

# **Информационная безопасность. Отчет по лабораторной работе № 3**

**Шифрование гаммированием**

Мухамеджанов Исматулло Иззатуллоевич

# Содержание

1	Цель работы	5
2	Указание к работе	6
3	Выполнение лабораторной работы	7
4	Выводы	9
5	Список литературы	10

# List of Figures

3.1 Программа (1) . . . . . 8

## List of Tables

# 1 Цель работы

Освоить на практике применение шифрование гаммирование [1].

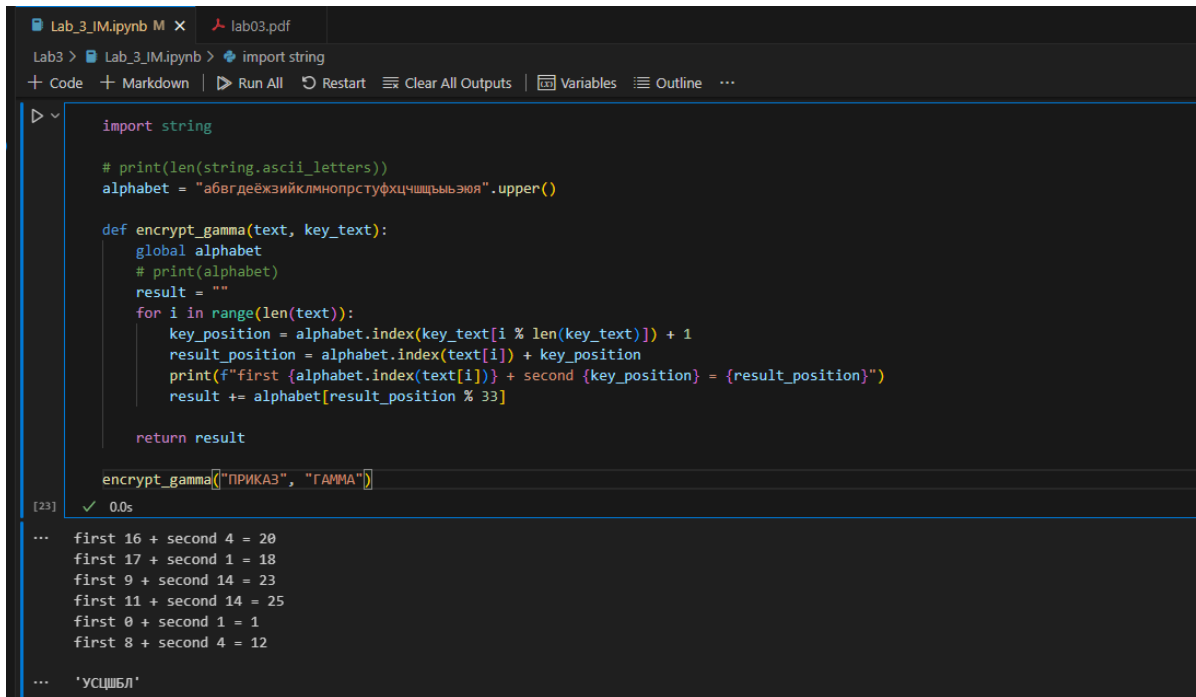
## **2 Указание к работе**

Шифрование Гаммированием

### 3 Выполнение лабораторной работы

1. Из всех схем шифрования простейшей и наиболее надежной является схема однократного использования (рис. 1). Формируется  $m$ -разрядная случайная двоичная последовательность — ключ шифра. Отправитесь производит побитовое сложение по модулю два ( $\text{mod } 2$ ) ключа  $k = k_1k_2...k_i...k_m$  и  $m$ -разрядной двоичной последовательности  $P = P_1P_2...P_i...P_m$  соответствующей посылаемому сообщению:

$c_i = p_i \boxplus k_i$ ,  $i = 1, m$ , где  $p_i$  —  $i$ -й бит исходного текста,  $k_i$  —  $i$ -й бит ключа, — операция побитового сложения (XOR),  $c$  —  $i$ -и  $m$  получившейся криптограммы  
 $c = c_1c_2 \dots c_i \dots c_m$ .



The image shows a Jupyter Notebook interface with a dark theme. The top bar displays the file name 'Lab\_3\_IM.ipynb' and a PDF viewer 'lab03.pdf'. The notebook is open to a cell containing Python code for a Caesar cipher encryption. The code imports the 'string' module, defines an alphabet string, and implements an 'encrypt\_gamma' function. The function iterates over the input text, calculates the position of each character in the alphabet, and shifts it by a key value. The output of the function is displayed in the cell's output area.

```
import string

# print(len(string.ascii_letters))
alphabet = "абвгдеёжзийклмнопрстуфхцчщъыьэя".upper()

def encrypt_gamma(text, key_text):
    global alphabet
    # print(alphabet)
    result = ""
    for i in range(len(text)):
        key_position = alphabet.index(key_text[i % len(key_text)]) + 1
        result_position = alphabet.index(text[i]) + key_position
        print(f"first {alphabet.index(text[i])} + second {key_position} = {result_position}")
        result += alphabet[result_position % 33]

    return result

encrypt_gamma("ПРИКАЗ", "ГАММА")
```

[23] ✓ 0.0s

```
... first 16 + second 4 = 20
... first 17 + second 1 = 18
... first 9 + second 14 = 23
... first 11 + second 14 = 25
... first 0 + second 1 = 1
... first 8 + second 4 = 12
... 'УСЦЩБЛ'
```

Figure 3.1: Программа (1)



## **4 Выводы**

Освоены шифры методом гаммирования

## **5 Список литературы**

1. Методические материалы курса