

Лабораторная работа №3

Шифрование гаммированием

Доборщук Владимир Владимирович, НФИмд-02-22

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 9 |
| 4.1 | Модули и вспомогательные функции | 9 |
| 4.2 | Реализация шифрования гаммированием | 9 |
| 4.3 | Тестирование | 11 |
| 4.4 | Результаты тестирования | 12 |
| 5 | Выводы | 14 |
| | Список литературы | 15 |

Список иллюстраций

| | | |
|-----|---|----|
| 4.1 | Вывод программы с реализованным шифром гаммирования ко- нечной гаммы | 12 |
|-----|---|----|

Список таблиц

1 Цель работы

Цель данной работы — изучить методы шифрования гаммированием.

2 Задание

Заданием является:

- Реализовать алгоритм шифрования гаммированием конечной гаммы.

3 Теоретическое введение

Из всех схем шифрования простейшей и наиболее надежной является схема однократного использования. Формируется m -разрядная случайная двоичная последовательность - ключ шифра. Отправитель производит побитовое сложение по модулю два ($\text{mod } 2$) ключа

$$k = k_1 k_2 \dots k_i \dots k_m$$

и m -разрядной двоичной последовательности

$$p = p_1 p_2 \dots p_i \dots p_m$$

соответствующей посылаемому сообщению

$$c_i = p_i \oplus k_i, i = \overline{1, m}$$

Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст *гаммы* шифра, т.е. *псевдослучайной последовательности (ПСП)* с выходов генератора \mathbb{G} [1]. Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но является детерминированной, т.е. известен алгоритм ее формирования. Чаще Обычно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N - число букв алфавита открытого текста).

Простейший генератор псевдослучайной последовательности можно представить рекуррентным соотношением:

$$\gamma_i = a \cdot \gamma_{i-1} + b \bmod(m), i = \overline{1, m}$$

где y_i - i -й член последовательности псевдослучайных число, a, y_0, b - ключевые параметры.

При использовании генератора ПСП получаем бесконечную гамму. Однако, возможен режим шифрования конечной гаммы. В роли конечной гаммы может выступать фраза. Как и ранее, используется алфавитный порядок букв.

Замечание

В примере, данном в описании лабораторной работы, допущена ошибка - берется алфавит без буквы “ё”, т.е. алфавит длины 32, хотя указан алфавит длины 33.

4 Выполнение лабораторной работы

Для реализации шифров мы будем использовать Python, так как его синтаксис позволяет быстро реализовать необходимые нам алгоритмы.

4.1 Модули и вспомогательные функции

Дополнительно мы используем библиотеку numpy и импортируем её.

```
import numpy as np
```

Также, реализовали функцию получения английского и русского алфавита.

```
# Cyrillic or Latin alphabet getter
def get_alphabet(option="eng"):
    if option == "eng":
        return list(map(chr, range(ord("a"), ord("z")+1)))
    elif option == "rus":
        return list(map(chr, range(ord("a"), ord("я")+1)))
```

4.2 Реализация шифрования гаммированием

Шифрование гаммированием реализуем в виде функции gamma_encryption следующего вида:

```

# Gamma Encryption
def gamma_encryption(message: str, gamma: str):
    alphabet = get_alphabet()
    if message.lower() not in alphabet:
        alphabet = get_alphabet("rus")

    print(alphabet)
    m = len(alphabet)

    def encrypt(letters_pair: tuple):
        idx = (letters_pair[0] + 1) + (letters_pair[1] + 1) % m
        if idx > m:
            idx = idx - m

        return idx - 1

    message_cleared = list(filter(lambda s: s.lower() in alphabet,
↪ message))
    gamma_cleared = list(filter(lambda s: s.lower() in alphabet,
↪ gamma))

    message_indices = list(map(lambda s: alphabet.index(s.lower()),
↪ message_cleared))
    gamma_indices = list(map(lambda s: alphabet.index(s.lower()),
↪ gamma_cleared))

    for i in range(len(message_indices) - len(gamma_indices)):
        gamma_indices.append(gamma_indices[i])

```

```

print(f'{message.upper()} -> {message_indices}\n{gamma.upper()}
↳ -> {gamma_indices}')

encrypted_indices = list(map(lambda s: encrypt(s),
↳ zip(message_indices, gamma_indices)))

print(f"ENCRYPTED FORM: {encrypted_indices}\n")

return ''.join(list(map(lambda s: alphabet[s],
↳ encrypted_indices))).upper()

```

На вход она принимает переменные `message` (передаваемое сообщение), `gamma` (гамма-ключ).

В ходе обработке мы работаем с индексами элементов массива-строки, предварительно проверяя, является ли первый элемент сообщения частью передаваемого алфавита.

Далее, мы очищаем сообщений от символов, не входящих в алфавит, что дает нам возможность сделать соответствие индексов гаммы-ключа и самого сообщения.

В результате, каждый из символов проходит процедуру шифрованию, и мы получаем зашифрованное сообщение по определенной гамме.

4.3 Тестирование

Для тестирования мы создали следующую функцию, которую вызываем в блоке *Main*:

```

# --- Tests ---

def test_encryption(message: str, gamma: str):
    print(f'ENCRYPTION RESULT: {gamma_encryption(message, gamma)}')

```

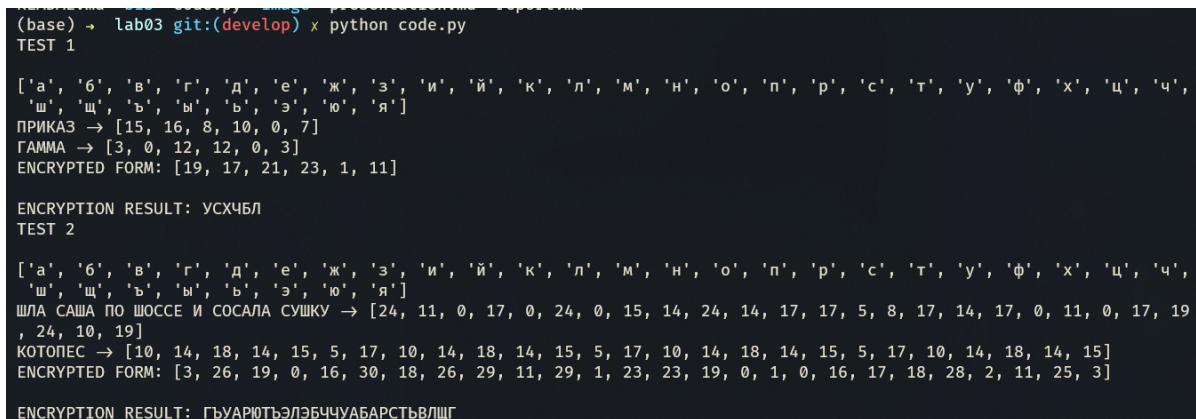
Данные тесты возвращают строку шифро-текста в качестве результата.

Для их вызова, реализуем функцию main следующим образом:

```
# --- Main function ---  
  
def main():  
    message = "приказ"  
    gamma = "гамма"  
  
    print("TEST 1\n")  
    test_encryption(message, gamma)  
  
    message = "Шла Саша по шоссе и сосала сушку"  
    gamma = "Котопес"  
  
    print("TEST 2\n")  
    test_encryption(message, gamma)
```

4.4 Результаты тестирования

Запустив наш программный код, получим результат, изображенный на рисунке 4.1.



```
(base) → lab03 git:(develop) x python code.py  
TEST 1  
  
['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч',  
 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']  
ПРИКАЗ → [15, 16, 8, 10, 0, 7]  
ГАММА → [3, 0, 12, 12, 0, 3]  
ENCRYPTED FORM: [19, 17, 21, 23, 1, 11]  
  
ENCRYPTION RESULT: УСХЧБЛ  
TEST 2  
  
['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч',  
 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']  
ШЛА САША ПО ШОССЕ И СОСАЛА СУШКУ → [24, 11, 0, 17, 0, 24, 0, 15, 14, 24, 14, 17, 17, 5, 8, 17, 14, 17, 0, 11, 0, 17, 19,  
 24, 10, 19]  
КОТОПЕС → [10, 14, 18, 14, 15, 5, 17, 10, 14, 18, 14, 15, 5, 17, 10, 14, 18, 14, 15, 5, 17, 10, 14, 18, 14, 15]  
ENCRYPTED FORM: [3, 26, 19, 0, 16, 30, 18, 26, 29, 11, 29, 1, 23, 23, 19, 0, 1, 0, 16, 17, 18, 28, 2, 11, 25, 3]  
  
ENCRYPTION RESULT: ГЬУАРЮТЪЭЛЭБЧУАБАРСТЬВЛЩГ
```

Рис. 4.1: Вывод программы с реализованным шифром гаммирования конечной гаммы

Явно получим вот такой результат:

TEST 1

```
['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',  
↪ 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ',  
↪ 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
```

ПРИКАЗ -> [15, 16, 8, 10, 0, 7]

ГАММА -> [3, 0, 12, 12, 0, 3]

ENCRYPTED FORM: [19, 17, 21, 23, 1, 11]

ENCRYPTION RESULT: УСХЧБЛ

TEST 2

```
['а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',  
↪ 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ',  
↪ 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
```

ШЛА САША ПО ШОССЕ И СОСАЛА СУШКУ -> [24, 11, 0, 17, 0, 24, 0, 15, 14,
↪ 24, 14, 17, 17, 5, 8, 17, 14, 17, 0, 11, 0, 17, 19, 24, 10, 19]

КОТОПЕС -> [10, 14, 18, 14, 15, 5, 17, 10, 14, 18, 14, 15, 5, 17, 10,
↪ 14, 18, 14, 15, 5, 17, 10, 14, 18, 14, 15]

ENCRYPTED FORM: [3, 26, 19, 0, 16, 30, 18, 26, 29, 11, 29, 1, 23, 23,
↪ 19, 0, 1, 0, 16, 17, 18, 28, 2, 11, 25, 3]

Сравнивая результат шифрования с примером из описания лабораторной работы, можем убедиться, что наша реализация корректна.

5 Выводы

В рамках выполненной лабораторной работы мы изучили и реализовали алгоритм шифрования гаммированием конечной гаммы.

Список литературы

1. Сокол Д.С., Урубкин М.Ю. Шифрование и расшифрование текстовой информации при помощи ключевого слова на основе метода гаммирования и шифра Виженера. 2019.