

# **Лабораторная работа № 7**

**Элементы криптографии. Однократное гаммирование**

Абдуллаев Сайидазизхон Шухратович

# Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Ход работы	8
Ответы на котнрольные вопросы	10
Выводы	12

# Список иллюстраций

1	Импорт библиотек и написание функций . . . . .	8
2	Шифрование открытого текста . . . . .	8
3	Проверка правильности работы кода . . . . .	8
4	Расшифровка зашифрованного текста новым ключом . . . . .	9

## **Список таблиц**

## **Цель работы**

Освоить на практике применение режима однократного гаммирования

# Задание

1. Определить вид шифротекста при известном ключе и известном открытом тексте. 2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста

# Теоретическое введение

С точки зрения теории криптоанализа, метод шифрования однократной случайной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым (далее для краткости авторы будут употреблять термин “однократное гаммирование”, держа в уме все вышесказанное). Обоснование, которое привел Шеннон, основываясь на введенном им же понятии информации, не дает возможности усомниться в этом - из-за равных априорных вероятностей криптоаналитик не может сказать о дешифровке, верна она или нет. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть скольконибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях.

## Ход работы

1. Импортируем все необходимые библиотеки и пишем функцию генерирования ключа, а также функцию гаммирования. (Рис. [-@fig:001]).

```
In [1]: import random
import string

In [2]: def gen_key(length, symbols = string.ascii_letters + string.digits):
    return ''.join(random.choice(symbols) for i in range(length))

def gaming(text, key):
    text_conv = [ord(i) for i in text]
    key_conv = [ord(i) for i in key]
    return ''.join(chr(a^b) for a, b in zip(text_conv, key_conv))
```

Рис. 1: Импорт библиотек и написание функций

2. Определяем вид шифротекста при известном ключе и известном открытом тексте. (Рис. [-@fig:002]).

```
In [4]: text = "С Новым Годом, друзья!"
key = gen_key(len(text))
text_shifr = ganning(text, key)
print ("Шифротекст имеет вид:", text_shifr)

Шифротекст имеет вид: Ёо!аЬофЪаКсУТ!аиВу
```

Рис. 2: Шифрование открытого текста

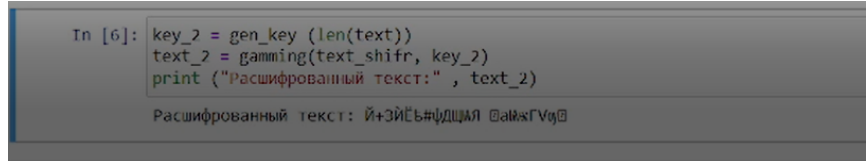
3. Применяем функцию “gamming” к полученному шифру и ключу, чтобы проверить правильность работы программы. В результате снова получаем исходный текст (Рис. [-@fig:003]).

```
In [5]: ganning(ganning(text, key), key)
Out[5]: 'С Новым годом, друзья!'
```

Рис. 3: Проверка правильности работы кода



4. Определяем ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. (Рис. [-@fig:004]).



```
In [6]: key_2 = gen_key (len(text))
text_2 = gamming(text_shifr, key_2)
print ("Расшифрованный текст:" , text_2)

Расшифрованный текст: Й+ЭЙЕЪ#ЩДЦАЛ @аЪхГVq@
```

Рис. 4: Расшифровка зашифрованного текста новым ключом

## Ответы на контрольные вопросы

1. Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Однократное гаммирование – это когда каждый символ попарно с символом ключа складываются по модулю 2 (XOR) (обозначается знаком  $\oplus$ ).
2. Недостатки: Размер ключевого материала должен совпадать с размером передаваемых сообщений. Кроме того, если одну и ту же гамму использовать дважды для разных сообщений, то шифр из совершенно стойкого превращается в «совершенно нестойкий» и допускает дешифрование практически вручную.
3. Преимущества: Метод шифрования случайной однократной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение – информация о вскрытом участке гаммы не дает информации об остальных ее частях. К достоинствам также можно отнести простоту реализации и удобство применения.
4. Потому что каждый символ открытого текста должен складываться с символом ключа попарно.
5. В режиме однократного гаммирования используется сложение по модулю 2 (XOR) между элементами гаммы и элементами подлежащего сокрытию

текста. Особенность заключается в том, что этот алгоритм шифрования является симметричным. Поскольку двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, шифрование и расшифрование выполняется одной и той же программой.

6. Сложить по модулю 2 каждый символ открытого текста и ключа
7. Сложить по модулю 2 каждый символ открытого текста и шифротекста.
8. Необходимые и достаточные условия абсолютной стойкости шифра: Полная случайность ключа; Равенство длин ключа и открытого текста; Однократное использование ключа.

## **Выводы**

В результате выполнения данной работы было освоено на практике применение режима однократного гаммирования