Отчет по лабораторной работе №3

Дисциплина: Математические основы защиты информации и информационной безопасности

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

Список иллюстраций

1 Цель работы

Ознакомится и реализовать шифрование гаммированием.

2 Задание

Реализовать алгоритм шифрования гаммированием конечной гаммой.

3 Теоретическое введение

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

При использовании генератора ПСП получаем бесконечную гамму. Однако, возможен режим шифрования конечной гаммы. В роли конечной гаммы может выступать фраза. Как и ранее, используется алфавитный порядок букв, т.е. буква «а» имеет порядковый номер 1, «б» – 2 и т.д.

Например, зашифруем слово «ПРИКАЗ» («16 17 09 11 01 08») гаммой «ГАММА» («04 01 13 13 01»). Будем использовать операцию побитового сложения по модулю 33 (*mod 33*). Получаем: c_1 = 16 + 4(*mod 33*) = 20

c_2 = 17 + 1(*mod 33*) = 18

c_3 = 9 + 13(*mod 33*) = 22

c_4 = 11 + 13(*mod 33*) = 24

c_5 = 1 + 1(*mod 33*) = 2

c_6 = 8 + 4(*mod 33*) = 12

Криптограмма: «УСХЧБЛ» («20 18 22 24 02 12»)

4 Выполнение лабораторной работы

4.1 Шаг 1

Ознакомилась с предоставленными теоретическими данными. Для выполнения задания решила использовать язык Python. Создала переменную типа строка, содержащую русский алфавит. Написала функцию, выполняющую шифрование гаммированием конечной гаммой. Код функции и результат ее использования представлен на Рисунке 1 (рис. - fig. 4.1). Функция принимает на вход алфавит, фразу, которую нужно зашифровать, и гамму. Вначале если длина сообщения не совпадает с длинной гаммы, в конец гаммы дописывается необходимое количество символов гаммы, начиная с первого. Затем поочередно осуществляется поиск позиции i-го символа сообщения и гаммы в алфавите, к полученным значениям прибавляется 1 (так как индексация в переменной, содержащей алфавит начинается с 0, а мы условились, что буква “а” имеет индекс 1) и они запоминаются в переменные *a* и *b* соответственно. К строке *shifr* прибавляется символ, находящийся в алфавите на позиции *(a+b) mod(длина алфавита) -1* (вычитание 1 необходимо для соответствия условию, что буква “а” имеет индекс 1). Функция возвращает получившуюся криптограмму.

```
In [43]: alfavit= "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

In [44]: def gam (alfavit, message, gamma):
             shifr=""
             j=0
             if (len(message)>len(gamma)):
                 while len(message)!=len(gamma):
                     gamma=gamma+gamma[j]
                     j=j+1
             for i in range (len(message)):
                 a=alfavit.find(message[i])+1
                 b=alfavit.find(gamma[i])+1
                 print(a,b)
                 shifr=shifr+alfavit[(a+b)%(len(alfavit))-1]
             return shifr

In [45]: gam(alfavit, "приказ", "гамма")

         17 4
         18 1
         10 14
         12 14
         1 1
         9 4

Out[45]: 'усцшбл'
```

Рис. 4.1: Реализация шифрования гаммированием конечной гаммой

4.2 Шаг 2

Получившаяся криптограмма не соответствует данной в примере, так как в примере на самом деле используется алфавит, не содержащий буквы “ё”. Для проверки изменила алфавит и вновь осуществила гаммирование конечной гаммой. Результат представлен на Рисунке 2 (рис. - fig. 4.2) и он совпадает с примером.

```
In [40]: alfavit= "абвгдежзийклмнопрстуфхцчшщъыьэюя"

In [41]: def gam (alfavit, message, gamma):
             shifr=""
             j=0
             if (len(message)>len(gamma)):
                 while len(message)!=len(gamma):
                     gamma=gamma+gamma[j]
                     j=j+1
             for i in range (len(message)):
                 a=alfavit.find(message[i])+1
                 b=alfavit.find(gamma[i])+1
                 print(a,b)
                 shifr=shifr+alfavit[(a+b)%(len(alfavit))-1]
             return shifr

In [42]: gam(alfavit, "приказ", "гамма")

         16 4
         17 1
         9 13
         11 13
         1 1
         8 4

Out[42]: 'усхчбл'
```

Рис. 4.2: Реализация шифрования гаммированием конечной гаммой для другого алфавита

5 Выводы

Я ознакомилась с шифрованием гаммирования конечной гаммой и реализовала его. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.