

Лабораторная работа №8

Основы информационной безопасности

Феоктистов Владислав Сергеевич

22 сентября 2022

Российский университет дружбы народов, Москва, Россия

НПМбд-01-19

Элементы криптографии.
Шифрование (кодирование)
различных исходных текстов одним
ключом.

Целью данной работы является освоение на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочесть оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P_1 и P_2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C_1 и C_2 обоих текстов P_1 и P_2 при известном ключе ; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочесть оба текста, не зная ключа и не стремясь его определить.

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных (ключа), чаще всего того же размера (ключ можно зациклить). Под наложением, по сути, подразумевается выполнение операции сложения по модулю 2 (XOR) (обозначаемая знаком \oplus) между элементами гаммы (ключа) и элементами, подлежащих сокрытию. 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 (функцией).

Ход выполнения лабораторной работы

Функция добавления пробелов

Написали функцию добавления пробелов в конце строки до нужного размера строки. Это необходимо, поскольку длины открытых сообщений должны совпадать, иначе их нельзя будет зашифровать одним и тем же ключом гаммирования.

```
def add_endspaces(text, required_length):  
    """  
    Функция добавления пробелов в конце строки до нужного размера.  
    :param text: исходный текст, длину которого нужно увеличить  
    :param required_length: требуемая длина строки  
  
    :return: строку с добавленными пробелами на конце.  
    """  
    if len(text) > required_length:  
        raise Exception('Требуемая длина меньше длины исходного текста!')  
    return text + (' ' * (required_length - len(text)))
```

Figure 1: Функция добавления пробелов

Функция генерации случайной строки

Для удобства написали функцию генерации случайной строки заданной длины *length* с использованием заданных символов *letters*, причем по умолчанию - это латинские символы верхнего и нижнего регистров.

```
def generate_random_str(length, letters=''):
    """
    Функция возвращает строку со случайной последовательностью символов,
    которые указаны в переменной letters. По умолчанию - латинские символы
    верхнего и нижнего регистра.
    :param length: длина желаемой строки
    :param letters: символы, которые будут использовать для создания строки.
    По умолчанию - латинские символы верхнего и нижнего регистра.

    :return: строка последовательности случайных символов, указанных в параметре
    letters.
    """
    if letters == '':
        letters = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
    return ''.join(letters[randint(0, len(letters)-1)] for i in range(length))
```

Figure 2: Функция генерации случайной строки

Функция однократного гаммирования и представления строки

Написали функцию, производящую однократное гаммирование посредством побитового сложения по модулю 2 элементов открытого текста и ключа.

```
def gammimg(text, key):  
    """  
    Функция однократного гаммирования открытого текста по ключу гаммирования.  
    Зашифрованный текст получается путем поэлементного сложения по модулю 2  
    открытого текста и ключа.  
    :param text: открытый текст, подлежащий сокрытию (шифрование)  
    :param key: ключ для однократного гаммирования  
    :return: зашифрованный текст  
    """  
    if len(text) != len(key):  
        raise Exception('Длина шифруемого текста должна совпадать с длиной ключа!')  
    return ''.join(chr(ord(t) ^ ord(k)) for t,k in zip(text, key))
```

Figure 3: Функция однократного гаммирования

Код эмуляции сценария расшифровки злоумышленником двух сообщений

В условии точки входа программы написали небольшой код, эмулирующий ситуацию, когда злоумышленник знает начало первого сообщения и может с помощью него постепенно расшифровать оба сообщения, которые были зашифрованы однократным гаммированием одним ключом.

```

name = "01_01_2023"
P1 = "Решить! В следующем мире появятся новые (математический язык, "Синтез речи""
P2 = "Меня поразило, что в нашем мире никогда не было, а в 2023 году!"

P1_hacked = "Hacked:"
P2_hacked = ""
k = 0
iter, k = k

length1 = len(P1), len(P2)
P1 = end_spaces(P1, length1)
P2 = end_spaces(P2, length1)
K = generate_random_str(length1)

C1 = generate(P1, K)
C2 = generate(P2, K)
C = generate(C1, C2)

while len(P1_hacked) != length1 and len(P2_hacked) != length1:
    if k % 2 == 0:
        P2_hacked_tmp = generate(C1[len(P1_hacked)], P1_hacked)
        print("Успешно перевернули часть строки ciphertext", P2_hacked_tmp)
        if k != prev_k:
            P1_hacked = P1_hacked_prev
            P2_hacked = P2_hacked_tmp
        k += 1
        P1_hacked = P2_hacked_prev
    else:
        P1_hacked_tmp = generate(C1[len(P2_hacked)], P2_hacked)
        print("Успешно перевернули часть строки ciphertext", P1_hacked_tmp)
        if k != prev_k:
            P1_hacked = P1_hacked_prev
            P2_hacked = P2_hacked_prev
        k += 1
        P2_hacked = P1_hacked_prev
    k += 1
    P1_hacked = P2_hacked_prev
    P2_hacked = P1_hacked_prev
    P1_hacked = input("Введите предполагаемое значение для строки ciphertext: ")
    P2_hacked = input("Введите предполагаемое значение для строки ciphertext: ")
    if input("Continue? [y/n]: ") != "n":
        break
    prev_k = k
    k += 1

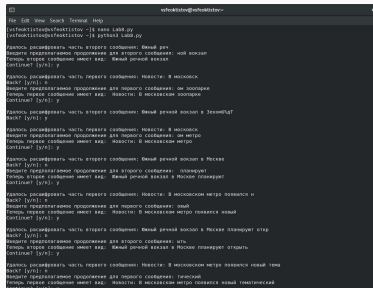
if len(P1_hacked) == length1 and len(P2_hacked) == length1:
    print("The ciphertext was successfully decrypted!")

```

Figure 4: Код эмуляции сценария расшифровки злоумышленником двух сообщений

Запуск программы эмуляции сценария

Как можно видеть, зная лишь только часть одного из сообщений, можно постепенно расшифровать оба сообщения.



```
10) vafekistov@vafekistov:~$  
File Edit View Search Terminal Help  
[vafekistov@vafekistov ~]$ cat L000.py  
[vafekistov@vafekistov ~]$ python3 L000.py  
Далось расшифровать часть второго сообщения: Выйди реч.  
Введите предложение(предложения) для второго сообщения: мой вокзал  
Получу второе сообщение имеет вид: Выйди речей вокзал  
Continue? [y/n]: y  
Далось расшифровать часть первого сообщения: Новости: В московск.  
Выйди [y/n]: n  
Введите предложение(предложения) для первого сообщения: он метро  
Получу первое сообщение имеет вид: Новости: В московском метро  
Continue? [y/n]: y  
Далось расшифровать часть второго сообщения: Выйди речей вокзал в Ленинград  
Выйди [y/n]: y  
Далось расшифровать часть первого сообщения: Новости: В московск.  
Введите предложение(предложения) для первого сообщения: он метро  
Получу первое сообщение имеет вид: Новости: В московском метро  
Continue? [y/n]: y  
Далось расшифровать часть второго сообщения: Выйди речей вокзал в Москва  
Выйди [y/n]: n  
Введите предложение(предложения) для второго сообщения: планируют  
Получу второе сообщение имеет вид: Выйди речей вокзал в Москве планируют  
Continue? [y/n]: y  
Далось расшифровать часть первого сообщения: Новости: В московском метро появился н  
Выйди [y/n]: n  
Введите предложение(предложения) для первого сообщения: оны  
Получу первое сообщение имеет вид: Новости: В московском метро появился новый  
Continue? [y/n]: y  
Далось расшифровать часть второго сообщения: Выйди речей вокзал в Москве планируют отар.  
Выйди [y/n]: n  
Введите предложение(предложения) для второго сообщения: ить  
Получу второе сообщение имеет вид: Выйди речей вокзал в Москве планируют отарить  
Continue? [y/n]: y  
Далось расшифровать часть первого сообщения: Новости: В московском метро появился новый тонн  
Выйди [y/n]: n  
Введите предложение(предложения) для первого сообщения: тический  
Получу первое сообщение имеет вид: Новости: В московском метро появился новый тический  
Continue? [y/n]: y
```

Figure 5: Запуск программы эмуляции сценария

Вывод: использование одного и того же ключа для шифрования нескольких сообщений методов однократного гаммирования - это плохая идея.

В процессе выполнения лабораторной работы освоил на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом; написал программу, позволяющую расшифровать два сообщения, зная только один из них.