Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №10**

**Дисциплина: Криптографические протоколы**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Цель работы:** реализовать рюкзачную криптосистему шифрования и расшифрования вводимых сообщений. Открытый ключ показывать пользователю, закрытый ключ записывать в файл.

**Ход работы:**

Рюкзачная криптосистема (или шифр с открытым ключом) — это класс криптографических алгоритмов, которые используют два ключа: публичный и приватный. Рассмотрим основные концепции таких алгоритмов:

1. **Публичный ключ**: известен всем и используется для шифрования сообщений. Он может быть распространен открыто, безопасно передаваться и храниться в открытом доступе.
2. **Приватный ключ**: известен только владельцу и используется для расшифровки зашифрованных сообщений. Он должен быть строго секретным и храниться в надежном месте.

Процесс работы рюкзачной криптосистемы:

* **Генерация ключей**:
  + Пользователь генерирует пару ключей: публичный и приватный.
  + Публичный ключ передается другим пользователям.
* **Шифрование**:
  + Пользователь, желающий отправить зашифрованное сообщение, использует публичный ключ получателя для шифрования.
  + Получатель использует свой приватный ключ для расшифровки сообщения.

**Пример использования:** Алиса генерирует ключи: публичный и приватный.Боб передает Алисе свой публичный ключ.Алиса шифрует сообщение с помощью публичного ключа Боба.Боб расшифровывает сообщение с помощью своего приватного ключа.

Важно отметить, что рюкзачные криптосистемы используют математические задачи, такие как задача о рюкзаке, для обеспечения безопасности. Эти системы широко применяются в современной криптографии, включая протоколы HTTPS, SSH и другие. Однако в реальных системах безопасности используются более сложные алгоритмы, такие как RSA и ECC.

Программный продукт состоит из 3 классов, каждый из которых отвечает за определенную часть алгоритма.

Класс **KeyPair** реализует генерацию пары ключей: публичного и приватного. Методы класса:

1. **\_\_init\_\_(self, n=128)**. Это **конструктор** класса, который вызывается при создании объекта типа KeyPair. Принимает необязательный аргумент n, который по умолчанию равен 128. Генерирует публичный и приватный ключи, используя метод Generate(n). Присваивает полученные ключи атрибутам public\_key и private\_key объекта.
2. **Generate(n)** (статический метод). Этот метод **генерирует ключи**. Принимает аргумент n, который определяет длину последовательности случайных чисел. Создает случайную последовательность из n чисел, каждое из которых выбирается случайно в диапазоне от 1 до 1000. Вычисляет публичный ключ как список, в котором каждый элемент равен 2 в степени соответствующего числа из последовательности. Приватный ключ вычисляется как сумма всех элементов публичного ключа. Возвращает пару ключей: публичный и приватный.
3. **GetPublicKey(self)**. Этот метод возвращает публичный ключ объекта KeyPair.
4. **GetPrivateKey(self)**. Этот метод возвращает приватный ключ объекта KeyPair.

Класс KeyPair представлен на рисунке 1.

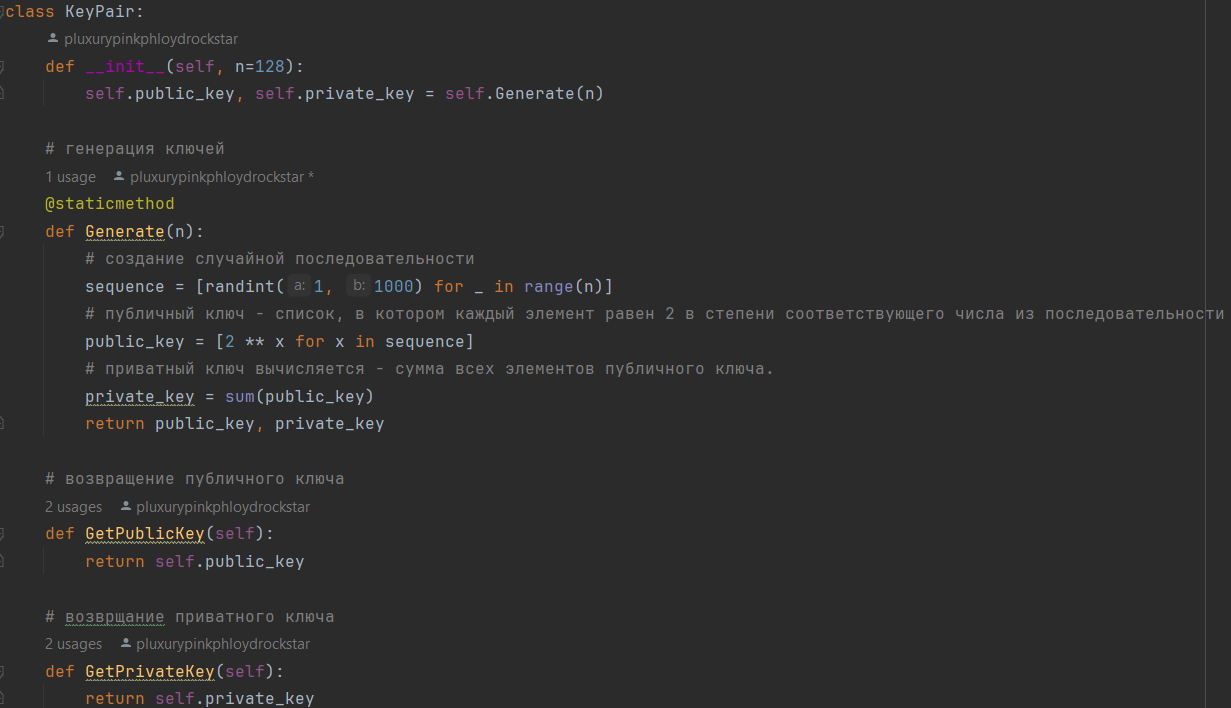


Рисунок 1 – Класс генерации ключей.

Класс **CipherText** реализует шифрование сообщений с использованием публичного ключа. Методы класса:

1. **\_\_init\_\_(self, message, public\_key)**. Это **конструктор** класса, который вызывается при создании объекта типа CipherText. Принимает два аргумента: message (сообщение, которое нужно зашифровать) и public\_key (публичный ключ). Вызывает метод Encrypt(message, public\_key) для шифрования сообщения. Присваивает полученный зашифрованный текст атрибуту cipher\_text объекта.
2. **Encrypt(message, public\_key)** (статический метод). Этот метод **шифрует сообщение**. Принимает аргументы message (сообщение) и public\_key (публичный ключ). Создает пустой список cipher\_text. Для каждого символа в сообщении: вычисляет код символа с помощью функции ord(char), умножает код символа на элемент публичного ключа, выбирая его по индексу len(cipher\_text) % len(public\_key) и добавляет полученное значение в список cipher\_text. Возвращает список cipher\_text, содержащий зашифрованный текст.
3. **Get(self)**. Этот метод возвращает зашифрованный текст, хранящийся в атрибуте cipher\_text.

Класс CipherText представлен на рисунке 2.

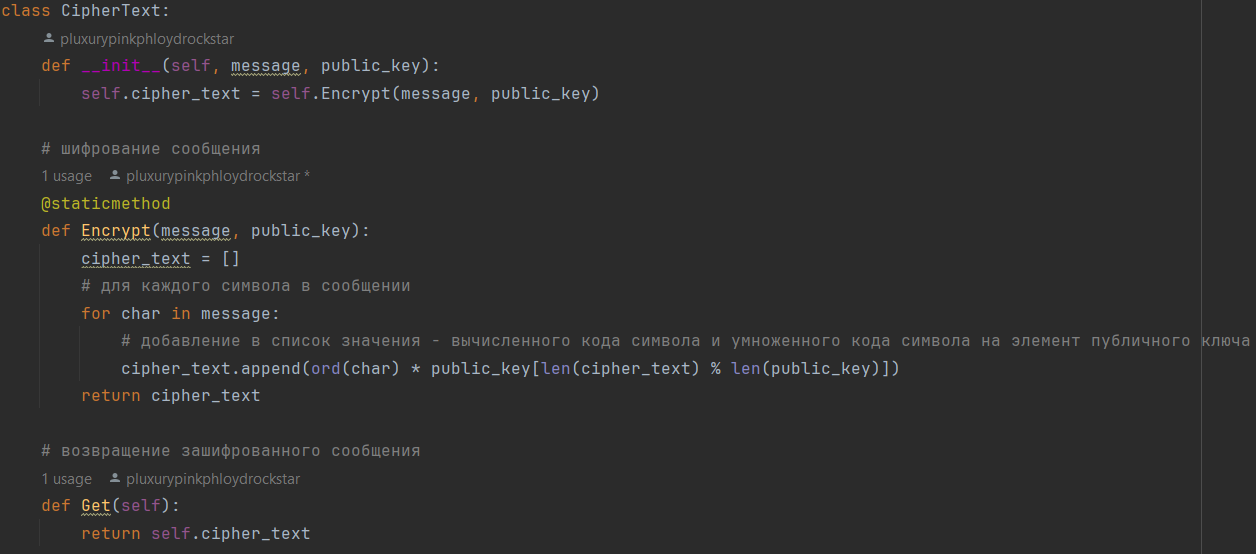


Рисунок 2 – Класс шифрования сообщения.

Финальный класс **PrivateKey** реализует расшифровку зашифрованного текста с использованием приватного ключа. Методы класса:

1. **\_\_init\_\_(self, private\_key)**. Это **конструктор** класса, который вызывается при создании объекта типа PrivateKey. Принимает аргумент private\_key (приватный ключ). Присваивает полученный приватный ключ атрибуту private\_key объекта.
2. **DecipherString(self, cipher\_text)**. Этот метод **расшифровывает зашифрованный текст**. Принимает аргумент cipher\_text (зашифрованный текст). Создает пустую строку original\_message. Для каждого зашифрованного символа в cipher\_text: вычисляет код символа, разделив зашифрованный символ на приватный ключ (используется операция модуля), преобразует полученный код символа в символ с помощью функции chr(char\_code) и добавляет символ к original\_message. Возвращает расшифрованное сообщение.

Класс PrivateKey представлен на рисунке 3.

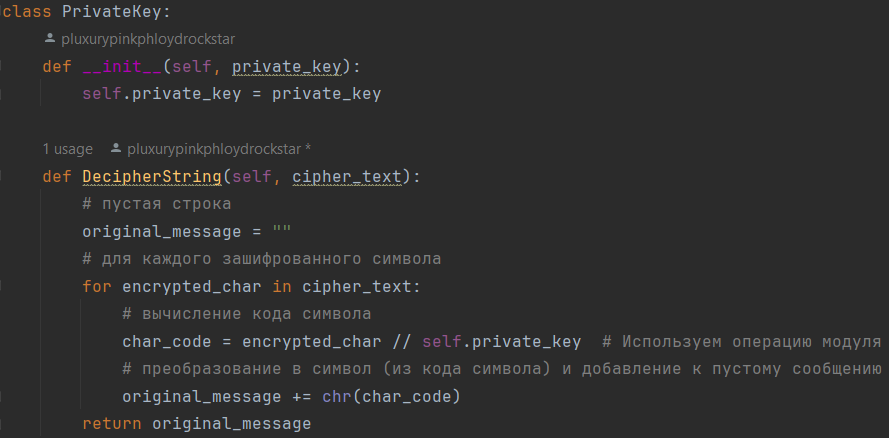


Рисунок 3 – Класс расшифровки зашифрованного сообщения.

В конце программы реализовано следующее: функция SavePrivateKeyToFile() сохраняет приватный ключ в файл, а функция SaveDecryptedMessageToFile() охраняет расшифрованное сообщение в файл. Создается объект KeyPair, который генерирует пару ключей: публичный и приватный. Выводится на экран публичный ключ. Затем создается сообщение “Hello, World!” и шифруется с использованием публичного ключа. Расшифровывает зашифрованное сообщение с помощью приватного ключа. Сохраняется приватный ключ в файл “private\_key.txt” и сохраняется расшифрованное сообщение в файл “decrypted\_message.txt” (рисунок 4).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Вызов соответствующих функций.

**Результаты работы программы:**

На рисунке 5 представлен результат работы вышеописанной программы.

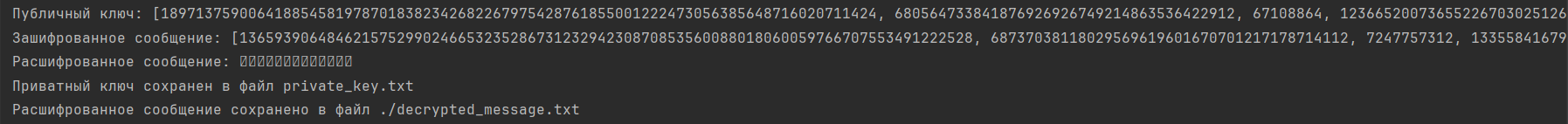


Рисунок 5 – Результат работы программы.

**Листинг программы**

Файл LW\_10.py

# реализация рюкзачной криптосистемы

from random import randint

# класс генерации публичного и приватного ключа

class KeyPair:

def \_\_init\_\_(self, n=128):

self.public\_key, self.private\_key = self.Generate(n)

# генерация ключей

@staticmethod

def Generate(n):

# создание случайной последовательности

sequence = [randint(1, 1000) for \_ in range(n)]

# публичный ключ - список, в котором каждый элемент равен 2 в степени соответствующего числа из последовательности

public\_key = [2 \*\* x for x in sequence]

# приватный ключ вычисляется - сумма всех элементов публичного ключа.

private\_key = sum(public\_key)

return public\_key, private\_key

# возвращение публичного ключа

def GetPublicKey(self):

return self.public\_key

# возврщание приватного ключа

def GetPrivateKey(self):

return self.private\_key

# класс шифрования сообщения

class CipherText:

def \_\_init\_\_(self, message, public\_key):

self.cipher\_text = self.Encrypt(message, public\_key)

# шифрование сообщения

@staticmethod

def Encrypt(message, public\_key):

cipher\_text = []

# для каждого символа в сообщении

for char in message:

# добавление в список значения - вычисленного кода символа и умноженного кода символа на элемент публичного ключа

cipher\_text.append(ord(char) \* public\_key[len(cipher\_text) % len(public\_key)])

return cipher\_text

# возвращение зашифрованного сообщения

def Get(self):

return self.cipher\_text

# класс расшифровки зашифрованного сообщения с использованием приватного ключа

class PrivateKey:

def \_\_init\_\_(self, private\_key):

self.private\_key = private\_key

def DecipherString(self, cipher\_text):

# пустая строка

original\_message = ""

# для каждого зашифрованного символа

for encrypted\_char in cipher\_text:

# вычисление кода символа

char\_code = encrypted\_char // self.private\_key # Используем операцию модуля

# преобразование в символ (из кода символа) и добавление к пустому сообщению

original\_message += chr(char\_code)

return original\_message

# функция записи приватного ключа в файл

def SavePrivateKeyToFile(private\_key, filename="private\_key.txt"):

with open(filename, "w") as file:

file.write(str(private\_key))

print(f"Приватный ключ сохранен в файл {filename}")

# функция записи расшифрованного сообщения в файл

def SaveDecryptedMessageToFile(decrypted\_message, filename="./decrypted\_message.txt"):

with open(filename, "w") as file:

file.write(message)

print(f"Расшифрованное сообщение сохранено в файл {filename}")

# генерация ключей

key\_pair = KeyPair()

print("Публичный ключ:", key\_pair.GetPublicKey())

# сообщение

message = "Hello, World!"

# шифрование сообщения

cipher\_text = CipherText(message, key\_pair.GetPublicKey()).Get()

private\_key = PrivateKey(key\_pair.GetPrivateKey())

# расшифрование сообщения

decrypted\_message = private\_key.DecipherString(cipher\_text)

print("Зашифрованное сообщение:", cipher\_text)

print("Расшифрованное сообщение:", decrypted\_message)

# запись в файл приватного ключа и расшифрованного сообщения

SavePrivateKeyToFile(key\_pair.GetPrivateKey())

SaveDecryptedMessageToFile(decrypted\_message)