Отчет по лабораторной работе №3

Шифрование гаммированием

Асеинова Елизавета Валерьевна

14 октября 2023

Содержание

Список иллюстраций

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 алфавита для последующего шифрования

```
[23] import numpy as np
     def alphabet(lang):
         if lang == 'eng':
           return(list(map(chr, range(ord('a'), ord('z') +1))))
         elif lang == 'rus':
           return(list(map(chr, range(ord('а'), ord('я') +1))))
```

Рис. 4.1.: Алфавит для реализации шифров

2. Описываем функцию с принципом работы алгоритма шифрования конечной гаммой

```
[44] def encrypt(message: str, gamma: str):
         alph = alphabet('eng')
         # if message.lower() not in alph:
         #   alph = alphabet('rus')
         length = len(alph)
         def gamma_en(letters_pair: tuple):
             ind = (letters_pair[0] + 1) + (letters_pair[1]+1) % length
             if ind > length:
               ind = ind-length
             return ind-1
         clear_message = list(filter(lambda s: s.lower() in alph, message))
         clear_gamma = list(filter(lambda s: s.lower() in alph, gamma))
         ind_message = list(map(lambda s: alph.index(s.lower()), clear_message))
         ind_gamma = list(map(lambda s: alph.index(s.lower()), clear_gamma))
         for i in range(len(ind_message) - len(ind_gamma)):
           ind_gamma.append(ind_gamma[i])
         print(f'{message.upper()} -> {ind_message}\n{gamma.upper()} -> {ind_gamma}')
         ind_encrypt = list(map(lambda s: gamma_en(s), zip(ind_message, ind_gamma)))
         print(f'Форма шифрования: {ind_encrypt}\n')
         return ''.join(list(map(lambda s: alph[s], ind_encrypt))).upper()
```

Рис. 4.2.: Функция алгоритма шифрования конечной гаммой

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

```
[40] def check(message:str, gamma: str):
       print(f'Результат шифрования: {encrypt(message, gamma)}')
```

```
[41] message = 'ПРИКАЗ'
     gamma = 'ГАММА'
     check(message, gamma)

     ПРИКАЗ -> [15, 16, 8, 10, 0, 7]
     ГАММА -> [3, 0, 12, 12, 0, 3]
     Форма шифрования: [19, 17, 21, 23, 1, 11]

     Результат шифрования: УСХЧБЛ
```

```
message = 'HELLO DARKNESS MY OLD FRIEND'
gamma = 'TALK'
check(message, gamma)

HELLO DARKNESS MY OLD FRIEND -> [7, 4, 11, 11, 14, 3, 0, 17, 10, 13, 4, 18, 18, 12, 24, 14, 11, 3, 5, 17, 8, 4, 13, 3]
TALK -> [19, 0, 11, 10, 19, 0, 11, 10, 19, 0, 11, 10, 19, 0, 11, 10, 19, 0, 11, 10, 19, 0, 11, 10]
Форма шифрования: [1, 5, 23, 22, 8, 4, 12, 2, 4, 14, 16, 3, 12, 13, 10, 25, 5, 4, 17, 2, 2, 5, 25, 14]

Результат шифрования: BFXWIEMCEOQDMNKZFERCCFZO
```

Рис. 4.3.: Реализация шифрования гаммированием на примере

Полученное сообщение аналогично приведенному в Методических материалах. Также на скриншоте можно увидеть пример на английском языке

5. Выводы

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

6. Список литературы

1. Методические материалы курса[1]