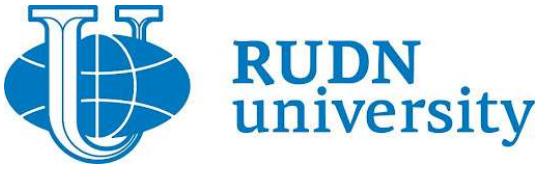
# Лабораторная работа №7

Однократное гаммирование

Доборщук В.В., НФИбд-01-18




11 декабря 2021

# Содержание

# Список иллюстраций

# Цель работы

Освоить на практике применение режима однократного гаммирования.

# Выполнение лабораторной работы

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 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

## Реализация функционала

Создали дополнительную функцию для генерации случайного ключа:

```
def gen_key(text):
    rn = np.random.randint(0, 255, len(text))
    key = [hex(e)[2:] for e in rn]
    return key
```

Реализована следующая функция для определения вида шифротекста при известных ключе и открытом тексте. На вход подается строка, после чего она переводится в 16-ричную систему, после чего в программе генерируется ключ случайным образом. С его помощью получается зашифрованное сообщение в шестнадцатеричной системе счисления.

```
def crypt_message(open_text, key):
    print(f"Open Text: {open_text}")
    hex_open_text = []
    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    print("Hex Open Text: ", *hex_open_text)

    print("Key: ", *key)
    hex_crypted_text = []
    for i in range(len(hex_open_text)):
```

```
        hex_crypted_text.append("{:02x}".format(int(key[i],
↪   16)^int(hex_open_text[i], 16)))

    print("Hex Crypted Text: ", *hex_crypted_text)
    crypted_text =
↪   bytearray.fromhex("".join(hex_crypted_text)).decode("cp1251")
    print(f"Crypted Text: {crypted_text}")

    return crypted_text
```

Далее определяем ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста. Функция получает на вход открытый текст и зашифрованный, после чего преобразует строки в 16-ричный формат и выполняет операцию сложения по модулю 2 (XOR).

```
def find_key(open_text, crypted_text):
    print(f"Open Text: {open_text}\nCrypted Text: {crypted_text}")
    hex_open_text = []
    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    hex_crypted_text = []
    for ch in crypted_text:
        hex_crypted_text.append(ch.encode("cp1251").hex())

    print("Hex Open Text: ", *hex_open_text)
    print("Hex Crypted Text: ", *hex_crypted_text)

    key = [hex(int(i,16)^int(j,16))[2:] for (i,j) in zip(hex_open_text,
↪   hex_crypted_text)]
    print("Key ", *key)

    return key
```

In [1]:
```
import numpy as np
```

In [24]:
```
def gen_key(text):
    rn = np.random.randint(0, 255, len(text))
    key = [hex(e)[2:] for e in rn]
    return key
```

In [25]:
```
def crypt_message(open_text, key):
    print(f"Open Text: {open_text}")
    hex_open_text = []
    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    print("Hex Open Text: ", *hex_open_text)

    print("Key: ", *key)
    hex_crypted_text = []
    for i in range(len(hex_open_text)):
        hex_crypted_text.append("{:02x}".format(int(key[i], 16)^int(hex_open_text[i], 16)))

    print("Hex Crypted Text: ", *hex_crypted_text)
    crypted_text = bytearray.fromhex("".join(hex_crypted_text)).decode("cp1251")
    print(f"Crypted Text: {crypted_text}")

    return crypted_text
```

In [ ]:
```
def find_key(open_text, crypted_text):
    print(f"Open Text: {open_text}\nCrypted Text: {crypted_text}")
    hex_open_text = []
    for ch in open_text:
        hex_open_text.append(ch.encode("cp1251").hex())

    hex_crypted_text = []
    for ch in crypted_text:
        hex_crypted_text.append(ch.encode("cp1251").hex())

    print("Hex Open Text: ", *hex_open_text)
    print("Hex Crypted Text: ", *hex_crypted_text)

    key = [hex(int(i,16)^int(j,16))[2:] for (i,j) in zip(hex_open_text, hex_crypted_text)]
    print("Key ", *key)

    return key
```

**Рис. 1:** Реализованные функции

С помощью реализованного функционала проверяем работоспособность нашего приложения:

```
In [26]: raw = "С Новым Годом, друзья!"
         key1 = gen_key(raw)

In [27]: ct = crypt_message(raw, key1)

         Open Text: С Новым Годом, друзья!
         Hex Open Text:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
         Key:  56 aa ae aa 11 4b 5c a2 b8 95 3c 3b 82 f f6 33 21 70 e3 83 2f 6e
         Hex Crypted Text:  87 8a 63 44 f3 b0 b0 82 7b 7b d8 d5 6e 23 d6 d7 d1 83 04 7f d0 4f
         Crypted Text: ‡ЉcDy°°,{{ШXn#ЦЧСf□□PO

In [29]: dct = crypt_message(ct, key1)

         Open Text: ‡ЉcDy°°,{{ШXn#ЦЧСf□□PO
         Hex Open Text:  87 8a 63 44 f3 b0 b0 82 7b 7b d8 d5 6e 23 d6 d7 d1 83 04 7f d0 4f
         Key:  56 aa ae aa 11 4b 5c a2 b8 95 3c 3b 82 f f6 33 21 70 e3 83 2f 6e
         Hex Crypted Text:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
         Crypted Text: С Новым Годом, друзья!
```

**Рис. 2:** Проверка шифрования

Все корректно отрабатывает с одним и тем же ключом.

## Проверка ключа

В конце проверяем совпадают ли полученный ключ для идентичного и неидентичного текста и начальный ключ.

```
In [32]: key2 = find_key(raw, ct)

         Open Text: С Новым Годом, друзья!
         Crypted Text: ‡ЉcDy°°,{{ШXn#ЦЧСf□□PO
         Hex Open Text:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21
         Hex Crypted Text:  87 8a 63 44 f3 b0 b0 82 7b 7b d8 d5 6e 23 d6 d7 d1 83 04 7f d0 4f
         Key  56 aa ae aa 11 4b 5c a2 b8 95 3c 3b 82 f f6 33 21 70 e3 83 2f 6e

In [34]: key1 == key2
Out[34]: True

In [35]: key3 = find_key("С Новым Годом, друзья?", ct)

         Open Text: С Новым Годом, друзья?
         Crypted Text: ‡ЉcDy°°,{{ШXn#ЦЧСf□□PO
         Hex Open Text:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 3f
         Hex Crypted Text:  87 8a 63 44 f3 b0 b0 82 7b 7b d8 d5 6e 23 d6 d7 d1 83 04 7f d0 4f
         Key  56 aa ae aa 11 4b 5c a2 b8 95 3c 3b 82 f f6 33 21 70 e3 83 2f 70

In [36]: key1 == key3
Out[36]: False
```

**Рис. 3:** Сравнение ключей

Получили, что для идентичного текста ключи совпадают, а для неидентичного, соответственно, не совпадают.

## Контрольные вопросы

1. Поясните смысл однократного гаммирования.

**Гаммирование** – выполнение операции XOR между элементами гаммы и элементами подлежащего сокрытию текста. 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.

2. Перечислите недостатки однократного гаммирования.

Абсолютная стойкость шифра доказана только для случая, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения.

3. Перечислите преимущества однократного гаммирования.

— Во-первых, такой способ симметричен, т.е. двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.

— Во-вторых, шифрование и расшифрование может быть выполнено одной и той же программой. Наконец, 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.

4. Почему длина открытого текста должна совпадать с длиной ключа?

Если ключ короче текста, то операция XOR будет применена не ко всем элементам и конец сообщения будет не закодирован. Если ключ будет длиннее, то появится неоднозначность декодирования.

5. Какая операция используется в режиме однократного гаммирования, назовите её особенности?

Наложение гаммы по сути представляет собой выполнение побитовой операции сложения по модулю 2, т.е. мы должны сложить каждый элемент гаммы с соответствующим элементом ключа. Данная операция является симметричной, так как

прибавление одной и той же величины по модулю 2 восстанавливает исходное значение.

6. Как по открытому тексту и ключу получить шифротекст?

В таком случае задача сводится к правилу:

$$C_i = P_i \bigoplus K_i$$

т.е. мы поэлементно получаем символы зашифрованного сообщения, применяя операцию исключающего или к соответствующим элементам ключа и открытого текста.

7. Как по открытому тексту и шифротексту получить ключ?

Подобная задача решается путем применения операции исключающего или к последовательностям символов зашифрованного и открытого сообщений:

$$K_i = P_i \bigoplus C_i$$

8. В чем заключаются необходимые и достаточные условия абсолютной стойкости шифра?

Необходимые и достаточные условия абсолютной стойкости шифра:

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

## Выводы

Мы изучили на практике применение режима однократного гаммирования.