Отчет по лабораторной работе №3

Шифрование гаммированием

Бармина Ольга Константиновна

2024 September 7th

Содержание

List of Figures

1 Цель работы

Целью данной работы является ознакомление с шифрованием гаммированием, а также его программная реализация.

2 Задание

1. Изучить способ шифрования гаммированием.
2. Реализовать алгоритм шифрования гаммированием конечной гаммой на языке программирования Python.

3 Теоретическое введение

Из всех схем шифрования простейшей и наиболее надежной является схема однократного использования:

Формируется $m-$ разрядная случайная двоичная последовательность - ключ шифра. Отправитель производит побитовое сложение по модулю два ($mod 2$) ключа $k = k_1 k_2 ... k_i ... k_m$ и $m-$ разрядной двоичной последовательности $p = p_1 p_2 ... p_i ... p_m$, соответствующей посылаемому сообщению:

$$c_i = p_i \oplus k_i, i = \overline{1, m}$$

где p_i - $i-$й бит исходного текста, k_i - $i-$й бит ключа, $\oplus$ - операция побитового сложения (XOR), c_i - $i-$й бит получившейся криптограммы: $c = c_1 c_2 ... c_i ... c_m$.

Операция побитного сложения является обратимой, то есть $(x \oplus y) \oplus y = x$, поэтому дешифрование осуществляется повторным применением операции $\oplus$ к криптограмме:

$$p_i = c_i \oplus k_i, i = \overline{1, m}$$

Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, то есть псевдослучайной последовательности (ПСП) с выходом генератора G. Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но является детерминированной, то есть известен алгоритм ее формирования. Обычно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N - число букв алфавита открытого текста) [1]

Простейший генератор псевдослуайной последовательности можно представить рекуррентным соотношением:

$$\gamma_i = a * \gamma_{i-1} + b * mod(m), i = \overline{1, m}$$

где γ_i - i-й член последовательности псевдослучайных чисел, a, γ_0, b - ключевые параметры. Такая последовательность состоит из целых чисел от 0 до m-1. Если элементы γ_i и γ_j совпадут, то совпадут и последующие участки: $\gamma_{i+1} = \gamma_{j+1}, \gamma_{i+2} = \gamma_{j+2}$. Таким образом, ПСП является периодической. Знание периода гаммы существенно облегчает криптоанализ. Максимальная длина периода равна m. Для ее достижения необходимо удовлетворить следующим условиям:

- b и m - взаимно простые числа;
- a-1 делится на любой простой делитель числа m;
- a-1 кратно 4, если m кратно 4.

4 Ход выполнения лабораторной работы

Для реализации шифров перестановки будем использовать среду JupyterLab. Выполним необходимую задачу.

1. Задаем функцию определения ключа, учитывая длину шифруемой последовательности.

```
def gen_key(m, pas):
    m = m.lower().replace(' ', '')
    pas = pas.lower().replace(' ', '')
    pas = list(pas)
    if len(m) == len(pas):
        return pas
    else:
        for i in range(len(m)-len(pas)):
            pas.append(pas[i%len(pas)])
    return pas
```

Figure 4.1: Ключ для реализации шифров

2. Прописываем функцию для шифрования переданного текста. Задаем тестовые данные и вызываем функцию:

```python
def gamma(text, pas):
    alphabet = 'абвгдежзийклмнопрстуфхцчшщъыьэюя'
    alphabet = list(alphabet)
    pas = gen_key(text, pas)
    text = list(text)

    res = ''
    for i in range(len(text)):
        c = (ord(text[i]) + ord(pas[i]) - 2*ord('a') + 2) % 31
        res += alphabet[c-1]
    return res
```

```python
gamma('приказ', 'гамма')
```

```
'усхчбл'
```

Figure 4.2: Функция алгоритма шифрования конечной гаммой

Полученное сообщение аналогично приведенному в Методических материалах.

5 Выводы

В рамках данной работы мы изучили и программно реализовали алгоритм шифрования гаммированием конечной гаммой.

6 Список литературы

1. Методические материалы курса[1]