Отчёт по лабораторной работе

Режим однократного гаммирования

Дмитрий Сергеевич Кулябов

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучение принципов шифрования в режиме однократного гаммирования, разработка приложения для шифрования и дешифрования данных, а также анализ криптостойкости метода.

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

3 Описание процесса выполнения задания

3.1 Теоретическое введение

Однократное гаммирование — это метод шифрования, при котором каждый символ открытого текста объединяется с соответствующим символом ключа с помощью операции XOR (исключающее ИЛИ).

Формулы:

- Шифрование: Ci=Pi⊠KiCi=Pi⊠Ki
- Дешифрование: Pi=Ci⊠KiPi=Ci⊠Ki
- Нахождение ключа: Ki=Pi⊠CiKi=Pi⊠Ci

3.2 Практическая реализация

Задача 1: Шифрование известного текста с заданным ключом Шаги:

```
Преобразовать текст и ключ в байты (например, в кодировке UTF-8).

Применить операцию XOR к каждому байту.

Получить шифротекст.
```

Задача 2: Подбор ключа для заданного шифротекста

Дано:

Шифротекст: b'\x12\x45\x67...' (из предыдущего шага)

Желаемый открытый текст: "С Новым Годом, друзья!"

Шаги:

Преобразовать желаемый текст в байты.

Вычислить ключ: Ki=Pi⊕CiKi=Pi⊕Ci.

```
[lchileshe@lchileshe ~]$ python3 --version
Python 3.9.21
[lchileshe@lchileshe ~]$ mkdir xor_lab
[lchileshe@lchileshe ~]$ cd xor_lab/
[lchileshe@lchileshe xor_lab]$ nona xor_cipher.py
bash: nona: command not found...
[lchileshe@lchileshe xor_lab]$ nano xor_cipher.py
```

```python
def xor_cipher(data: bytes, key: int) -> bytes:
    return bytes([b ^ key for b in data])

def identify_cipher_type(plaintext: bytes, ciphertext: bytes, key: int) -> st>
    if xor_cipher(plaintext, key) == ciphertext:
        return "XOR"
    else:
        return "Unknown or unsupported cipher"

def recover_key(plaintext: bytes, ciphertext: bytes) -> int:
    return plaintext[0] ^ ciphertext[0]

def main():
    message = "Happy New Year, friends!"
    key = 42  # sample key
    plaintext = message.encode()

    # Step 1: Encryption
    ciphertext = xor_cipher(plaintext, key)
    print("Encrypted (hex):", ciphertext.hex())

    # Step 2: Decryption with known key
```

```
Encrypted (hex): 624b5a5a530a644f5d0a734f4b58060a4c58434f444e590b
Decrypted: Happy New Year, friends!
Cipher Type: XOR
Recovered Key: 42
Decrypted with recovered key: Happy New Year, friends!
```

4 Выводы

Однократное гаммирование обеспечивает высокую криптостойкость при соблюдении усло

Метод уязвим при повторном использовании ключа или если ключ короче текста.

Операция XOR обратима, что делает метод удобным для шифрования и дешифрования.

5 Ответы на контрольные вопросы

1. Смысл однократного гаммирования Это метод шифрования, где каждый бит открытого текста комбинируется с битом ключа через XOR.
2. Недостатки

   ```
   Требуется ключ той же длины, что и сообщение.
   ```

   ```
   Ключ нельзя использовать повторно.
   ```

3. Преимущества

   ```
   Абсолютная стойкость при правильном использовании.
   ```

   ```
   Простота реализации.
   ```

4. Почему длина ключа должна совпадать с длиной текста? Если ключ короче, его приходится повторять, что снижает стойкость.
5. Используемая операция XOR — обратимая битовая операция.
6. Как получить шифротекст? Ci=Pi⊠KiCi=Pi⊠Ki.
7. Как получить ключ? Ki=Pi⊠CiKi=Pi⊠Ci.
8. Условия абсолютной стойкости

Ключ должен быть случайным.

Длина ключа ⩾ длины текста.

Ключ используется однократно.