

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.

```
if __name__ == '__main__':
    P1 = "Воспитан в английской школе полюбил кино театральные виды "Сибирь здесь""
    P2 = "Маленький человек в Москве планирует отплыть в путь 2023 года"

    P1_hacked = "Воскоче: "
    P2_hacked = ""
    k = 9
    prev_k = k

    length = max(len(P1), len(P2))
    P1 = add_whitespace(P1, length)
    P2 = add_whitespace(P2, length)
    K = generate_random_str(length)

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

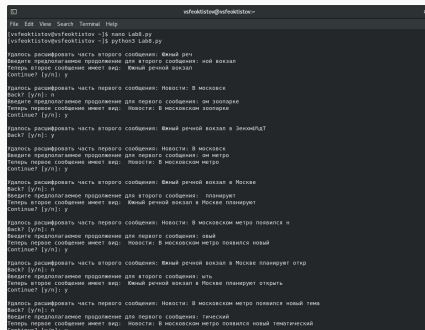
    while len(P1_hacked) != length and len(P2_hacked) != length:
        if k % 2 == 0:
            P2_hacked_tmp = gamming[C[:len(P1_hacked)], P1_hacked]
            print("\nВзломщик пытается узнать второе сообщение: ", P2_hacked_tmp)
            if k != prev_k:
                if input('back? (y/n): ') == 'y':
                    k = 1
                    P1_hacked = P1_hacked_prev
                    continue
                P2_hacked = P2_hacked_tmp
                P2_hacked_prev = P2_hacked
                print("\nВзломщик успешно расшифровал сообщение для второго сообщения: ")
                print("Текущее второе сообщение имеет вид: ", P2_hacked)
            else:
                P1_hacked_tmp = gamming[C[:len(P2_hacked)], P2_hacked]
                print("\nВзломщик пытается узнать первое сообщение: ", P1_hacked_tmp)
                if k != prev_k:
                    if input('back? (y/n): ') == 'y':
                        k = 1
                        P2_hacked = P2_hacked_prev
                        continue
                    P1_hacked = P1_hacked_tmp
                    P1_hacked_prev = P1_hacked
                    print("\nВзломщик успешно расшифровал сообщение для первого сообщения: ")
                    print("Текущее первое сообщение имеет вид: ", P1_hacked)
                if input('continue? (y/n): ') == 'n':
                    break
            prev_k = k
            k += 1

    if len(P1_hacked) == length and len(P2_hacked) == length:
        print("\n\nСудимое было полностью расшифровано!")
```

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.



```

xifeaklistov@xifeaklistov:~$
file Edit View Search Terminal Help
[xifeaklistov@xifeaklistov ~]$ nano lab8.py
[xifeaklistov@xifeaklistov ~]$ python3 lab8.py

Удалось расшифровать часть второго сообщения: Выйди реч.
Введите предполагаемое продолжение для второго сообщения: ной вокзал
Теперь второе сообщение имеет вид: Выйди речной вокзал
Continue? [y/n]: y

Удалось расшифровать часть первого сообщения: Новости: В Москва
Back? [y/n]: n
Введите предполагаемое продолжение для первого сообщения: он зоопарк
Теперь первое сообщение имеет вид: Новости: В московском зоопарке
Continue? [y/n]: y

Удалось расшифровать часть второго сообщения: Выйди речной вокзал в ЗеленодЛТ
Back? [y/n]: y

Удалось расшифровать часть первого сообщения: Новости: В Москва
Введите предполагаемое продолжение для первого сообщения: он метро
Теперь первое сообщение имеет вид: Новости: В московском метро
Continue? [y/n]: y

Удалось расшифровать часть второго сообщения: Выйди речной вокзал в Москва
Back? [y/n]: n
Введите предполагаемое продолжение для второго сообщения: планирует
Теперь второе сообщение имеет вид: Выйди речной вокзал в Москва планирует
Continue? [y/n]: y

Удалось расшифровать часть первого сообщения: Новости: В московском метро появился н
Back? [y/n]: n
Введите предполагаемое продолжение для первого сообщения: аый
Теперь первое сообщение имеет вид: Новости: В московском метро появился новый
Continue? [y/n]: y

Удалось расшифровать часть второго сообщения: Выйди речной вокзал в Москва планирует откр
Back? [y/n]: n
Введите предполагаемое продолжение для второго сообщения: ить
Теперь второе сообщение имеет вид: Выйди речной вокзал в Москва планирует открыть
Continue? [y/n]: y

Удалось расшифровать часть первого сообщения: Новости: В московском метро появился новый тема
Back? [y/n]: n
Введите предполагаемое продолжение для первого сообщения: тический
Теперь первое сообщение имеет вид: Новости: В московском метро появился новый тематический
Continue? [y/n]: y

```

Figure 5: Запуск программы эмуляции сценария

Вывод: использование одного и того же ключа для шифрования нескольких сообщений

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.