

Отчёт по лабораторной работе №3

Шифр гаммирования

Милёхин Александр НПМмд-02-21

Содержание

1	Цель работы	4
2	Теоретические сведения	5
2.1	Шифр гаммирования	5
3	Выполнение работы	7
3.1	Реализация шифратора и дешифратора Python	7
3.2	Контрольный пример	9
4	Выводы	10
	Список литературы	11

List of Figures

3.1	Пример работы алгоритма гаммирования	9
-----	--	---

1 Цель работы

Изучение алгоритма шифрования гаммированием

2 Теоретические сведения

2.1 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 крипто- стойкость 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 представить 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 гаммы $H(2)$.
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных $H(2)$ и т.д.

3 Выполнение работы

3.1 Реализация шифратора и дешифратора Python

```
def main():  
    #создаем алфавит  
  
    def main():  
dict = {"a" :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)

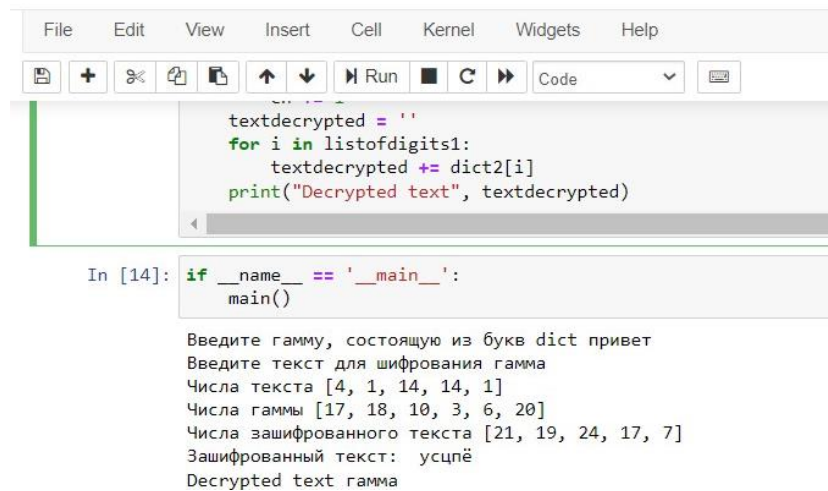
```

```

if __name__ == '__main__':
    main()

```

3.2 Контрольный пример



The screenshot shows a Jupyter Notebook with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The code cell contains the following Python code:

```

textdecrypted = ''
for i in listofdigits1:
    textdecrypted += dict2[i]
print("Decrypted text", textdecrypted)

```

Below the code cell, the execution output is displayed:

```

In [14]: if __name__ == '__main__':
         main()

```

The output text shows the steps of the algorithm:

```

Введите гамму, состоящую из букв dict привет
Введите текст для шифрования гамма
Числа текста [4, 1, 14, 14, 1]
Числа гаммы [17, 18, 10, 3, 6, 20]
Числа зашифрованного текста [21, 19, 24, 17, 7]
Зашифрованный текст: усьпё
Decrypted text гамма

```

Figure 3.1: Пример работы алгоритма гаммирования

4 Выводы

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

Список литературы

1. Шифрование методом гаммирования
2. Режим гаммирования в блочном алгоритме шифрования