РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

дисциплина: Информационная безопасность

Студент: Пиняева Анна Андреевна

Группа: НФИбд-02-20

МОСКВА

2023

Цель работы

Освоить на практике применение режима однократного гаммирования

Задача

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.

Ход работы

1. Код программы с комментарием (рис. 1).

Рис. 1 Код:

```python
# Импортируем модули random и string для генерации ключа и работы с символами.
import random
import string


def generate_key(size):
    # Генерируем случайный ключ заданного размера из букв и цифр.
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(size))

def text_to_binary(text):
# Преобразуем текст в его бинарное представление, где каждый символ представлен в виде 8-битного бинарного числа.
    return ''.join(format(ord(char),'08b')for char in text)

def binary_text(binare_str):
# Преобразуем бинарную строку обратно в текст, разбивая бинарные числа на группы по 8 бит и преобразуя их в символы.
    binary_chunks = [binary_str[i:i+8] for i in range(0, len(binary_str),8)]
    return ''.join(chr(int(chunk, 2))for chunk in binary_chunks)

def xor_encrypt (text, key):
    # Выполняем операцию XOR между символами текста и ключа.
    encrypted = [ord(a) ^ ord(b) for a, b in zip(text, key)]
    # Преобразуем полученные числа обратно в символы.
    return ''.join(chr(encrypted_char)for encrypted_char in encrypted)

msg = "С Новым годом, друзья!" # Сообщение

# Генерируем ключ той же длины, что и сообщение
key = generate_key(len(msg))

print ("Ключ:", key)

# Зашифровываем сообщение, используя XOR сгенерированного ключа.
msg2 = xor_encrypt(msg, key)

# Выводим зашифрованное сообщение в двоичном формате.
binary = text_to_binary (msg2)

print ("Зашифрованный текст:", binary)

# Расшифровываем сообщение, используя тот же ключ.
msg3 = xor_encrypt (msg2, key)

print ("Расшифрованный текст:", msg3)
```

N|Solid

---

2. Результат выполнения (рис. 2).

Рис. 2 Результат:

```
Ключ: H3gkDDO6euZ6Lit9QLt8zc
Зашифрованный текст: 1000110100100010011100011110101000101010110001110110100000011111000111001100010110100010101101
0001001011100011011101000000100010001110000010001010101010010000001101100000100011000000111110001000011100011101001
000011010101000010
Расшифрованный текст: С Новым годом, друзья!
```

N|Solid

Контрольные вопросы

1. Гаммирование представляет собой наложение (снятие) на открытые (за- шифрованные) данные последовательности элементов других данных, по- лученной с помощью некоторого криптографического алгоритма, для по- лучения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого пред- ставляет собой известную часть алгоритма шифрования.

2. Недостатки:

Неэффективность ключа: Ключ должен быть длиной сообщения.

Сложная генерация ключа: Генерация случайного ключа сложна.

Один раз использования: Ключи можно использовать только один раз.

Секретность ключа: Ключ должен быть абсолютно секретным.

Недоступность в реальных условиях: Трудности с передачей длинных ключей.

Отсутствие аутентификации: Не предоставляет аутентификацию.

Не гарантирует целостность данных: Не защищает от изменения данных.

Не предотвращает атаки перебора ключа: Уязвим к атакам перебора ключа.

3. Преимущества:

Теоретическая непреодолимость: Обеспечивает абсолютную секретность при правильном использовании случайных ключей.

Совершенная секретность: Невозможно расшифровать сообщение без ключа.

Отсутствие структуры: Зашифрованные данные не имеют паттернов.

Не зависит от алгоритма: Не подвержен атакам на алгоритмы.

4. Суть однократного гаммирования заключается в том, что каждый символ в сообщении “перемешивается” с соответствующим символом в ключе с использованием операции XOR. Если ключ короче сообщения, то операция XOR будет повторяться, и это создаст паттерны, которые могут быть использованы для анализа и расшифровки сообщения. Таким образом, чтобы обеспечить максимальную секретность, длина ключа должна совпадать с длиной сообщения.

5. В режиме однократного гаммирования (One-Time Pad), основной операцией, используемой для шифрования и дешифрования сообщения, является операция XOR (исключающее ИЛИ). Особенности этой операции в контексте однократного гаммирования включают:

1. Побитовая операция: Операция XOR выполняется побитово. Это означает, что каждый бит (0 или 1) в одном числе “исключает” соответствующий бит в другом числе. Результатом операции XOR между двумя битами является 0, если биты одинаковы, и 1, если они различны.

2. Симметричность: Операция XOR симметрична, что означает, что порядок операндов не имеет значения. Например, A XOR B равно B XOR A.

3. Обратимость: Операция XOR обратима. Это означает, что если вы примените XOR дважды с тем же значением, вы вернетесь к исходному значению. Это свойство используется при расшифровке сообщения, так как A XOR B XOR B равно A.

4. Отсутствие структуры: Операция XOR не создает структуру в зашифрованных данных, что делает анализ и попытки расшифровки сложными.

5. Нейтральный элемент: 0 является нейтральным элементом для операции XOR. Это означает, что XOR’ing любого числа с 0 дает исходное число.

В однократном гаммировании, каждый символ открытого текста “перемешивается” (XOR’ится) с соответствующим символом в ключе, что обеспечивает секретность и абсолютную непреодолимость шифра при правильном использовании случайных и одноразовых ключей.

6. Для получения шифротекста из открытого текста и ключа в схеме однократного гаммирования (One-Time Pad), вы используете операцию XOR (исключающее ИЛИ) для каждого символа открытого текста и соответствующего символа в ключе. Процесс выглядит следующим образом:

Предположим, у вас есть открытый текст (представленный в бинарной форме или в виде чисел) и ключ (также представленный в бинарной форме или в виде чисел).

Для каждой пары символов открытого текста и ключа выполняется операция XOR. Например, если открытый текст (в бинарной форме) имеет значение 1101, а ключ имеет значение 1010, то XOR'ing их даст 0111.

Повторяйте этот процесс для каждой пары символов открытого текста и ключа до тех пор, пока не закончите весь открытый текст.

Результатом будет шифротекст, который также будет представлен в виде бинарных чисел или символов, в зависимости от того, как представлены открытый текст и ключ.

7. Получение ключа из открытого текста и шифротекста в схеме однократного гаммирования (One-Time Pad) возможно только в том случае, если известны открытый текст и соответствующий шифротекст, а также если используется тот же ключ для нескольких сообщений или если ключ каким-то образом утек. Обычно ключ генерируется случайным образом и должен быть длиной, равной длине сообщения. Если ключ правильно сгенерирован и не утек, то нельзя вывести ключ, зная только открытый текст и шифротекст. Принцип однократного гаммирования заключается в том, что каждый символ открытого текста "перемешивается" с соответствующим символом ключа с использованием операции XOR. Эта операция обратима, и зная открытый текст и шифротекст, невозможно однозначно восстановить ключ без его знания. Если ключ был использован только один раз и безопасно уничтожен после использования, то он остается секретным. Если же ключ был утерян или скомпрометирован, это может привести к компрометации всей системы шифрования, и в этом случае ключ нельзя будет восстановить на основе только открытого текста и шифротекста.

8. Необходимые и достаточные условия для абсолютной стойкости шифра:

Ключи равной длины: Для каждого возможного ключа существует ровно одно шифрование и дешифрование.

Секретность ключей: Ключи должны быть сгенерированы случайным образом и оставаться секретными. Нет статистических связей между ключами и открытым текстом.

Ключи используются только один раз: Каждый ключ должен использоваться только один раз (одноразовая практика).

Анализ шифротекста бесполезен: Для каждого шифротекста существует бесконечное количество возможных открытых текстов, и анализ шифротекста не дает никакой информации о ключе или оригинальном сообщении.

Выводы

Освоили на практике применение режима однократного гаммирования. Написали прогамму, шифрующую текст. Ответили на контрольные вопросы.

Список используемой литературы

1. Методические материалы курса