

Прагматика выполнения лабораторной работы

Студенты должны разбираться в методах шифрования и познакомиться способом шифрования текста гаммированием. Все это необходимо для глубоко погружения в среду Centos и для повышения безопасности в системе.

Цель выполнения лабораторной работы

Освоить на практике применение режима однократного гаммирования.

Задачи выполнения лабораторной работы

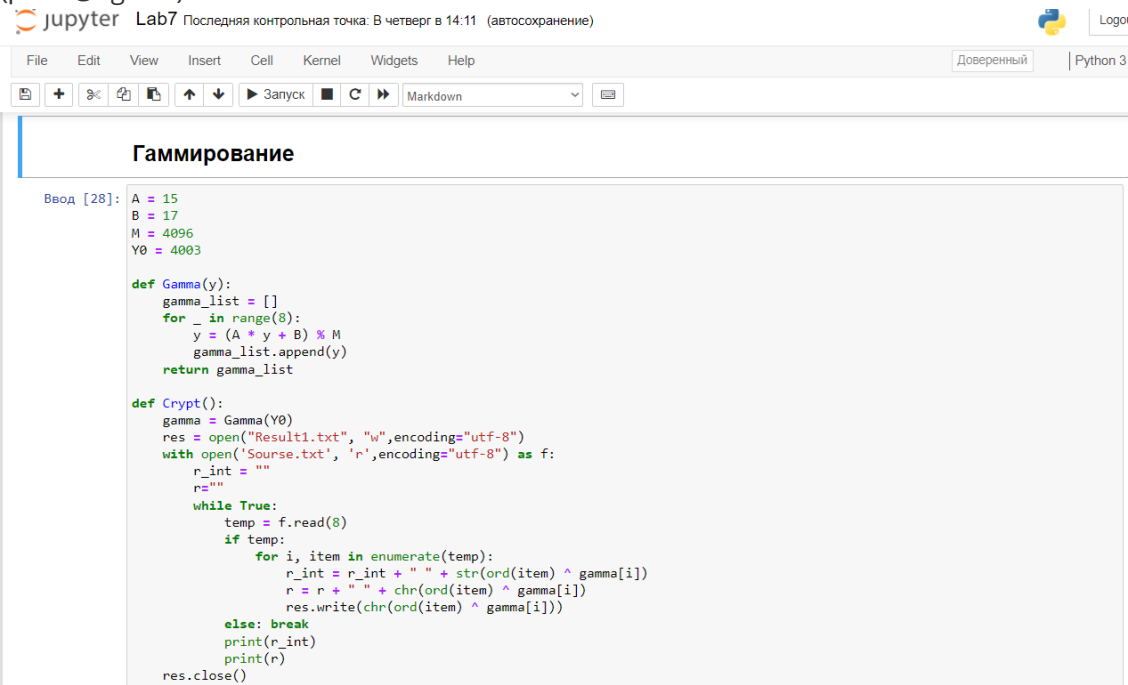
Разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

Результаты выполнения лабораторной работы

1. Разработала программу, позволяющую шифровать и дешифровать данные в режиме однократного гаммирования. Программа имеет следующий вид.

(рис. -@fig:001)



The screenshot shows a Jupyter Lab environment with a file named 'Гаммирование'. The code defines a function 'Gamma(y)' to generate a key stream and a function 'Crypt()' to perform encryption. The 'Crypt()' function reads from 'Source.txt' and writes to 'Result1.txt'.

```
Ввод [28]: A = 15
B = 17
M = 4096
Y0 = 4003

def Gamma(y):
    gamma_list = []
    for _ in range(8):
        y = (A * y + B) % M
        gamma_list.append(y)
    return gamma_list

def Crypt():
    gamma = Gamma(Y0)
    res = open("Result1.txt", "w", encoding="utf-8")
    with open('Source.txt', 'r', encoding="utf-8") as f:
        r_int = ""
        r = ""
        while True:
            temp = f.read(8)
            if temp:
                for i, item in enumerate(temp):
                    r_int = r_int + " " + str(ord(item) ^ gamma[i])
                    r = r + " " + chr(ord(item) ^ gamma[i])
                res.write(chr(ord(item) ^ gamma[i]))
            else: break
        print(r_int)
        print(r)
    res.close()
```

(рис. -@fig:002)

```
def DeCrypt():
    gamma = Gamma(Y0)
    res = open("NewResult.txt", "w", encoding="utf-8")
    with open('Result1.txt', 'r', encoding="utf-8") as f:
        r_int = ""
        r = ""
        while True:
            temp = f.read(8)
            if temp:
                for i, item in enumerate(temp):
                    r_int = r_int + " " + str(ord(item) ^ gamma[i])
                    r = r + chr(ord(item) ^ gamma[i])
                    res.write(chr(ord(item) ^ gamma[i]))
            else: break
        print(r_int)
        print(r)
    res.close()
```

Приложение написано на python 3. Я запускала его через jupyter Notebook. В данном коде имеется 3 основные функции. 1 - создает гамму, 2 - шифрует текст, 3 - расшифровывает шифротекст.

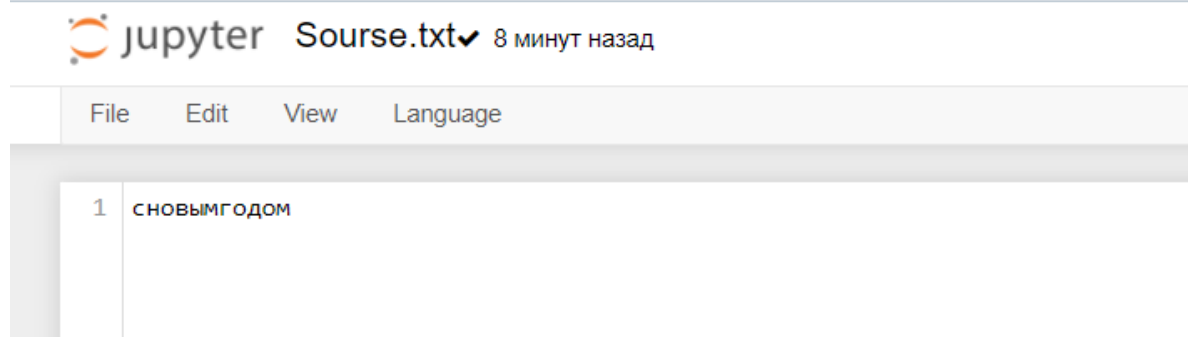
Результаты выполнения лабораторной работы

- Программа работает по следующему алгоритму. Сначала пользователь вводит свой текст, который хочет зашифровать в файл source.txt. Далее пользователь заходит в ноутбук, запускает функцию шифрование. Зашифрованный текст и гамма появляются в файле result.txt.

(рис. -@fig:003)

```
Ввод [34]: Crypt()
3807 2926 464 3377 885 2191 2749 3677
⚡ Γ ï Œ , Œ S Œ
3807 2926 464 3377 885 2191 2749 3677 3754 2925 466
⚡ Γ ï Œ , Œ S Œ 9 ö
```

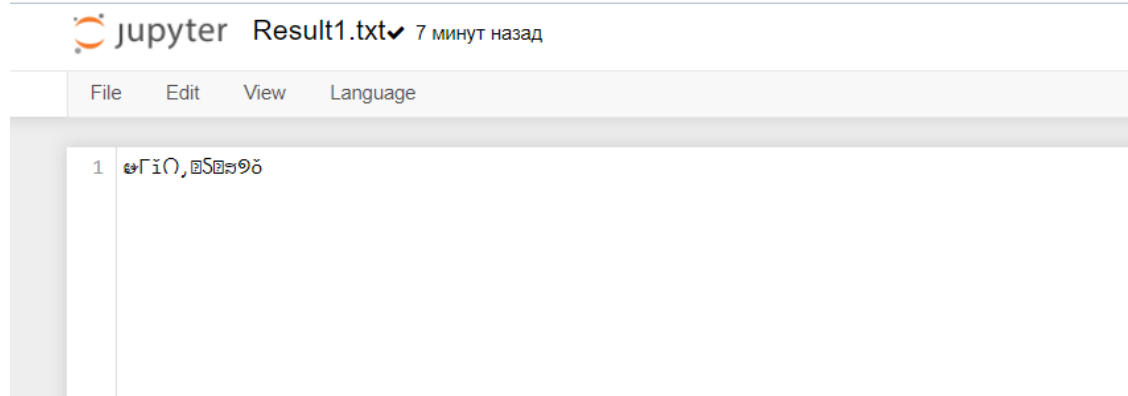
(рис. -@fig:006)



Результаты выполнения лабораторной работы

- Для расшифровки текста, пользователь запускает функцию расшифровки в ноутбуке. Функция на основе нашего шифротекста использует гамму и мы получаем исходный текст.

(рис. -@fig:005)



(рис. -@fig:004)

Ввод [35]: DeCrypt()

1089 1085 1086 1074 1099 1084 1075 1086 1076 1086 1084
с новым годом

Вывод

Освоен на практике метод однократного гаммирования. Написана программа по шифровке и дешифровке.

{.standout}Спасибо за внимание