Цель работы:

Освоить на практике применение режима однократного гаммирования.

Ход работы:

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Разработаем приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования.

```
In [8]: import random
        import string
        inp = input("Введите строку: ")

        Введите строку: с Новым Годом, друзья!

In [9]: def key_gener(size = 6, chars = string.ascii_letters + string.digits):
            return ''.join(random.choice(chars) for _ in range(size))
        def chan(s):
            return ":".join("{:02x}".format(ord(c)) for c in s)

In [10]: key = key_gener(len(inp))

In [15]: def gammirovanie(inp, key):
             vvod_ascii = [ord(i) for i in inp]
             key_ascii = [ord(i) for i in key]
             enc_str = ''.join(chr(s ^ k) for s, k in zip(vvod_ascii, key_ascii))
             retur enc_str
         def find_truekey(inp, enc_str):
             sm_ascii = [ord(i) for i in inp]
             enc_str_ascii = [ord(i) for i in enc_str]
             true_key = ''.join(chr(s ^ k) for s, k in zip(enc_str_ascii, sm_ascii))
             return true_key
         def unencrypt(enc_str, key):
             enc_str_ascii = [ord(i) for i in enc_str]
             key_ascii = [ord(i) for i in key]
             true_str = ''.join(chr(s ^ k) for s, k in zip(enc_str_ascii, key_ascii))
             return true_str

In [16]: enc_str = gammirovanie(inp, key)

In [17]: new_key = key_gener(len(inp))
         unencrypted_new_key = unencrypt(enc_str, new_key)
         true_key = find_truekey(inp, enc_str)
         unencrypted_true_key = unencrypt(enc_str, true_key)

In [18]: print(f'Закодированная строка: {enc_str}')
         print(f'В шестнадцатиричной системе: {chan(enc_str)}')

         Закодированная строка: гfщЍжЛoшёЏoψЍjzџBзђЄn‼
         В шестнадцатиричной системе: 433:66:449:47c:46b:41b:47b:77:451:40f:47b:471:40d:6a:7a:45f:412:437:452:404:43f:13

In [19]: print(f'Подобранный ключ: {new_key}')
         print(f'Строка расшифрованная ключом: {unencrypted_new_key}')
         print(f'Настоящий ключ: {true_key}')
         print(f'Декодированная строка: {unencrypted_true_key}')

         Подобранный ключ: 9jwf91Dk6hpqAGGT8ZTztK
         Строка расшифрованная ключом: ЊoKЂЪn└ѧѧЋÈь-=ЋЪѫIѼыX
         Настоящий ключ: rFTBYPGWB1OO1FZkRteHp2
         Декодированная строка: с Новым Годом, друзья!
```

Заключение:

В ходе выполнения лабораторной работы я изучил теорию и освоил на практике применение режима однократного гаммирования.

Ответы на контрольные вопросы:

1. Поясните смысл однократного гаммирования.

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического

алгоритма, для получения зашифрованных (открытых) данных.

Однократное гаммирование – это когда каждый символ попарно с символом ключа складываются по модулю 2 (XOR) (обозначается знаком ⊕).

2. Перечислите недостатки однократного гаммирования.

Недостатки: Размер ключевого материала должен совпадать с размером передаваемых сообщений. Также необходимо иметь эффективные процедуры для выработки случайных равновероятных двоичных последовательностей и специальную службу для развоза огромного количества ключей. А ещё, если одну и ту же гамму использовать дважды для разных сообщений, то шифр из совершенно стойкого превращается в «совершенно нестойкий» и допускает дешифрование практически вручную.

3. Перечислите преимущества однократного гаммирования.

Достоинства: С точки зрения теории криптоанализа метод шифрования случайной однократной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях. К достоинствам также можно отнести простоту реализации и удобство применения.

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

Потому что каждый символ открытого текста должен складываться с символом ключа попарно.

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

В режиме однократного гаммирования используется сложение по модулю 2 (XOR) между элементами гаммы и элементами подлежащего сокрытию текста. Особенность заключается в том, что этот алгоритм шифрования является симметричным. Поскольку двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, шифрование и расшифрование выполняется одной и той же программой.

6. Как по открытому тексту и ключу получить шифротекст?

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

$$C_i = P_i \oplus K_i,$$

где C_i — i-й символ получившегося зашифрованного послания, P_i — i-й символ открытого текста, K_i — i-й символ ключа, i = 1, m. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

7. Как по открытому тексту и шифротексту получить ключ?

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с, а именно, обе части равенства необходимо сложить по модулю 2 с P_i:

$$C_i \oplus P_i = P_i \oplus K_i \oplus P_i = K_i,$$

$$K_i = C_i \oplus P_i.$$

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

Необходимые и достаточные условия абсолютной стойкости шифра:

- Полная случайность ключа;
- Равенство длин ключа и открытого текста;
- Однократное использование ключа.