

# **Отчёт по лабораторной работе №8**

**Шифр гаммирования**

Ласурия Данил

# 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
2.2	Идея взлома . . . . .	6
<b>3</b>	<b>Выполнение работы</b>	<b>8</b>
3.1	Реализация взломщика, шифратора и дешифратора на Python . .	8
3.2	Контрольный пример . . . . .	11
<b>4</b>	<b>Выводы</b>	<b>12</b>
	<b>Список литературы</b>	<b>13</b>

# List of Figures

3.1	Работа алгоритма взлома ключа . . . . .	11
3.2	Работа алгоритма шифрования и дешифровки . . . . .	11

# 1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

## 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 гаммы  $H(2)$ .
4. Подсчет контрольной суммы участка данных, соответствующего сегменту данных  $H(2)$  и т.д.

## 2.2 Идея взлома

Шифротексты обеих телеграмм можно получить по формулам режима однократного гаммирования:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

Открытый текст можно найти, зная шифротекст двух телеграмм, зашифрованных одним ключом. Для это оба равенства складываются по модулю 2. Тогда с учётом свойства операции XOR получаем:

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар  $C_1 \oplus C_2$  (известен вид обеих шифровок). Тогда зная  $P_1$  имеем:

$$C_1 \oplus C_2 \oplus P_1 = P_1 \oplus P_2 \oplus P_1 = P_2$$

Таким образом, злоумышленник получает возможность определить те символы сообщения  $P_2$ , которые находятся на позициях известного шаблона сообщения  $P_1$ . В соответствии с логикой сообщения  $P_2$ , злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения  $P_2$ . Затем вновь используется равенство с подстановкой вместо  $P_1$  полученных на предыдущем шаге новых символов сообщения  $P_2$ . И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

## 3 Выполнение работы

### 3.1 Реализация взломщика, шифратора и дешифратора на Python

```
# создаем алфавит из русских букв и цифр
# он нужен для гаммирования
a = ord("а")
alphabeth = [chr(i) for i in range(a, a + 32)]
a = ord("0")
for i in range(a, a+10):
    alphabeth.append(chr(i))

a = ord("А")
for i in range(1040, 1072):
    alphabeth.append(chr(i))
print(alphabeth)
P1 = "МегаКринге"
P2 = "УльттраБаза"
# длина ключа 20
key = "05 0C 17 7F 0E 4E 37 D2 94 10"

def vzlom(P1, P2):
```



```

code = []
for i in range(10):
    code.append(alphabeth[(alphabeth.index(P1[i]) + alphabeth.index(P2[i]))])
#получили известные символы в шаблоне
print(code)
p3 = "".join(code)
print(p3)

vzlom(P1, P2)

def shifr(P1):
    # создаем алфавит
    dicts = {"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, "А": 34, "Б": 35, "В": 36,
             "Г": 37, "Д": 38, "Е": 39, "Ё": 40, "Ж": 41, "З": 42, "И": 43, "Й": 44, "К": 45, "Л": 46, "М": 47, "Н": 48, "О": 49, "П": 50,
             "Р": 51, "С": 52, "Т": 53, "У": 54, "Ф": 55, "Х": 56, "Ц": 57, "Ч": 58, "Ш": 59, "Щ": 60, "Ъ": 61, "Ы": 62, "Ь": 63, "Э": 64, "Ю": 65, "Я": 66}
    # меняем местами ключ и значение, такой словарь понадобится в будущем
    dict2 = {v: k for k, v in dicts.items()}
    text = P1
    gamma = input("Введите гамму(на русском языке! Да и пробелы тоже нельзя! Короче без пробелов и знаков препинания)")
    listofdigitsoftext = list() # сюда будем записывать числа букв из текста
    listofdigitsofgamma = list() # для гаммы
    # запишем числа в список
    for i in text:
        listofdigitsoftext.append(dicts[i])
    print("Числа текста", listofdigitsoftext)
    # то же самое сделаем с гаммой

```

```

for i in gamma:
    listofdigitsofgamma.append(dict2[i])
print("числа гаммы", listofdigitsofgamma)
listofdigitsresult = list() # сюда будем записывать результат
ch = 0
for i in text:
    try:
        a = dict2[i] + listofdigitsofgamma[ch]
    except:
        ch = 0
        a = dict2[i] + listofdigitsofgamma[ch]
    if a > 75:
        a = a%75
        print(a)
    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(dict2[i])
ch = 0
listofdigits1 = list()
for i in listofdigits:

```

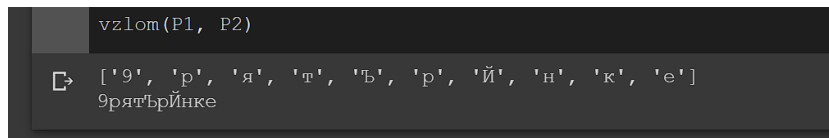
```

try:
    a = i - listofdigitsofgamma[ch]
except:
    ch=0
    a = i - listofdigitsofgamma[ch]
if a < 1:
    a = 75 + a
listofdigits1.append(a)
ch += 1
textdecrypted = ""
for i in listofdigits1:
    textdecrypted += dict2[i]
print("Расшифрованный текст", textdecrypted)

```

shifr(P1)

## 3.2 Контрольный пример

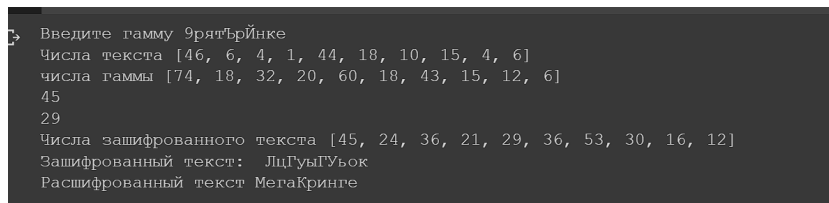


```

vzлом(P1, P2)
['9', 'р', 'я', 'т', 'Ъ', 'р', 'й', 'н', 'к', 'е']
9рятЪринке

```

Figure 3.1: Работа алгоритма взлома ключа



```

Введите гамму 9рятЪринке
Числа текста [46, 6, 4, 1, 44, 18, 10, 15, 4, 6]
числа гаммы [74, 18, 32, 20, 60, 18, 43, 15, 12, 6]
45
29
Числа зашифрованного текста [45, 24, 36, 21, 29, 36, 53, 30, 16, 12]
Зашифрованный текст: ЛцГуыГУьок
Расшифрованный текст МегаКринге

```

Figure 3.2: Работа алгоритма шифрования и дешифровки

## 4 Выводы

В ходе выполнения лабораторной работы было разработано приложение, позволяющее шифровать тексты в режиме однократного гаммирования.

# Список литературы

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