

Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №3

Шифрование гаммированием

Юдин Герман Станиславович 1132236901

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Шифрование гаммированием	6
3	Выводы	9
4	Список литературы	10

List of Figures

2.1	main_func	7
2.2	output	8

List of Tables

1 Цель работы

Освоить на практике шифрование гаммированием.

2 Выполнение лабораторной работы

Требуется реализовать:

1. Шифрование гаммированием

2.1 Шифрование гаммированием

Шифрование гаммированием - это метод симметричного шифрования, при котором каждый символ или байт исходного сообщения комбинируется с соответствующим символом или байтом ключа (гаммы) с помощью определенной операции, чаще всего XOR.

Основные шаги:

1. Выбор гаммы (ключа): Гамма — это последовательность, которая комбинируется с исходным сообщением. Гамма может быть случайной или генерироваться на основе ключа.
2. Применение гаммы к сообщению: Гамма “наложится” на исходное сообщение. Если гамма короче сообщения, она циклически повторяется.
3. Комбинирование гаммы и сообщения: Наиболее популярная операция для этого — XOR. Если мы говорим о символьном шифровании, то комбинирование может включать в себя сложение (или вычитание для дешифрования) позиций символов в алфавите.

4. Дешифрование: Чтобы дешифровать сообщение, мы применяем ту же операцию комбинирования к зашифрованному сообщению и той же гамме. Если использовалась операция XOR, то повторное применение XOR с той же гаммой вернёт исходное сообщение.

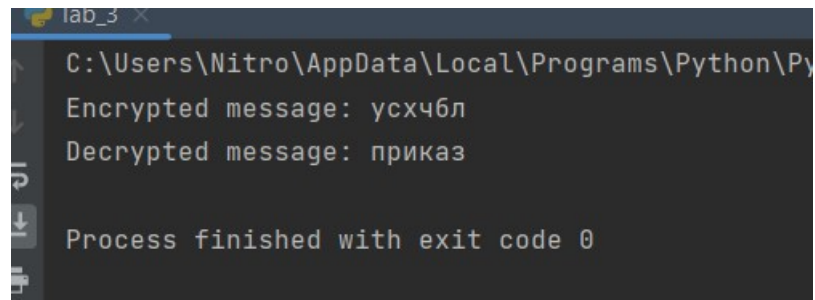
Чтобы реализовать программу был написан след. код на python:

1. Функции получения пар значений ключа и сообщения
2. Функция шифрования, которая берет пары значений и складывает их место в алфавите получая нужную букву шифрования.

```
1 def generate_gamma(gamma, message):
2     for i in range(len(message)):
3         yield gamma[i % len(gamma)], message[i]
4
5
6 def encrypt(gamma, message):
7     encrypted_message = ""
8     for g, m in generate_gamma(gamma, message):
9         encrypted_message += rus_alp[(rus_alp.index(m) + rus_alp.index(g) + 1) % len(rus_alp)]
10    return encrypted_message
11
12 def decrypt(gamma, encrypted_message):
13     decrypted_message = ""
14     for g, m in generate_gamma(gamma, encrypted_message):
15         decrypted_message += rus_alp[(rus_alp.index(m) - rus_alp.index(g) - 1) % len(rus_alp)]
16    return decrypted_message
17
18 rus_alp = "абвгдежзийклмнопрстуфхцчшщъыьэюя"
19 gamma = "гамма"
20 message = "приказ"
21
22 encrypted_message = encrypt(gamma, message)
23 print(f"Encrypted message: {encrypted_message}")
24
25 decrypted_message = decrypt(gamma, encrypted_message)
26 print(f"Decrypted message: {decrypted_message}")
```

Figure 2.1: main_func

Выходные значения программы (пример из методички).

A screenshot of a terminal window with a dark background. The window title bar at the top shows 'lab_3' followed by a close button icon. The terminal displays the following text: the first line is a file path 'C:\Users\Nitro\AppData\Local\Programs\Python\Py' (partially visible); the second line is 'Encrypted message: уcxчбл'; the third line is 'Decrypted message: приказ'; and the fourth line is 'Process finished with exit code 0'. On the left side of the terminal, there is a vertical toolbar with icons for back, forward, search, and other navigation functions.

```
lab_3 x
C:\Users\Nitro\AppData\Local\Programs\Python\Py
Encrypted message: уcxчбл
Decrypted message: приказ
Process finished with exit code 0
```

Figure 2.2: output

3 Выводы

В результате выполнения работы я освоил на практике применение шифрование гаммированием.

4 Список литературы

1. Методические материалы курса