

Презентация по лабораторной работе №8.

Основы информационной безопасности.

Рогожина Н.А.

9 мая 2024

Российский университет дружбы народов, Москва, Россия

Информация

- Рогожина Надежда Александровна
- Студентка 2го курса, НКАбд-02-22
- Компьютерные и информационные науки
- Российский университет дружбы народов
- Github

Вводная часть

Освоить 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, 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 \oplus) 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.

Выполнение

```
def encrypt(p1, p2, key=0):  
    print(f'P1: {p1}')    print(f'P2: {p2}')  
    hex_p1 = []  
    hex_p2 = []
```

```
for i in range(len(p1)):
    hex_p1.append(p1[i].encode('cp1251').hex())
    hex_p2.append(p2[i].encode('cp1251').hex())

print(f'Hex P1: {" ".join(hex_p1)}')
print(f'Hex P2: {" ".join(hex_p2)}')
```

```
if key == 0:
    k = np.random.randint(0, 255, len(p1))
    key = [hex(i)[2:] for i in k]
print(f'Hex key: ', *key)

hex_c1 = []
hex_c2 = []

for i in range(len(hex_p1)):
    hex_c1.append('{:02x}'.format(int(key[i], 16)^int(hex_p1[i], 16)))
    hex_c2.append('{:02x}'.format(int(key[i], 16)^int(hex_p2[i], 16)))
```

```
c1 = bytearray.fromhex('').join(hex_c1)).decode('cp1251')
c2 = bytearray.fromhex('').join(hex_c2)).decode('cp1251')

print(f'C1: {c1}')
print(f'C2: {c2}')
return key, c1, c2
```

```
def decode(c1, c2, p1):  
    print(f'C1: {c1}')
```

print(f'C2: {c2}')

print(f'P1: {p1}')


```
hex_c1 = []  
hex_c2 = []  
hex_p1 = []
```

```
for i in range(len(p1)):
    hex_c1.append(c1[i].encode('cp1251').hex())
    hex_c2.append(c2[i].encode('cp1251').hex())
    hex_p1.append(p1[i].encode('cp1251').hex())

print(f'Hex C1: {" ".join(hex_p1)}')
print(f'Hex C2: {" ".join(hex_c2)}')
print(f'Hex P1: {" ".join(hex_p1)}')
```

```
hex_p2 = []

for i in range(len(hex_p1)):
    hex_p2.append('{:02x}'.format(int(hex_c1[i], 16)^int(hex_c2[i], 16)))

print(f'Hex P2: {" ".join(hex_p2)}')

p2 = bytearray.fromhex(''.join(hex_p2)).decode('cp1251')

print(f'P2: {p2}')
```


Пример работы encode

```
In [8]: key, c1, c2 = encrypt(p1, p2)
```

P1: encode

P2: decode

Hex P1: 65 6e 63 6f 64 65

Hex P2: 64 65 63 6f 64 65

Hex key: 7f 3e 24 3d ac 73

C1: PGRM

C2: RM

```
In [9]: encrypt(c1, c2, key)
```

P1: PGRM

P2: RM

Hex P1: 1a 50 47 52 c8 16

Hex P2: 1b 5b 47 52 c8 16

Hex key: 7f 3e 24 3d ac 73

C1: encode

C2: decode

```
Out[9]: (['7f', '3e', '24', '3d', 'ac', '73'], 'encode', 'decode')
```

Рис. 1: encode

```
In [15]: decode(c1, c2, p1)

C1: PGRM
C2: RM
P1: encode
Hex C1: 65 6e 63 6f 64 65
Hex C2: 1b 5b 47 52 c8 16
Hex P1: 65 6e 63 6f 64 65
Hex P2: 64 65 63 6f 64 65
P2: decode
```

Рис. 2: decode

Выводы

В ходе работы у нас получилось освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Спасибо за внимание!
