Лабораторная работа №7

Основы Информационной Безопасности

Александрова Ульяна Вадимовна

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

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» является простой, но надёжной схемой шифрования данных. [**course?**]

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

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. Напомним, как работает операция XOR над битами: 0 ⊠ 0 = 0, 0 ⊠ 1 = 1, 1 ⊠ 0 = 1, 1 ⊠ 1 = 0.

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 про- граммой.

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

$$Ci = Pi \boxtimes Ki, \quad (7.1)$$

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 также в соответствии с (7.1), а именно, обе части равенства необходимо сложить по модулю 2 с Pi:

$$Ci \boxtimes Pi = Pi \boxtimes Ki \boxtimes Pi = Ki,$$

$$Ki = Ci \boxtimes Pi.$$

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

К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P.

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.

Рассмотрим пример.

Ключ Центра:

```
05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54
```

Сообщение Центра:

Штирлиц – Вы Герой!!

```
D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C3 E5 F0 EE E9 21 21
```

Зашифрованный текст, находящийся у Мюллера:

```
DD FE FF 8F E5 A6 C1 F2 B9 30 CB D5 02 94 1A 38 E5 5B 51 75
```

Дешифровальщики попробовали ключ:

```
05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 55 F4 D3 07 BB BC 54
```

и получили текст:

```
D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C1 EE EB E2 E0 ED 21
```

Штирлиц - Вы Болван!

Другие ключи дадут лишь новые фразы, пословицы, стихотворные строфы, словом, всевозможные тексты заданной длины.

4 Выполнение лабораторной работы

Так как в задании лабораторной работы не указано, в какой рабочей среде и на каком языке нужно писать программу, я решила написать в самом доступном Jupiter Notebook.

Я начала с разработки функции, которая будет подбирать ключ к открытому тексту по нескольким критериям: - полная случайность ключа; - равенство длин ключа и открытого текста; - однократное использование ключа (рис. 4.1).

```
import random
import string

text = 'С Новым Годом, друзья!'

def key_gen(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.digits+string.ascii_letters)
    return key

len(key_gen(text)) == len(text)
```

```
True
```

```
key = key_gen(text)
print(key)
```

```
omyXcsGVZlDsoUDI9lnezj
```

Рис. 4.1: функция key_gen

Сначала мы создаем переменную с самим открытым текстом **text**, затем внутри функции создаем пустую стринговую переменную, а дальше открываем цикл, где проходим по длине текста (включая пробелы). В цикле складываем случайные значения цифр и шифра. Потом провела проверку, одинакова ли длина у текста и ключа.

Затем я разработала функцию, которая будет производит шифрование (и расшифровку) текста, где мы будем применять сложение по модулю 2 (XOR) всех символов ключа и всех символов текста (рис. 4.2).

```
def shifr(text, key):
    sh_text = ''
    for t,k in zip(text, key):
        xor = ord(t) ^ ord(k) # реализуем умножение по модулю два, при этом переводим элементы в биты
        sh_text += chr(xor)
    return sh_text

sh_text = shifr(text, key)
sh_text
```

```
'юМЄАёиφvщђΨэѓydČѹϴљЩеК'
```

Рис. 4.2: функция shifr

Чтобы не писать новые функции для поиска ключа, мы можем просто воспользоваться функцией **shifr**, которая по сути и сможет найти ключ по фрагменту и шифротекстсу, что следует из этого рассуждения

`Ci ⊠ Pi = Pi ⊠ Ki ⊠ Pi = Ki,`

`Ki = Ci ⊠ Pi.`

Аналогично, для дешифровки по уже новому ключу воспользуемся той же функцией и получим переменную **decode**, которая и должна выводит фрагмент расшифрованного текста (рис. 4.3).

```
new_key = shifr(sh_text, text[15:21]) # находим ключ по той же функции, но меняем аргументы на шифротект и открытый текс
```

```
decode = shifr(new_key, sh_text) # расшифровывем. все работает!
```

```
print('Открытый текст: ', text, '\nИзвестный ключ к открытому тексту: ', key, '\nШифротекст: ', sh_text)
print('\n\nВозможный ключ по шифротексту и фрагменту: ', new_key, '\nРасшифрованный фрагмент: ', decode)
```

```
Открытый текст:  С Новым Годом, друзья!
Известный ключ к открытому тексту:  omyXcsGVZlDsoUDI9lnezj
Шифротекст:  юМЄАёиφvщђΨэѓydČѹϴљЩеК


Возможный ключ по шифротексту и фрагменту:  zЙ'Q□w
Расшифрованный фрагмент:  друзья
```

Рис. 4.3: вывод программы

5 Листинг программы

```
import random
import string

text = 'С Новым Годом, друзья!'

def key_gen(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.digits+string.ascii_letters)
    return key

key = key_gen(text)

def shifr(text, key):
    sh_text = ''
    for t,k in zip(text, key):
        xor = ord(t) ^ ord(k) # реализуем умножение по модулю два, при этом перев
        sh_text += chr(xor)
    return sh_text

sh_text = shifr(text, key)
```

```
new_key = shifr(sh_text, text[15:21])
decode = shifr(new_key, sh_text)

print('Открытый текст: ', text, '\nИзвестный ключ к открытому тексту: ', key, '\n
print('\n\nВозможный ключ по шифротексту и фрагменту: ', new_key, '\nРасшифрованн
```

6 Выводы

Я освоила на практике применение режима однократного гаммирования.

7 Контрольные вопросы

1. Поясните смысл однократного гаммирования.

При однократном гаммировании каждый символ открытого текста гаммируется (XOR) только один раз. **2. Перечислите недостатки однократного гаммирования.**

Частичная уязвимость, значение ключа обновляется каждый раз, что не всегда удобно. **3. Перечислите преимущества однократного гаммирования.**

Простота реализации и использования. **4. Почему длина открытого текста должна совпадать с длиной ключа?**

Так как каждый символ ключа гаммируется с каждым символом текста. **5. Какая операция используется в режиме однократного гаммирования, назовите её особенности?**

XOR, то есть исключающее ИЛИ,то есть сложение по модулю 2. **6. Как по открытому тексту и ключу получить шифротекст?**

Нужно применить XOR к каждому элементу ключа и текста. **7. Как по открытому тексту и шифротексту получить ключ?**

Аналогично, однако точный ключ получить нельзя. **8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?**

- полная случайность ключа; - равенство длин ключа и открытого текста; - однократное использование ключа.