Отчёт по лабораторной работе №3. Шифрование гаммированием

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Аронова Юлия Вадимовна, 1032212303
Группа: НФИмд-01-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

25 ноября, 2021, Москва

Цели и задачи работы

Целью данной лабораторной работы является ознакомление с методом *шифрования гаммированием*, а также его последующая программная реализация для случая конечной гаммы.

Задачи:

1. Рассмотреть алгоритм шифрования гаммированием;
2. Реализовать его для случая конечной гаммы на языке программирования Python.

Теоретическое введение

Шифры гаммирования

Шифры гаммирования (или *аддитивные шифры*) осуществляют шифрование путем сложения символов исходного текста P_i и ключа K_i по модулю, равному числу букв в алфавите (N).

Table 1: Наложение гаммы путём сложения по модулю

Модуль	Шифрование	Дешифровка
N	$C_i = (P_i + K_i) \% N$	$P_i = (C_i + N - K_i) \% N$
2	$C_i = P_i \oplus K_i$	$P_i = C_i \oplus K_i$

Здесь C_i – i-ый символ криптограммы, $\%$ – взятие остатка.

Стойкость шифра

Стойкость аддитивных шифров определяется качеством гаммы, которое зависит от длины периода и случайности распределения по периоду. Для обеспечения абсолютной стойкости необходимо, чтобы:

- последовательность символов в пределах периода гаммы была случайной;
- символы алфавита гаммы были распределены равновероятно;
- гамма совпадала по размеру или была больше исходного открытого текста;
- гамма применялась только один раз.

Генерация гамм

Так, могут использоваться или *истинно случайные гаммы*, или *псевдослучайные гаммы* – последовательности чисел, вычисленные по определённой процедуре, но имеющие все свойства случайной последовательности чисел в рамках решаемой задачи.

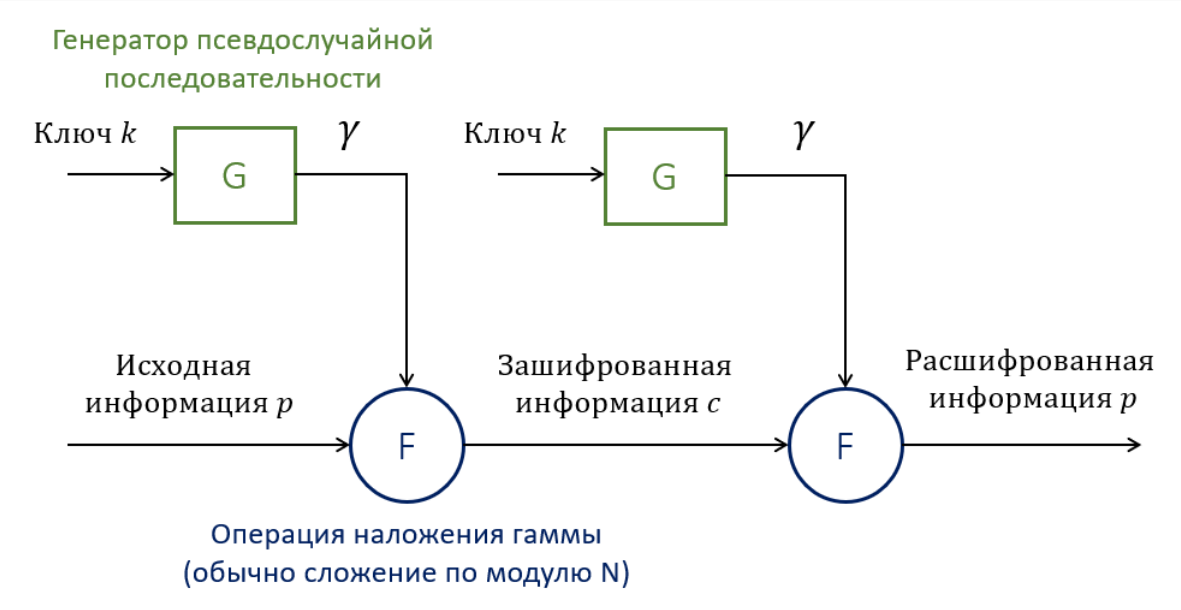

Figure 1: Схема гаммирования с использованием генератора псевдослучайных чисел

Ход выполнения и результаты

Реализация (1/2)

```
abc_rus = [chr(code) for code in range(ord('а'), ord('я') + 1)]
abc_eng = [chr(code) for code in range(ord('a'), ord('z') + 1)]

letter2number_rus = {abc_rus[i] : i for i in range(len(abc_rus))}
letter2number_eng = {abc_eng[i] : i for i in range(len(abc_eng))}

abc = {"rus" : abc_rus, "eng" : abc_eng}

letter2number = {"rus" : letter2number_rus, "eng" : letter2number_eng}
```

Реализация (2/2)

```
def gamma_cipher(message, key, language):
    mes = message.lower() # приводим сообщение к нижнему регистру
    n = len(abc[language]) # размерность алфавита

    gamma = key.lower() # приводим гамму к нижнему регистру
    while len(gamma) < len(mes): # пока она короче сообщения..
        gamma += gamma[len(gamma) - len(key)] # дополняем её повторениями

    message_encrypted = "" # криптограмма
    for i in range(len(mes)): # для каждого символа в сообщении
        m = letter2number[language][mes[i]]
        g = letter2number[language][gamma[i]]
        message_encrypted += abc[language][(m + g) % n]

    return message_encrypted
```

Результаты

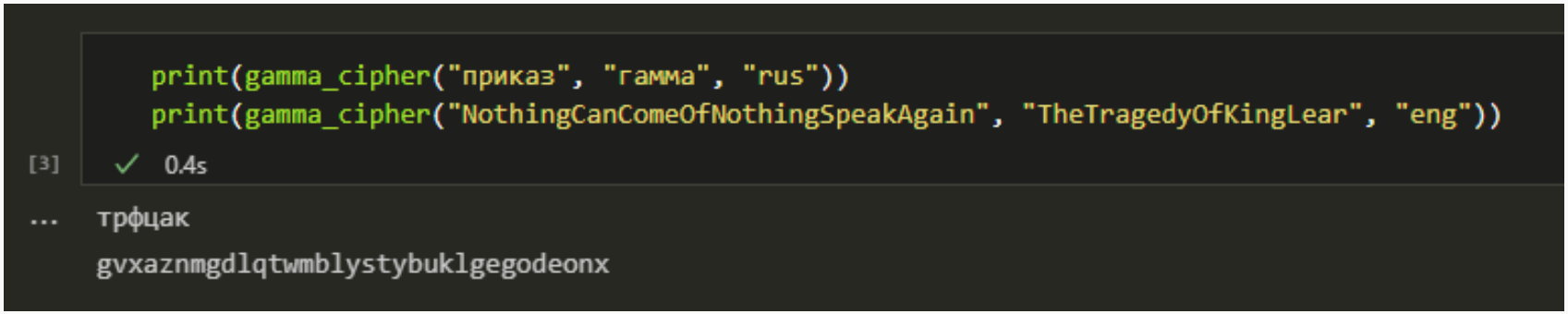

Figure 2: Пример шифрования гаммированием на основе конечной гаммы

Заключение

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с методом шифрования гаммированием, а алгоритм шифрования заданной конечной гаммой был успешно реализован на языке программирования **Python**.

Спасибо за внимание