# Основы информационной безопасности

## Лабораторная работа № 7. Элементы криптографии. Однократное гаммирование

Нзита Диатезилуа Катенди

# Содержание

# Список иллюстраций

# ПЦель работы

Освоить на практике применение режима однократного гаммирования.

# Задание

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 открытого текста

# Теоретические сведения

Гаммиирование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст[@intro_crypto_2017]:. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных.

# Выполнение лабораторной работы

Генерация ключа (gen_key): Функция создает случайный ключ для каждого символа текста. Каждый элемент ключа представлен в шестнадцатеричной системе. (рис. @fig:001)

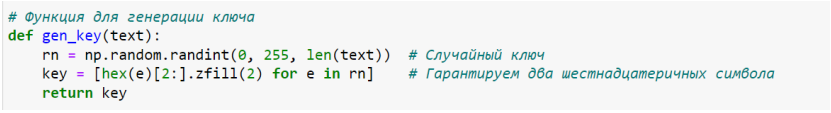


Рис. 1: Генерация ключа (gen_key)

Шифрование текста (crypt_message): Для каждого символа текста применяется кодировка в шестнадцатеричное значение, а затем используется операция XOR с ключом для получения зашифрованного текста.(рис. @fig:002)


Рис. 2: Шифрование текста (crypt_message)

Расшифровка текста (decrypt_message): Процесс аналогичен шифрованию, но

операция XOR применяется к зашифрованному тексту с ключом для восстановления исходного текста. (рис. @fig:003)

```python
# Функция для расшифровки сообщения
def decrypt_message(crypted_text, key):
    hex_crypted_text = []

    for ch in crypted_text:
        hex_crypted_text.append(ch.encode("cp1251").hex())

    print("Шестнадцатеричный зашифрованный текст: ", *hex_crypted_text)
    print("Ключ: ", *key)

    hex_open_text = []
    for i in range(len(hex_crypted_text)):
        # Операция XOR для расшифровки
        open_val = int(hex_crypted_text[i], 16) ^ int(key[i], 16)
        hex_open_text.append("{:02x}".format(open_val))

    print("Шестнадцатеричный открытый текст: ", *hex_open_text)
    open_text = bytearray.fromhex("".join(hex_open_text)).decode("cp1251", errors="ignore")
    print(f"Открытый текст: {open_text}")

    return open_text
```

Рис. 3: Расшифровка текста (decrypt_message)

Нахождение ключа (find_key): Операция XOR между шестнадцатеричными значениями открытого текста и зашифрованного текста позволяет найти ключ.(рис. @fig:004)

```python
# Функция для нахождения ключа
def find_key(open_text, crypted_text):
    print(f"Открытый текст: {open_text}\nЗашифрованный текст: {crypted_text}")
    hex_open_text = []

    # Кодируем открытый текст в шестнадцатеричный формат
    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    hex_crypted_text = []
    # Кодируем зашифрованный текст в шестнадцатеричный формат
    for ch in crypted_text:
        hex_crypted_text.append(ch.encode("cp1251").hex())

    print("Шестнадцатеричный открытый текст: ", *hex_open_text)
    print("Шестнадцатеричный зашифрованный текст: ", *hex_crypted_text)

    # Ключ извлекается путем применения XOR (^) между открытым текстом и зашифрованным текстом
    key = [hex(int(i, 16) ^ int(j, 16))[2:].zfill(2) for (i, j) in zip(hex_open_text, hex_crypted_text)]
    print("Ключ: ", *key)

    return key
```

Рис. 4: Нахождение ключа (find_key)

Проверка Шифрования(рис. @fig:005)

```python
# Пример использования
raw = "С Новым Годом, друзья!"  # Открытый текст
key1 = gen_key(raw)  # Генерация ключа

ct = crypt_message(raw, key1)  # Шифрование текста
print("\n--- Расшифровка ---")
dct = decrypt_message(ct, key1)  # Расшифровка
```

```
Открытый текст: С Новым Годом, друзья!
Шестнадцатеричный открытый текст:  43 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
Ключ:  9b 84 a2 9e a4 d4 f8 e3 d5 49 a0 94 f9 72 a8 ca cd 6f 93 6b aa 97
Шестнадцатеричный зашифрованный текст:  d8 a4 6f 70 46 2f 14 c3 16 a7 44 7a 15 5e 88 2e 3d 9c 74 97 55 b6
Зашифрованный текст: ШЖорF/Г§Dz^€.=њt–U¶

--- Расшифровка ---
Шестнадцатеричный зашифрованный текст:  d8 a4 6f 70 46 2f 14 c3 16 a7 44 7a 15 5e 88 2e 3d 9c 74 97 55 b6
Ключ:  9b 84 a2 9e a4 d4 f8 e3 d5 49 a0 94 f9 72 a8 ca cd 6f 93 6b aa 97
Шестнадцатеричный открытый текст:  43 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
Открытый текст: С Новым Годом, друзья!
```

Рис. 5: Проверка Шифрования

Если используется исходный открытый текст, то ключи key1 и key2 должны совпадать, и результат будет True.

Если изменен открытый текст (например, добавлен вопросительный знак), ключи key1 и key3 будут различаться, и результат будет False.(рис. @fig:006)

```python
# Проверка нахождения ключа
key2 = find_key(raw, ct)
print(key1 == key2)  # Должно быть True

# Тест с измененным текстом
key3 = find_key("С Новым Годом, друзья?", ct)
print(key1 == key3)  # Должно быть False
```

```
Открытый текст: С Новым Годом, друзья!
Зашифрованный текст: Убцеє„Iu%/©¦„в=цµ
Шестнадцатеричный открытый текст:  43 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
Шестнадцатеричный зашифрованный текст:  d3 e1 9f 65 ba 84 b2 75 89 05 2f a9 a6 05 17 84 ad e2 3d 9f b5 14
Ключ:  90 c1 52 8b 58 7f 5e 55 4a eb cb 47 4a 29 37 60 5d 11 da 63 4a 35
True
Открытый текст: С Новым Годом, друзья?
Зашифрованный текст: Убцеє„Iu%/©¦„в=цµ
Шестнадцатеричный открытый текст:  43 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 3f
Шестнадцатеричный зашифрованный текст:  d3 e1 9f 65 ba 84 b2 75 89 05 2f a9 a6 05 17 84 ad e2 3d 9f b5 14
Ключ:  90 c1 52 8b 58 7f 5e 55 4a eb cb 47 4a 29 37 60 5d 11 da 63 4a 2b
False
```

Рис. 6: Ожидаемый результат

# Контрольные вопросы

1. Поясните смысл однократного гаммирования.

Гаммиирование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть.

2. Перечислите недостатки однократного гаммирования.

- Если один и тот же ключ используется для шифрования нескольких сообщений, это может привести к уязвимостям. Например, если злоумышленник узнает открытый текст и соответствующий шифротекст, он может использовать эту информацию для взлома ключа.
- Однократное гаммирование не обеспечивает аутентификацию или целостность данных. Это означает, что злоумышленник может изменить шифротекст без заметных изменений в открытом тексте.

3. Перечислите преимущества однократного гаммирования.

- Однократное гаммирование обеспечивает высокий уровень конфиденциальности, поскольку шифротекст не может быть легко взломан без знания ключа.
- Однократное гаммирование обеспечивает равномерное распределение вероятностей для каждого символа в шифротексте, что делает его статистически неразличимым от случайной последовательности.

- Однократное гаммирование является простым и быстрым методом шифрования.

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

Если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть.

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

В режиме однократного гаммирования используется операция XOR (исключающее ИЛИ). Операция XOR комбинирует биты открытого текста и ключа, чтобы получить шифротекст. Особенностью операции XOR является то, что она возвращает 1 только в том случае, если один из входных битов равен 1, но не оба.

6. Как по открытому тексту и ключу получить шифротекст?

Нужно побитово сложить по модулю численное представление символов в ключе и в открытом тексте.

7. Как по открытому тексту и шифротексту получить ключ?

Нужно побитово сложить по модулю численное представление символов в шифротексте и в открытом тексте.

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

Необходимые и достаточные условия абсолютной стойкости шифра:

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

# Выводы

В результаты выполнения работы были освоены практические навыки применения режима однократного гаммирования.

# Список литературы