

# **Отчёт по лабораторной работе №7**

**Шифр гаммирования**

Поздняков Данила Романович НПИбд-02-19

# Содержание

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

# List of Figures

3.1 Работа 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 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 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 гамм  $H(2)$ .
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
    dict = {"а" :1, "б" :2 , "в" :3 , "г" :4 , "д" :5 , "е" :6 , "ё" :7 , "ж": 8, "з":
            "м": 14, "н": 15, "о": 16, "п": 17,
            "р": 18, "с": 19, "т": 20, "у": 21, "ф": 22, "х": 23, "ц": 24, "ч": 2
            "ы": 29, "ь": 30, "э": 31, "ю": 32, "я": 32
    }

    # меняем местами ключ и значение, такой словарь понадобится в будущем
    dict2 = {v: k for k, v in dict.items()}
    gamma = input("Введите гамму(на русском языке! Да и пробелы тоже нельзя! Короче")
    text = input("Введите текст для шифрования").lower()
    listofdigitsoftext = list() #сюда будем записывать числа букв из текста
    listofdigitsofgamma = list() #для гаммы
    #запишем числа в список
    for i in text:
        listofdigitsoftext.append(dict[i])
    print("Числа текста", listofdigitsoftext)
    #то же самое сделаем с гаммой
    for i in gamma:
        listofdigitsofgamma.append(dict[i])
```

```

print("числа гаммы", listofdigitsofgamma)
listofdigitsresult = list() #сюда будем записывать результат
ch = 0
for i in text:
    try:
        a = dict[i] + listofdigitsofgamma[ch]
    except:
        ch=0
        a = dict[i] + listofdigitsofgamma[ch]
    if a>=33:
        a = a%33
    ch+=1
    listofdigitsresult.append(a)
print("Числа зашифрованного текста", listofdigitsresult)
# теперь обратно числа представим в виде букв
textencrypted=""
for i in listofdigitsresult:
    textencrypted+=dict2[i]
print("Зашифрованный текст: ", textencrypted)
#теперь приступим к реализации алгоритма дешифровки
listofdigits = list()
for i in textencrypted:
    listofdigits.append(dict[i])
ch = 0
listofdigits1 = list()
for i in listofdigits:
    a = i - listofdigitsofgamma[ch]
    #проблемы тут могут быть
    if a < 1:

```



```

        a = 33 + a

    listofdigits1.append(a)

    ch+=1

textdecrypted = ""

for i in listofdigits1:

    textdecrypted+=dict2[i]

print("Decrypted text", textdecrypted)

```

## 3.2 Контрольный пример

```

In [6]: main()

Введите текст гаммы линукслинукс
Введите текст для шифровкибезопасность
Числа текста [2, 6, 9, 16, 17, 1, 19, 15, 16, 19, 20, 30]
Числа гаммы [13, 10, 15, 21, 12, 19, 13, 10, 15, 21, 12, 19]
Числа шифротекста [15]
шифротекст н
рассшифровка 6
Числа шифротекста [15, 16]
шифротекст но
рассшифровка бе
Числа шифротекста [15, 16, 24]
шифротекст ноц
рассшифровка без
Числа шифротекста [15, 16, 24, 4]
шифротекст ноцт
рассшифровка безо
Числа шифротекста [15, 16, 24, 4, 29]
шифротекст ноцы
рассшифровка безоп
Числа шифротекста [15, 16, 24, 4, 29, 20]
шифротекст ноцтп
рассшифровка безопа
Числа шифротекста [15, 16, 24, 4, 29, 20, 32]
шифротекст ноцтыя
рассшифровка безопаз
Числа шифротекста [15, 16, 24, 4, 29, 20, 32, 25]
шифротекст ноцтыяч
рассшифровка безопазч
Числа шифротекста [15, 16, 24, 4, 29, 20, 32, 25, 31]
шифротекст ноцтыяча
рассшифровка безопазно
Числа шифротекста [15, 16, 24, 4, 29, 20, 32, 25, 31, 7]
шифротекст ноцтыячае
рассшифровка безопазноз
Числа шифротекста [15, 16, 24, 4, 29, 20, 32, 25, 31, 7, 32]
шифротекст ноцтыячаея
рассшифровка безопазнозз
Числа шифротекста [15, 16, 24, 4, 29, 20, 32, 25, 31, 7, 32, 16]
шифротекст ноцтыячаеяо
рассшифровка безопазноззо

```

Figure 3.1: Работа алгоритма гаммирования

## **4 Выводы**

Изучили алгоритмы шифрования на основе гаммирования

# Список литературы

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