Отчёт по лабораторной работе №3

Шифр гаммирования

Степанов Виктор НПМмд-02-22

Содержание

[Цель работы 1](#_Toc116133517)

[Теоретические сведения 1](#_Toc116133518)

[Шифр гаммирования 1](#_Toc116133519)

[Выполнение работы 2](#_Toc116133520)

[Реализация шифратора и дешифратора Python 2](#_Toc116133521)

[Контрольный пример 3](#_Toc116133522)

[Выводы 4](#_Toc116133523)

# Цель работы

Изучение алгоритма шифрования гаммированием

# Теоретические сведения

## Шифр гаммирования

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Принцип шифрования гаммированием заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел и наложении полученной гаммы шифра на открытые данные обратимым образом (например, используя операцию сложения по модулю 2). Процесс дешифрования сводится к повторной генерации гаммы шифра при известном ключе и наложении такой же гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, если гамма шифра не содержит повторяющихся битовых последовательностей и изменяется случайным образом для каждого шифруемого слова. Если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа.

Метод гаммирования становится бессильным, если известен фрагмент исходного текста и соответствующая ему шифрограмма. В этом случае простым вычитанием по модулю 2 получается отрезок псевдослучайной последовательности и по нему восстанавливается вся эта последовательность.

Метод гаммирования с обратной связью заключается в том, что для получения сегмента гаммы используется контрольная сумма определенного участка шифруемых данных. Например, если рассматривать гамму шифра как объединение непересекающихся множеств H(j), то процесс шифрования можно пердставить следующими шагами:

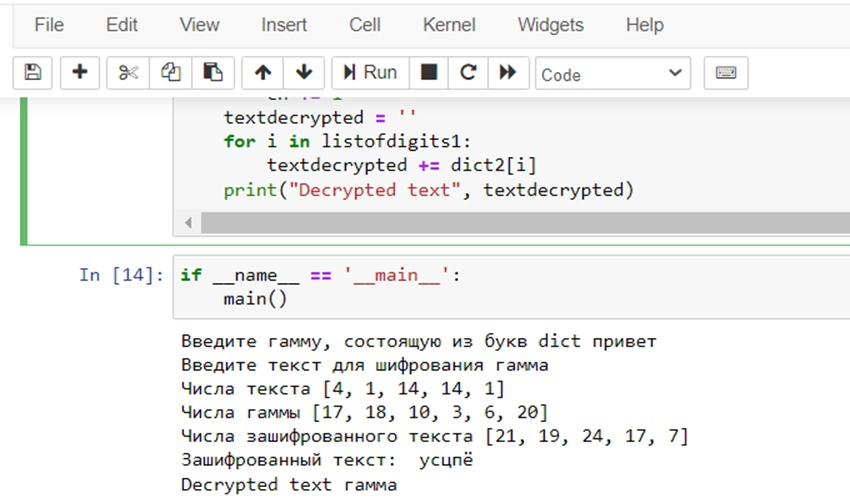
1. Генерация сегмента гаммы H(1) и наложение его на соответствующий участок шифруемых данных.
2. Подсчет контрольной суммы участка, соответствующего сегменту гаммы H(1).
3. Генерация с учетом контрольной суммы уже зашифрованного участка данных следующего сегмента гамм H(2).
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных H(2) и т.д.

# Выполнение работы

## Реализация шифратора и дешифратора Python

def main():  
 #создаем алфавит  
 def main():  
 dict = {"а" :1, "б" :2, "в" :3, "г" :4, "д" :5, "е" :6, "ё" :7, "ж" :8,   
 "з" :9, "и" :10, "й" :11, "к" :12, "л" :13, "м" :14, "н" :15, "о" :16,   
 "п" :17, "р" :18, "с" :19, "т" :20, "у" :21, "ф" :22, "х" :23, "ц" :24,   
 "ч" :25, "ш" :26, "щ" :27, "ъ" :28, "ы" :29, "ь" :30, "э" :31, "ю" :32,   
 "я" :33}  
 dict2 = {v: k for k, v in dict.items()}  
 gamma = input("Введите гамму, состоящую из букв dict ").lower()  
 text = input("Введите текст для шифрования ").lower()  
 listtext = list()  
 listgamma = list()  
 for i in text:  
 listtext.append(dict[i])  
 print("Числа текста", listtext)  
 for i in gamma:  
 listgamma.append(dict[i])  
 print("Числа гаммы", listgamma)  
 listresult = list()  
 ch = 0  
 for i in text:  
 try:  
 a = dict[i] + listgamma[ch]  
 except:  
 ch = 0  
 a = dict[i] + listgamma[ch]  
 if a>=33:  
 a = a%33  
 ch += 1  
 listresult.append(a)  
 print("Числа зашифрованного текста", listresult)  
 textencrypted = ""  
 for i in listresult:  
 textencrypted += dict2[i]  
 print("Зашифрованный текст: ", textencrypted)  
 listofdigits = list()  
 for i in textencrypted:  
 listofdigits.append(dict[i])  
 ch = 0  
 listofdigits1 = list()  
 for i in listofdigits:  
 a = i - listgamma[ch]  
 if a < 1:  
 a += 33  
 listofdigits1.append(a)  
 ch += 1  
 textdecrypted = ''  
 for i in listofdigits1:  
 textdecrypted += dict2[i]  
 print("Decrypted text", textdecrypted)  
   
   
 if \_\_name\_\_ == '\_\_main\_\_':  
 main()

## Контрольный пример



Пример работы алгоритма гаммирования

# Выводы

Я изучил алгоритмы шифрования на основе гаммирования и реализовал их на языке программирования Python.