Цель работы

Освоить на практике применение режима однократного гаммирования.

Порядок выполнения работы

XOR две строки

Определить функцию с именем xor_strings, которая принимает два строковых параметра s1 и s2.

```
def xor_strings(s1, s2):
    return ''.join(chr(ord(a) ^ ord(b)) for a, b in zip(s1, (b for b in s2 *
(len(s1) // len(s2) + 1))))
```

- zip(s1, (b for b in s2 * (len(s1) // len(s2) + 1))): здесь используется выражение-генератор (b for b in s2 * (len(s1) // len(s2) + 1)) для повторения строки s2 так, чтобы ее длина была как минимум той же длины, что и s1.
- ord(a) ^ ord(b): операция XOR для каждого символа a и b.
- chr(...): преобразовать результат XOR обратно в символы.
- ''.join(...): Объединить все символы в строку и вернуть

Зашифровать открытый текст с помощью одноразового блокнота
Определить функцию с именем encrypt, которая принимает два параметра: открытый текст и ключ.

```
def encrypt(plaintext, key):
    return xor_strings(plaintext, key)
```

Расшифровать зашифрованный текст с помощью одноразового блокнота
Определить функцию с именем decrypt, которая принимает два параметра: зашифрованный текст и ключ.

```
def decrypt(ciphertext, key):
    return xor_strings(ciphertext, key)
```

Учитывая зашифрованный текст и целевой открытый текст, найдите соответствующий ключ.
Определить функцию с именем find_key_for_plaintext, которая принимает два параметра ciphertext и target_plaintext.

```
def find_key_for_plaintext(ciphertext, target_plaintext):
    return xor_strings(ciphertext, target_plaintext)
```

использование
Убедитесь, что приведенный ниже код выполняется только тогда, когда этот файл запускается в качестве основной программы.

```
if __name__ == "__main__":
    # Известная информация
    known_plaintext = "С Новым Годом, друзья!"
```

```
    known_key = "123456745673456789O898"
    #Предположим, у нас есть известный ключ
    # Процесс шифрования
    plaintext_bytes = known_plaintext.encode('utf-8')
    key_bytes = known_key.encode('utf-8')
    ciphertext_bytes = xor_strings(plaintext_bytes.decode('utf-8'),
key_bytes.decode('utf-8')).encode('utf-8')
    ciphertext = ciphertext_bytes.decode('utf-8')
    print(f"ciphertext: {ciphertext}")

    # Процесс расшифровки
    decrypted_text_bytes = xor_strings(ciphertext, key_bytes.decode('utf-
8')).encode('utf-8')
    decrypted_text = decrypted_text_bytes.decode('utf-8')
    print(f"plaintext: {decrypted_text}")

    # Найдите ключ, необходимый для определенного открытого текста
    target_plaintext = "С Новым Годом, друзья!"
    found_key_bytes = xor_strings(ciphertext, target_plaintext).encode('utf-8')
    found_key = found_key_bytes.decode('utf-8')
    print(f"key: {found_key}")
```

Контрольные вопросы

1. Пояснение смысла однократного гаммирования

Однократное гаммирование (one-time pad) — это метод шифрования, при котором текст сообщения (открытый текст) складывается по модулю два (XOR) с случайным ключом такой же длины. Если ключ используется только один раз и хранится в секрете, то данный метод считается абсолютно надёжным.

2. Недостатки однократного гаммирования

- Сложность управления ключами
- Затруднённая дистрибуция ключей
- Необходимость безопасного хранения ключей

3. Преимущества однократного гаммирования

- Абсолютная безопасность (при правильном применении)
- Простота реализации
- Высокая скорость шифрования и дешифрования

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

Для обеспечения безопасности каждый символ открытого текста должен быть зашифрован соответствующим символом ключа. Если длина ключа не совпадает с длиной открытого текста, то шифр становится уязвимым.

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

Используется операция **XOR (исключающее ИЛИ)**.

- Особенности: простота выполнения, обратимость, непредсказуемость результата при использовании случайного ключа.

6. Как по открытому тексту и ключу получить шифротекст?

Шифротекст получают путём применения операции XOR между каждым битом открытого текста и соответствующим битом ключа.

7. Как по открытому тексту и шифротексту получить ключ?

Чтобы восстановить ключ, нужно применить операцию XOR между каждым битом открытого текста и соответствующим битом шифротекста.

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

- Ключ должен быть абсолютно случайным.
- Длина ключа должна быть не меньше длины открытого текста.
- Ключ не должен использоваться для шифрования других сообщений.
- Ключ должен оставаться секретным.

ВЫВОДЫ

Освоил на практике применение режима однократного гаммирования.