = time.time(). Он фиксирует текущее время перед выполнением шифрования, а затем после выполнения шифрования вычисляет разницу времени end\_time - start\_time. Результат выводится на элемент интерфейса time\_label.

|  |
| --- |
| def timestart():  def encrypt\_ede2():  start\_time = time.time()  …  end\_time = time.time()  execution\_time = end\_time - start\_time  return execution\_time  execution\_time = encrypt\_ede2()  time\_label.config(text=f"Подсчёт времени: {execution\_time:.6f} секунд")  def timeend():  def decrypt\_ede2():  start\_time = time.time()  …  end\_time = time.time()  execution\_time = end\_time - start\_time  return execution\_time  execution\_time = decrypt\_ede2()  time\_label.config(text=f"Время на расшифровку: {execution\_time:.6f} секунд") |

Листинг 1.1 – Основной код приложения

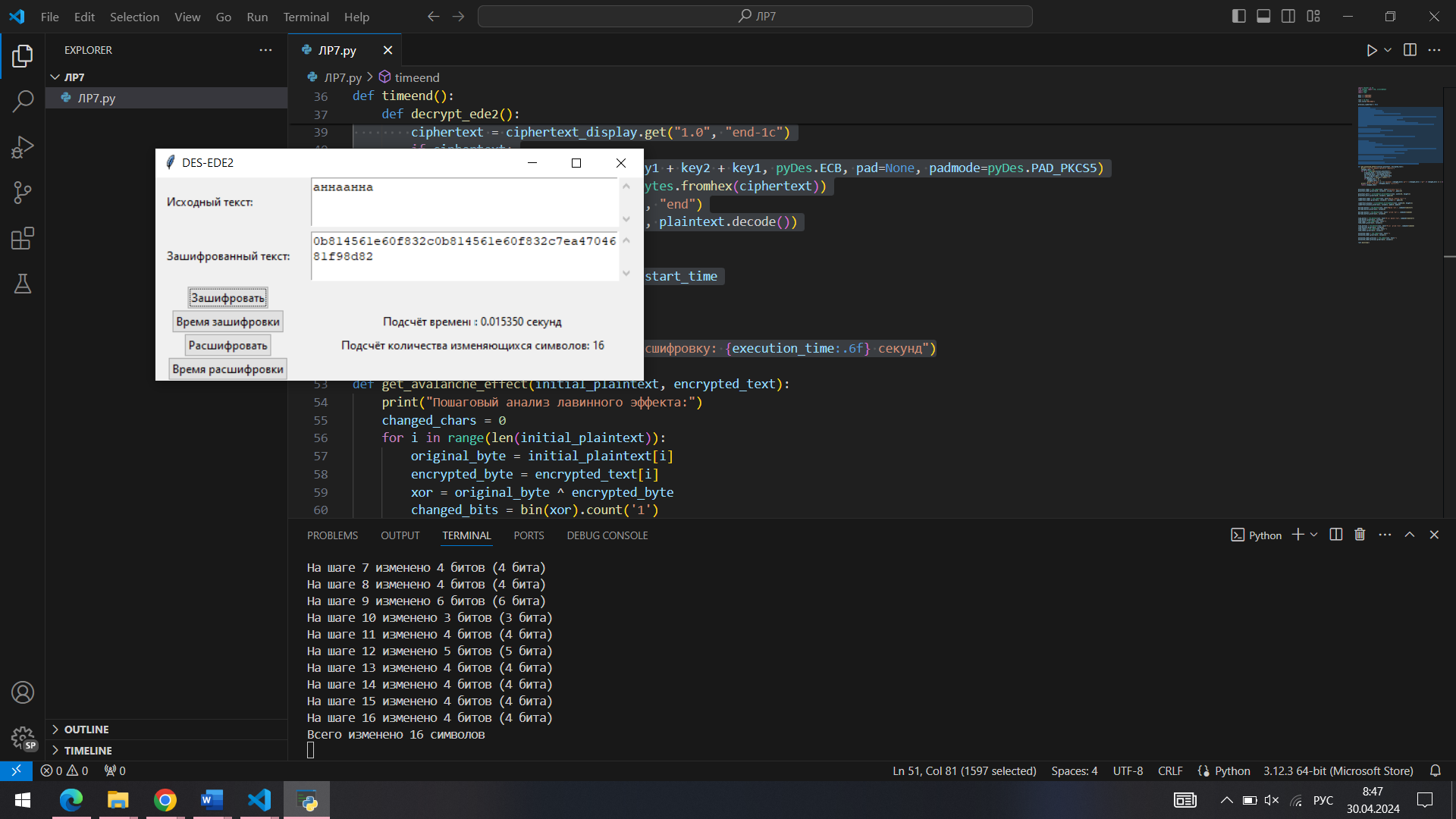


Рисунок 1.5 – Подсчёт времени

**Вывод:** в данной лабораторной работе мы изучили и приобрели практические навыки разработки и использования приложений для реализации блочных шифров.