Отчёт по лабораторной работе №7

Элементы криптографии. Однократное гаммирование

Данилова Анастасия Сергеевна

Содержание

0.1 Цель лабораторной работы 1
1 Выполнение работы 1
1.1 Контрольные вопросы 2
2 Вывод 3

0.1 Цель лабораторной работы

Освоить на практике применение режима однократного гаммирования.

1 Выполнение 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 открытого текста.

```python
def encrypt(plaintext, key):
    ciphertext = ""
    for i in range(len(plaintext)):
        char = plaintext[i]
        key_char = key[i % len(key)]  # Получаем символ ключа по модулю длины ключа
        encrypted_char = chr((ord(char) + ord(key_char)) % 65536)  # Шифрование символа путем сложения кодов символов и взятия остатка от деления на 65536
        ciphertext += encrypted_char
    return ciphertext

def decrypt(ciphertext, key):
    plaintext = ""
    for i in range(len(ciphertext)):
        char = ciphertext[i]
        key_char = key[i % len(key)]  # Получаем символ ключа по модулю длины ключа
        decrypted_char = chr((ord(char) - ord(key_char)) % 65536)  # Дешифрование символа путем вычитания кодов символов и взятия остатка от деления на 65536
        plaintext += decrypted_char
    return plaintext

def find_possible_plaintext(ciphertext, known_fragment):
    possible_plaintexts = []
    for i in range(len(ciphertext) - len(known_fragment) + 1):
        key = ""
        for j in range(len(known_fragment)):
            key += chr((ord(ciphertext[i+j]) + ord(known_fragment[j])) % 65536)  # Восстановление ключа путем сложения кодов символов шифротекста и фрагмента и взятия ост
        possible_plaintext = decrypt(ciphertext, key)  # Дешифрование шифротекста с использованием восстановленного ключа
        possible_plaintexts.append(possible_plaintext)
    return possible_plaintexts

```

Код

```python
# Пример использования функций
key = "050C177F0E4E37D2941009"
plaintext = "С Новым Годом, друзья!"

# Шифрование
ciphertext = encrypt(plaintext, key)
print("Зашифрованный текст:", ciphertext)

# Дешифрование
decrypted_text = decrypt(ciphertext, key)
print("Расшифрованный текст:", decrypted_text)
```

Код

```
PS C:\Users\nastd\OneDrive\Документы\рудн\VSC> python lab7.py
Зашифрованный текст: iP[illegible]d
Расшифрованный текст: С Новым Годом, друзья!
PS C:\Users\nastd\OneDrive\Документы\рудн\VSC>
```

Результат

1.1 Контрольные вопросы

1. Поясните смысл однократного гаммирования.

Шифрование гаммированием — это шифрование симметричным методом, сущность которого заключается в «наложении» последовательности, сформированной из случайных чисел, на открытый текст. При шифровании гаммированием используется последовательность случайных символов (гамма), которая комбинируется с исходным текстом с использованием операции XOR. 2. Перечислите недостатки однократного гаммирования.

- Если ключ слишком короткий, может возникнуть проблема обратного расчета ключа из шифротекста и открытого текста.
- Если ключ слишком длинный, он может быть сложным для генерации и передачи.
- Если ключ повторяется в процессе шифрования, можно легко восстановить ключ и расшифровать сообщение.

3. Перечислите преимущества однократного гаммирования.

- Если ключ длинный и случайно сгенерирован, алгоритм становится практически непреодолимым для взлома.
- Однократное гаммирование обеспечивает высокую стойкость к пассивному перехвату шифротекста.
- При использовании случайного ключа и правильной реализации, алгоритм обеспечивает конфиденциальность передаваемой информации.

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

Длина открытого текста должна совпадать с длиной ключа, чтобы каждый символ открытого текста мог быть комбинирован с символом ключа для создания символа шифротекста. Если длина открытого текста меньше, лишний ключ может быть неправильно использован или игнорирован. Если длина открытого текста больше, часть ключа может остаться не использованной.

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

В режиме однократного гаммирования используется операция XOR (исключающее ИЛИ). Ее особенность заключается в том, что она позволяет объединять символы открытого текста и ключа таким образом, что если символы совпадают, результат будет 0, если не совпадают - результат будет 1.

6. Как по открытому тексту и ключу получить шифротекст?

Для получения шифротекста по открытому тексту и ключу, каждый символ открытого текста комбинируется с соответствующим символом ключа с использованием операции XOR.

7. Как по открытому тексту и шифротексту получить ключ?

По открытому тексту и шифротексту невозможно получить точный ключ. Шифротекст не содержит информации о ключе, если ключ был случайно сгенерирован и процесс шифрования был правильно выполнен.

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

- Ключ должен быть такой же длины, как и открытый текст, и должен быть случайно сгенерирован.
- Ключ должен использоваться только один раз для каждого открытого текста (не должен повторяться).
- Алгоритм шифрования должен быть выполняться правильно без ошибок или утечек информации.

2 Вывод

Мы освоили на практике применение режима однократного гаммирования.