Лабораторная работа №7

Элементы криптографии. Однократное гаммирование

Кекишева Анастасия Дмитриевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить на практике применение режима однократного гаммирования.

2 Задание

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.

3 Теоретическое введение

Гаммирование, или Шифр XOR, — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Последовательность случайных чисел называется гамма-последовательностью и используется для зашифровывания и расшифровывания данных. Суммирование обычно выполняется в каком-либо конечном поле [1].

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование)той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте.

Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком ⊠) между элементами гаммы и элементами подлежащего сокрытию текста. Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же программой.

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила: > Ci = Pi ⊠Ki,

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

Если известны шифротекст и открытый текст, то задача нахождения ключа решается через формулу, а именно, обе части равенства необходимо сложить по модулю 2 с Pi: > Ci ⊠Pi = Pi ⊠Ki ⊠Pi = Ki, > Ki = Ci ⊠Pi.

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов [2].

4 Выполнение лабораторной работы

1. Написала код (рис. 4.1, 4.2), который определяет вид шифротекста при известном ключе и известном открытом тексте, а также определяет ключ.

```
def main(de_text, en_text):   # de - расшифрованный, en - зашифрованный
    dict = {"а": 1, "б": 2, "в": 3, "г": 4, "д": 5, "е": 6, "ё": 7, "ж": 8, "з": 9,
            "и": 10, "й": 11, "к": 12, "л": 13, "м": 14, "н": 15,"о": 16, "п": 17,
            "р": 18, "с": 19, "т": 20, "у": 21, "ф": 22, "х": 23, "ц": 24, "ч": 25,
            "ш": 26, "щ": 27, "ъ": 28, "ы": 29, "ь": 30, "э": 31, "ю": 32, "я": 33,
            " ": 34, ",": 35, "!": 36}
    dict2 = {n: m for m, n in dict.items()}
    digits_de_text = list()
    digits_en_text = list()

    for i in de_text:
        digits_de_text.append(dict[i])
    print("Числа текста: ", digits_de_text)

    for j in en_text:
        digits_en_text.append(dict[j])
    print("Числа зашифрованного текста: ", digits_en_text)

    digits_res = list()
    h = 0

    for i in de_text:
        try:
            a = dict[i] + digits_en_text[h]
        except:
            h = 0
            a = dict[i] + digits_en_text[h]
        if a >= 36:
            a = a % 36
        h += 1
        digits_res.append(a)
    print("Числа шифровки: ", digits_res)
```

Рис. 4.1: Первая часть алгоритма

```
text_en = ""
for i in digits_de_text:
    text_en += dict2[i]
print("Шифровка: ", text_en)

digits = list()
for i in text_en:
    digits.append(dict[i])
h = 0
digits1 = list()
for i in digits:
    a = i - digits_en_text[h]
    if a < 1:
        a = 36 + a
    digits1.append(a)
    h += 1
text_de = ""

for i in digits1:
    text_de += dict2[i]
print("Рассшифровка: ", text_de)
```

Рис. 4.2: Вторая часть алгоритма

Мой код преобразет текст, написанный в нижнем регистре, поэтому первоначальную фразу я сделала таковой, командой lower(). Далее проверила, чтобы текст для шифровки также состоял из такого же количества символов и запустила программу. Результат: ысэндллттбун,ф!ц!ьциг!

```
text = "С Новым Годом, друзья!"
de_text = text.lower()
print(de_text)
```

```
с новым годом, друзья!
```

```
len(de_text)
```

```
22
```

```
en_text = "шнта оамтмтанл пращуты!"
len(en_text)
```

```
22
```

```
main(de_text, en_text)
```

```
Числа текста:  [19, 34, 15, 16, 3, 29, 14, 34, 4, 16, 5, 16, 14, 35, 34, 5, 18, 21, 9, 30, 33, 36]
Числа зашифрованного текста:  [26, 15, 20, 1, 34, 16, 1, 14, 20, 14, 20, 1, 15, 13, 34, 17, 18, 27, 21, 20, 29, 36]
Числа шифровки:  [9, 13, 35, 17, 1, 9, 15, 12, 24, 30, 25, 17, 29, 12, 32, 22, 0, 12, 30, 14, 26, 0]
Шифровка:  с новым годом, друзья!
Рассшифровка:  ысэндллттбун,ф!ц!ьциг!
```

Рис. 4.3: Результат

5 Выводы

Освоила на практике применение режима однократного гаммирования, написав программу, которая определяет вид шифротекста при известном ключе и известном открытом тексте и определяет ключ.

Список литературы

1. Однократное гаммирование [Электронный ресурс]. URL: https://studfile.net/preview/272674/page:7/.
2. Лабораторная работа No 7. Элементы криптографии. Однократное гаммирование [Электронный ресурс]. URL: https://esystem.rudn.ru/pluginfile.php/2090421/mod_resource/content/2/007-lab_crypto-gamma.pdf.