Защита лабораторной работы № 8. Элементы криптографии. Элементы криптографии. Шифрование (кодирование) различных исходных текстов одним ключом

Смирнова Мария
НФИбд-01-18

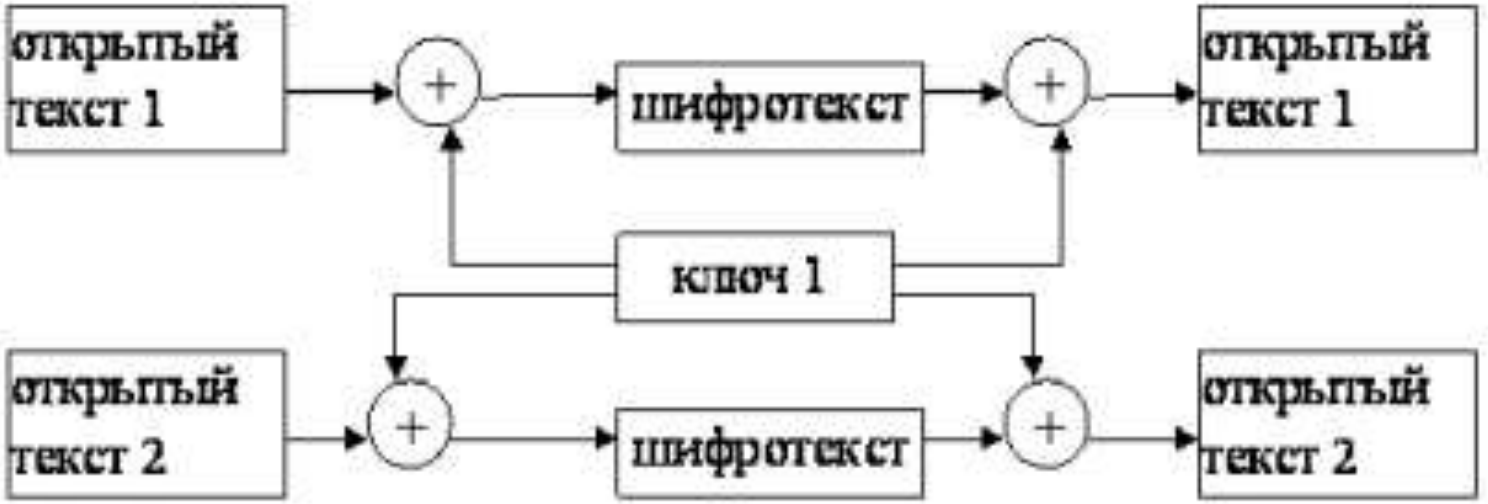
Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

Теоретические сведения

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. Режим шифрования однократного гаммирования одним ключом двух видов открытого текста реализуется в соответствии со схемой, приведённой на рисунке (рис. -@fig:001)

Теоретические сведения



Общая схема шифрования двух различных текстов одним ключом

Задание

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочитать оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты P 1 и P 2 в режиме однократного гаммирования. Приложение должно определить вид шифротекстов C1 и C2 обоих текстов P1 и P2 при известном ключе; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

Выполнение лабораторной работы

Ввод [1]:
```
import numpy as np
import sys
import operator as op
```

Ввод [2]:
```
def keygen(text):
    b = np.random.randint(0, 255, len(text))
    key = [hex(i)[2:] for i in b]
    return key
```

Подключение библиотек и генерация ключа

Ввод [3]:
```
p1 = "НаВашисходящийот1204"
p2 = "ВСеверныйфилиалБанка"
```

Тексты P1 и P2

Ввод [7]:
```
def coder(p1,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"P1:, {hex_p1}")
    print(f"P2:, {hex_p2}")
    key = keygen(p1)
    print(f"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)))
    print(f"C1_h:, {hex_c1}")
    print(f"C2_h:, {hex_c2}")
    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
```

Ввод [8]:
```
key, c1, c2 = coder(p1, p2)
```

```
P1:, ['cd', 'e0', 'c2', 'e0', 'f8', 'e8', 'f1', 'f5', 'ee', 'e4', 'ff', 'f9', 'e8', 'e9', 'e
e', 'f2', '31', '32', '30', '34']
P2:, ['c2', 'd1', 'e5', 'e2', 'e5', 'f0', 'ed', 'fb', 'e9', 'f4', 'e8', 'eb', 'e8', 'e0', 'e
b', 'c1', 'e0', 'ed', 'ea', 'e0']
Key: ['68', '61', '63', '57', '14', 'ae', '8b', 'a', '7a', '7e', 'a7', 'b8', 'a2', '4e', 'c9
', 'fe', '73', '45', 'e3', '9']
C1_h:, ['a5', '81', 'a1', 'b7', 'ec', '46', '7a', 'ff', '94', '9a', '58', '41', '4a', 'a7',
'27', '0c', '42', '77', 'd3', '3d']
C2_h:, ['aa', 'b0', '86', 'b5', 'f1', '5e', '66', 'f1', '93', '8a', '4f', '53', '4a', 'ae',
'22', '3f', '93', 'a8', '09', 'e9']
C1:, ЃЃЎ·мFzя"аXAJ§'BwУ»
C2:, €°†µc^fc"ЊOSJ®°?"Ё й
```

Шифрование

Ввод [17]:
```
decoder(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"C1_h:, {hex_c1}")
print(f"C2_h:, {hex_c2}")
print(f"P1_h:, {hex_p1}")
hex_p2 = []
for i in range(len(p1)):
    hex_p2.append("{:02x}".format(int(hex_c1[i],16) ^ int(hex_c2[i],16) ^ int(hex_p1[i],16)))
print(f"P2_h:, {hex_p2}")
p2 = bytearray.fromhex("".join(hex_p2)).decode("cp1251")
print(f"P2:, {p2}")
return p1, p2
```

Ввод [18]:
```
p1, p2 = decoder(c1, c2, p1)
```

```
C1:, ЃЃЎ·мFzя"љXAJ§'ВwУ=
C2:, Є°†µc^fc"ЉOSJ®"?"Ё й
P1:, НаВашисходящийот1204
C1_h:, ['a5', '81', 'a1', 'b7', 'ec', '46', '7a', 'ff', '94', '9a', '58', '41', '4a', 'a7',
'27', '0c', '42', '77', 'd3', '3d']
C2_h:, ['aa', 'b0', '86', 'b5', 'f1', '5e', '66', 'f1', '93', '8a', '4f', '53', '4a', 'ae',
'22', '3f', '93', 'a8', '09', 'e9']
P1_h:, ['cd', 'e0', 'c2', 'e0', 'f8', 'e8', 'f1', 'f5', 'ee', 'e4', 'ff', 'f9', 'e8', 'e9',
'ee', 'f2', '31', '32', '30', '34']
P2_h:, ['c2', 'd1', 'e5', 'e2', 'e5', 'f0', 'ed', 'fb', 'e9', 'f4', 'e8', 'eb', 'e8', 'e0',
'eb', 'c1', 'e0', 'ed', 'ea', 'e0']
P2:, ВСеверныйфилиалБанка
```

Дешифрование

Контрольные вопросы

1. Как, зная один из текстов (P1 или P2), определить другой, не зная при этом ключа?

Это наглядно показано в 4 пункте работы.

Контрольные вопросы

2. Что будет при повторном использовании ключа при шифровании текста?

Исходный текст может быть восстановлен с помощью статистического анализа двух вариантов зашифрованного текста. Важнейшим правилом криптозащиты является достаточно частая смена ключей. Причем частота может определяться исходя из длительности использования ключа или исходя из объема зашифрованного текста.

Контрольные вопросы

3. Как реализуется режим шифрования однократного гаммирования одним ключом двух открытых текстов?

Ответ представлен на схеме (рис. -@fig:006) Общая схема шифрования двух различных текстов одним ключом

Контрольные вопросы

4. Перечислите недостатки шифрования одним ключом двух открытых текстов.

В алгоритм сложнее внести изменения и более длинные ключи. Так же снижается безопасность обоих текстов.

Контрольные вопросы

5. Перечислите преимущества шифрования одним ключом двух открытых текстов.

Небольшое число ключей для передачи, простота алгоритма, удобство для обеих сторон.

Выводы

В процессе выполнения лабораторной работы мы освоили на практике применение режима однократного гаммирования одним ключом на языке python.