### Лабораторная работа №3

Шифрование гаммированием

Доборщук В.В.

15 октября 2022

Российский университет дружбы народов, Москва, Россия



#### Докладчик

- Доборщук Владимир Владимирович
- студент группы НФИмд-02-22, студ. билет 1132223451
- учебный ассистент кафедры прикладной информатики и теории вероятностей
- Российский университет дружбы народов
- · doborschuk-vv@rudn.ru



Цели и задачи

**Цель работы** — изучить методы шифрования гаммированием.

#### Задачами являются:

• Реализовать алгоритм шифрования гаммированием конечной гаммы

# Теоретическое введение

#### Теоретическое введение

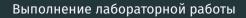
Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, т.е. ncesdocnyuaŭhoŭ nocnedosamenыюcmu ( $\Pi C\Pi$ ) с выходов генератора  $\mathbb G$ . Псевдослучайная последовательность по своим статистическим свойствам неотличима от случайной последовательности, но 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).

#### Шифрование гаммированием

Простейший генератор псевдослучайной последовательности можно представить рекуррентным соотношением:

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

Выполнение лабораторной работы



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

#### Модули и вспомогательные фукнции

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)))
```

```
# 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()
```

#### 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)}')
```

Для вызова теста, 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"
   датта = "Котопес"
   print("TEST 2\n")
   test_encryption(message, gamma)
```

Рис. 1: Вывод программы с реализованным шифром гаммирования конечной гаммы

#### Результаты тестирования

ENCRYPTION RESULT: УСХЧБЛ

#### TEST 1

```
['a', '6', 'в', 'г', 'Д', 'e', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'o', 'п', 'p', 'с', 'т', \hookrightarrow 'y', 'ф', 'х', 'ц', 'ч', 'ш', 'ш', 'ь', 'ы', 'ь', 'э', 'ю', 'я'] ПРИКАЗ \rightarrow [15, 16, 8, 10, 0, 7] ГАММА \rightarrow [3, 0, 12, 12, 0, 3] ENCRYPTED FORM: [19, 17, 21, 23, 1, 11]
```

#### TEST 2

```
['a', '6', 'B', '\Gamma', '\Lambda', 'e', 'M', '3', 'u', '\tilde{u}', 'K', '\Lambda', 'M', 'H', 'o', '\Pi', '\Gamma', '\Gamma
```

## Выводы



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