

# Лабораторная работа № 7

Элементы криптографии. Однократное гаммирование

Сухарев Кирилл

# Содержание

Цель работы	5
Условные обозначения и термины	6
Теоретические вводные данные	7
Техническое оснащение и выбранные методы проведения работы	9
Выполнение работы	10
Контрольные вопросы	14
Выводы	16
Библиография	17

# List of Figures

0.1	Работа программы . . . . .	11
-----	----------------------------	----

# List of Tables

## Цель работы

Освоить на практике применение режима однократного гаммирования

## Условные обозначения и термины

Шифрование - обратимое преобразование информации в целях сокрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней.

Гаммирование - метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст.

Случайная величина - в теории вероятностей, величина, принимающая в зависимости от случая те или иные значения с определёнными вероятностями.

# Теоретические вводные данные

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» (рис. 7.1) является простой, но надёжной схемой шифрования данных.

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов. К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении  $\tilde{N}$  все различные ключевые последовательности  $K$  возможны и равновероятны, а значит, возможны и любые

сообщения  $P$ .

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.



# Техническое оснащение и выбранные методы проведения работы

В качестве языка программирования для написания приложения был выбран Python версии 3.9.

# Выполнение работы

1. Для начала подключим два модуля - random (для генерации ключа) и string (для использования алфавита всех символов). Используя данные модули, опишем функцию генерации шифр-ключа указанной длины.

Листинг 1. Функция generate\_key

```
import random
import string

def generate_key(length):
    return ''.join(random.choice(string.ascii_letters + string.digits) for _ in range(length))
```

2. Теперь опишем две функции для удобства вывода информации. Одну - для преобразования строки в шестнадцатиричный формат, другой - для ее вывода сразу в двух форматах.

Листинг 2. Функции to\_hex и beautify

```
def to_hex(input_string):
    return ''.join('{:02x}'.format(ord(symbol)) for symbol in input_string)

def beautify(text):
    return "{} | {}".format(text, to_hex(text))
```

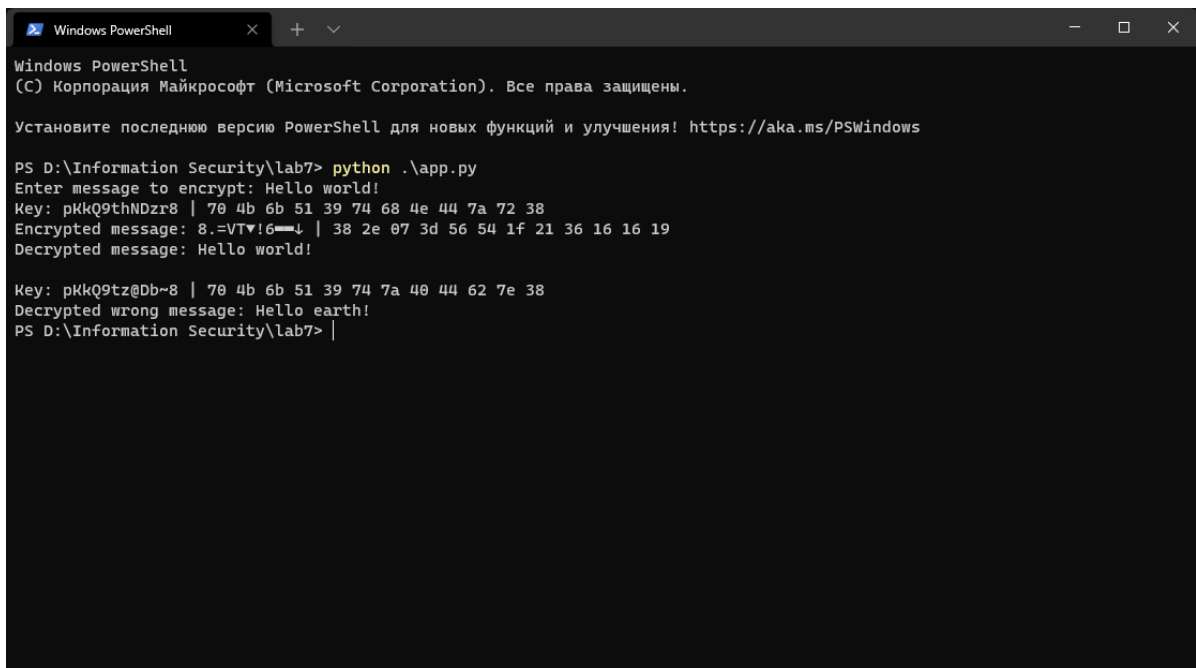
3. Выполняя побитовое сложение по модулю два, опишем функцию однократного гаммирования. Несмотря на то, что эта функция способна выполнить как

декодирование, так и поиск ключа, для удобства выпишем для этих действий отдельные функции.

Листинг 3. Функции `single_gamming`, `decrypt`, `find_key`

```
def single_gamming(message, key):  
    return ''.join(chr(ord(m) ^ ord(k)) for m, k in zip(message, key))  
  
def decrypt(encrypted, key):  
    return single_gamming(encrypted, key)  
  
def find_key(message, encrypted):  
    return single_gamming(message, encrypted)
```

4. Продемонстрируем работу программы (fig. 0.1).



```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Установите последнюю версию PowerShell для новых функций и улучшения! https://aka.ms/PSWindows

PS D:\Information Security\lab7> python .\app.py
Enter message to encrypt: Hello world!
Key: pKkQ9thNDzr8 | 70 4b 6b 51 39 74 68 4e 44 7a 72 38
Encrypted message: 8.=VTv!6— | 38 2e 07 3d 56 54 1f 21 36 16 16 19
Decrypted message: Hello world!

Key: pKkQ9tz@Db~8 | 70 4b 6b 51 39 74 7a 40 44 62 7e 38
Decrypted wrong message: Hello earth!
PS D:\Information Security\lab7> |
```

Figure 0.1: Работа программы

Листинг 4. Весь код программы

```

import random
import string

def generate_key(length):
    return ''.join(random.choice(string.ascii_letters + string.digits) \
                    for _ in range(length))

def to_hex(input_string):
    return ' '.join('{:02x}'.format(ord(symbol)) \
                    for symbol in input_string)

def beuatify(text):
    return "{} | {}".format(text, to_hex(text))

def single_gamming(message, key):
    return ''.join(chr(ord(m) ^ ord(k)) for m, k in zip(message, key))

def decrypt(encrypted, key):
    return single_gamming(encrypted, key)

def find_key(message, encrypted):
    return single_gamming(message, encrypted)

message = input("Enter message to encrypt: ")

key = generate_key(len(message))
print("Key:", beuatify(key))

encrypted = single_gamming(message, key)
print("Encrypted message:", beuatify(encrypted))

```

```
print("Decrypted message:", decrypt(encrypted, key))

wrong_message = "Hello earth!"
wrong_key = find_key(encrypted, wrong_message)
print("\nKey:", beuatify(wrong_key))

print("Decrypted wrong message:", decrypt(encrypted, wrong_key))
```

# Контрольные вопросы

1. Поясните смысл однократного гаммирования.

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

2. Перечислите недостатки однократного гаммирования.

Нестойкость шифра при повторном использовании ключа и последовательность доступа к информации.

3. Перечислите преимущества однократного гаммирования.

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

4. Почему длина открытого текста должна совпадать с длиной ключа?

Потому что для каждого символа строки выполняется операция сложения по модулю 2 с символом ключа. Поэтому размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности?

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) между элементами гаммы и элементами подлежащего сокрытию текста. Метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

6. Как по открытому тексту и ключу получить шифротекст?

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:  $C_i = P_i \oplus K_i$ , где  $\tilde{N}_i$  —  $i$ -й символ получившегося зашифрованного послания,  $P_i$  —  $i$ -й символ открытого текста,  $K_i$  —  $i$ -й символ ключа.

7. Как по открытому тексту и шифротексту получить ключ?

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с правилом  $C_i \oplus P_i = K_i$ , а именно, обе части равенства необходимо сложить по модулю 2 с  $P_i$

8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.

## Выводы

Было освоено на практике применение режима однократного гаммирования



# Библиография

1. Использование однократного гаммирования. URL: <https://www.arhivinfo.ru/1-15068.html> (Дата обращения: 11.12.2021).
2. Д. С. Кулябов, А. В. Королькова, М. Н. Геворкян. Информационная безопасность компьютерных сетей: лабораторные работы. // Факультет физико-математических и естественных наук. М.: РУДН, 2015. 64 с..