Основы информационной безопасности. Лабораторная работа № 7

Элементы криптографии. Однократное гаммирование

Нзита Диатезилуа Катенди

19 октября 2024 г.

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Нзита Диатезилуа Катенди
- студент
- Российский университет дружбы народов
- 1032215220@pfur.ru
- https://github.com/NzitaKatendi

Вводная часть

Цели и задачи

Целью данной работы являестя освоить на практике применение режима однократного гаммирования.

Задачи:

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.

Инструмент: Python

Выполнение лабораторной работы

Функции для реализации однократного гаммирования

```python
# Функция для генерации ключа
def gen_key(text):
    rn = np.random.randint(0, 255, len(text))  # Случайный ключ
    key = [hex(e)[2:].zfill(2) for e in rn]    # Гарантируем два шестнадцатеричных символа
    return key
```

Рис. 1: Генерация ключа

Шифрование и дешифрование методом однократного гаммирования

```python
# Функция для шифрования сообщения
def crypt_message(open_text, key):
    print(f"Открытый текст: {open_text}")
    hex_open_text = []

    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    print("Шестнадцатеричный открытый текст: ", *hex_open_text)
    print("Ключ: ", *key)

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

    print("Шестнадцатеричный зашифрованный текст: ", *hex_crypted_text)
    crypted_text = bytearray.fromhex("".join(hex_crypted_text)).decode("cp1251", errors="ignore")
    print(f"Зашифрованный текст: {crypted_text}")

    return crypted_text
```

Рис. 2: Шифрование текста

Расшифровка текста

```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

Нахождение ключа

```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

Проверка Шифрования

```
# 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: ШюрF/ВГШ§Dz‼^€.=œt—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

Ожидаемый результат

```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: Убцее„IuЫ□/O|□□„в=цы□
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: Убцее„IuЫ□/O|□□„в=цы□
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

Заключение

Выводы

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

Список литературы

1. Ященко В. В. Введение в криптографию. МЦНМО, 2017. 349 с.