

# Лабораторная №7

## Основы информационной безопасности

---

Жибицкая Е.Д.

Российский университет дружбы народов, Москва, Россия

Цель

---

- Освоение на практике применения режима однократного гаммирования, разработка приложения, позволяющего шифровать и дешифровать данные в режиме однократного гаммирования

## Ход работы

---

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

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 текста.

```
import random
import string

# Генерация ключа
def key_generation(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.ascii_letters + string.digits)
    return key

# Шифровка дешифровка текста
def encryption(text, key):
    n_text = ''
    for i in range(len(text)):
        n_text += chr(ord(text[i]) ^ ord(key[i]))
    return n_text

text = 'С Новым годом, друзья!'
key = key_generation(text)
encrypt = encryption(text, key)
decrypt = encryption(encrypt, key)

target_fragment = 'С Новым г' # Фрагмент текста
keys = find_key(encrypt, target_fragment)

print('Открытый текст:', text)
print('Ключ:', key)
print('Шифротекст:', encrypt)
print('Дешифровка:', decrypt)
```

Рис. 1: Создание функций

```
Открытый текст: С Новым годом, друзья!
Ключ: m1MXWECPSx7dHhE56Hluwf
Шифротекст: ЫIèÀæŸŵрQцѓњVDeĖŦhйиG
Дешифровка: С Новым годом, друзья!
```

Рис. 2: Результат работы программы

```

#Нахождение ключ для преобразования фрагмента шифротекста в целевой текст
def find_key(n_text, fragment):
    ... possible_keys = []
    ... for i in range(len(n_text) - len(fragment) + 1):
    ...     key_part = ''.join(chr(ord(c) ^ ord(p))
    ...         for c, p in zip(n_text[i:i+len(fragment)], fragment))
    ...     possible_keys.append(key_part)
    ... return possible_keys

# Проверка результатов
print("\nПоиск ключа для фрагмента:", target_fragment)
for idx, key_part in enumerate(keys):
    ... decrypted_part = encryption(encrypt[idx:idx+len(target_fragment)], key_part)
    ... if decrypted_part == target_fragment:
    ...     x = decrypted_part
    ...     y = key_part
print(x, 'ключ:', y)

```

Рис. 3: Функция для 2 части задания

```

> Открытый текст: С Новым годом, друзья!
  Ключ: m1MXWECPSx7dnhE56Hluwf
  Шифротекст: ЫIèÀкѸрQцгъVDeĖŤñйиG
  Дешифровка: С Новым годом, друзья!

```

```

Поиск ключа для фрагмента: С Новым г
С Новым г ключ: кЕ@Н9@ИV

```

Рис. 4: Результат шифровки и дешифровки

## Выводы

---



- В ходе работы при получены навыки однократного гаммирования, реализована программа осуществляющая шифровку и дешифровку данных