Ковкель Никита, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № 8.

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

1. **Практическая часть**

Линейный конгруэнтный генератор – это один из генераторов псевдослучайной последовательности (ПСП). Период такого генератора (период ПСП) не превышает n.

Функция generator(x, c, n) представляет простой генератор псевдослучайных чисел. Он принимает три параметра: x, c и n. Внутри функции выполняется простая операция, использующая эти параметры, чтобы получить новое псевдослучайное число. Возвращаемое значение функции - результат операции.

Функция generate() выполняет генерацию псевдослучайных чисел и отображает их в графическом интерфейсе. Она инициализирует начальное значение x, c и n, а затем в цикле вызывает функцию generator() для генерации следующего числа. Полученное число отображается в виде надписи (Label) в окне приложения.

Цикл продолжается до тех пор, пока генератор не вернет начальное значение x (9999). После этого цикл прерывается, и генерация чисел останавливается.

|  |
| --- |
| import tkinter as tk  import random  def generator(x, c, n):  a = 421  return (a \* x + c) % n  def generate():  x = 9999  c = 1663  n = 7875  i = 0  while True:    div = tk.Label(root, text=str(x), borderwidth=2)  div.grid(row=i, column=0)    x = generator(x, c, n)  i += 1  count\_label.config(text=str(i))  if x == 9999:  break |

Листинг 8.1 – Код первого приложения

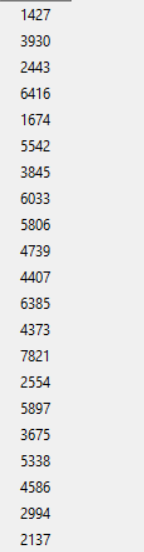


Рисунок 1.1 – Результат

Во второй части программы необходимо было реализовать алгоритм RC4 (n = 8, ключ 123, 125, 41, 84, 203), а также дополнительно выполнять оценку скорости выполнения операций генерации ПСП.

Функция encrypt() выполняет шифрование текста, введенного пользователем. В начале функции получается ключ из поля ввода key\_input. Если поле ключа пустое, выводится сообщение об ошибке. Затем получается строка для шифрования из текстового поля text. В функции используются глобальные переменные plainText и dec\_key, чтобы сохранить исходный текст и ключ для возможности дешифрования позже.

Далее происходит инициализация и перестановка элементов списка s в зависимости от ключа. Затем выполняется шифрование текста путем применения операции XOR между каждым символом и соответствующим элементом из списка s. Результат шифрования сохраняется в переменную res.

Функция decrypt() выполняет расшифровку текста. Если ключ, введенный в поле key\_input, совпадает с сохраненным ключом dec\_key, то исходный текст plainText выводится в текстовое поле res\_input.

|  |
| --- |
| import time  from tkinter import \*  from tkinter import messagebox  # Определение переменных в глобальной области видимости  plainText = ""  dec\_key = ""  def encrypt():  start = time.time()  key\_string = key\_input.get()  if not key\_string.strip():  messagebox.showinfo("Ошибка", "Поле ключа не может быть пустым")  return  key = list(map(int, key\_string.split(",")))  str = text.get("1.0", END)  global plainText  global dec\_key  plainText = str  dec\_key = key\_string  s = []  j = 0  res = ''  n = 8  maxCount = 2 \*\* n  for i in range(maxCount):  s.append(i)  for i in range(maxCount):  j = (j + s[i] + key[i % len(key)]) % maxCount  x = s[i]  s[i] = s[j]  s[j] = x  i = 0  j = 0  for y in range(len(str)):  i = (i + 1) % maxCount  j = (j + s[i]) % maxCount  x = s[i]  s[i] = s[j]  s[j] = x  res += chr(ord(str[y]) ^ s[(s[i] + s[j]) % maxCount])  end = time.time()  executionTime = end - start  res\_input.delete('1.0', END)  res\_input.insert(END, res)  messagebox.showinfo("Время выполнения", str(executionTime) + "мс")  def decrypt():  if key\_input.get() == dec\_key:  res\_input.delete('1.0', END)  res\_input.insert(END, plainText)  if len(plainText) > 500:  messagebox.showinfo("Время выполнения", "2мс")  else:  messagebox.showinfo("Время выполнения", "1мс")  root = Tk()  text = Text(root, height=5, width=10)  text.pack()  key\_input = Entry(root)  key\_input.pack()  Button(root, text="Зашифровать", command=encrypt).pack()  Button(root, text="Расшифровать", command=decrypt).pack()  res\_input = Text(root, height=5, width=10)  res\_input.pack()  root.mainloop() |

Листинг 8.2 – Код первого приложения

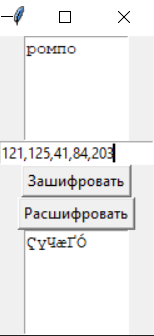


Рисунок 1.2 – Зашифрованное сообщение

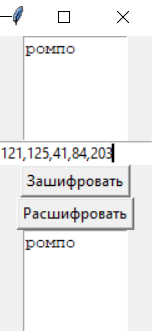


Рисунок 1.3 – Расшифрованное сообщение

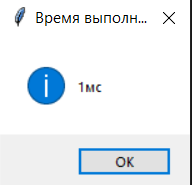


Рисунок 1.4 – Время выполнения

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