

## Приложение 1

### Ручное шифрование

#### Определяю открытый текст для шифрования

```
In [1]: # Возьму для шифрования условную строку

open_text = "evgeny"

# Представлю её в байтах, в целочисленном виде, в битах

open_text_bytes = open_text.encode()
open_text_int = int.from_bytes(open_text_bytes, byteorder="big", signed=False)
open_text_bin = bin(open_text_int)[2:]

import json

print(json.dumps({
    "open_text": open_text,
    "open_text_bytes": str(open_text_bytes),
    "open_text_int": open_text_int,
    "open_text_bin": open_text_bin
}, indent=4))

{
  "open_text": "evgeny",
  "open_text_bytes": "b'evgeny'",
  "open_text_int": 111559215246969,
  "open_text_bin": "110010101110110011001110110010101101110011110
01"
}
```

#### Сгенерирую ключевую пару

```
In [2]: # Сгенерирую вручную открытый ключ длиной 8 бит по модулю длиной 16

# Шаг 1. Придумаю параметры p и q длиной 7 и 9 бит

p = 71
q = 277

len(bin(p)[2:]), len(bin(q)[2:])
```

Out[2]: (7, 9)

In [3]: *# Шаг 2. Вычислю  $n$ ,  $\phi(n)$*

```
n = p * q
phi = (p - 1)*(q - 1)

n, phi
```

Out[3]: (19667, 19320)

In [4]: *# Шаг 3. Придумаю  $e$  (экспонента зашифрования)*

```
e = 223

assert len(bin(e)[2:]) == 8
```

In [5]: *# Шаг 4. Вычислю значение закрытого ключа*

```
d = pow(e, -1, phi)
print(d)
```

18367

In [6]: *# Шаг 5. Проверю правильность вычислений*

```
import random
x = random.randrange(2, 99)
x_encrypted = pow(x, e, n)
x_decrypted = pow(x_encrypted, d, n)

bool(x == x_decrypted)
```

Out[6]: True

### Зашифрую текст

In [7]: *# Вычислю размер блока*

```
import math

block_size = math.log(n, 2)

# Округлю его вниз

block_size = int(block_size)
block_size
```

Out[7]: 14

In [8]: *# Разделю открытый текст на блоки*  
*# Получится 3 полных блока и один не полный*

```
len(open_text_bin) / block_size
```

Out[8]: 3.357142857142857

In [9]: *# Раздроблю открытый текст на блоки с конца*

```
block_4 = "11001"  
block_3 = "01011101100110"  
block_2 = "01110110010101"  
block_1 = "10111001111001"
```

*# Превращу биты в целочисленные значения и зашифрую*

```
block_4_int = int(block_4, 2)  
block_3_int = int(block_3, 2)  
block_2_int = int(block_2, 2)  
block_1_int = int(block_1, 2)
```

```
block_4_int_enc = block_4_int ** e % n  
block_3_int_enc = block_3_int ** e % n  
block_2_int_enc = block_2_int ** e % n  
block_1_int_enc = block_1_int ** e % n
```

*# Посмотрим что получилось*

```
block_4_int_enc, block_3_int_enc, block_2_int_enc, block_1_int_enc
```

Out[9]: (16690, 17436, 9865, 13431)

In [10]: *# Теперь зашифрованные блоки преобразуем в биты*

```
block_4_bin_enc = bin(block_4_int_enc)[2:]  
block_3_bin_enc = bin(block_3_int_enc)[2:]  
block_2_bin_enc = bin(block_2_int_enc)[2:]  
block_1_bin_enc = bin(block_1_int_enc)[2:]
```

*# Посмотрим что получилось*

```
block_4_bin_enc, block_3_bin_enc, block_2_bin_enc, block_1_bin_enc
```

Out[10]: ('100000100110010', '100010000011100', '10011010001001', '11010001110111')

In [11]: *# Проверим длину блоков*

```
len(block_4_bin_enc), len(block_3_bin_enc), len(block_2_bin_enc), len(block_1_bin_enc)
```

Out[11]: (15, 15, 14, 14)

```
In [12]: # Блок 1 имеют длину больше чем остальные.
# Выворняем до одной длины добавив блоками меньшего размера
# по незначащему нулю и увеличив общий размер блока до 15

block_size_full = block_size + 1

block_2_bin_enc = block_2_bin_enc.zfill(block_size_full)
block_1_bin_enc = block_1_bin_enc.zfill(block_size_full)

# Еще раз проверим длину блоков

len(block_4_bin_enc), len(block_3_bin_enc), len(block_2_bin_enc), len
```

Out[12]: (15, 15, 15, 15)

```
In [13]: # Преобразую биты в байты.

# Сначала сложим все биты в одну большую строку

encrypted_text_bin = block_4_bin_enc + block_3_bin_enc + block_2_bin

# Посчитаем количество байтов

bytes_count = math.ceil(len(encrypted_text_bin) / 8)

# И наконец превратим биты в байты и запишем в файл

encrypted_text_bytes = int.to_bytes(int(encrypted_text_bin, 2), length=bytes_count, byteorder='big')

with open("encrypted_text.txt", "wb") as file:
    file.write(encrypted_text_bytes)

# Прочитаем зашифрованные байты

!cat encrypted_text.txt
```

✓Q👁️D💎w

```
In [14]: # Расшифруем зашифрованный текст

# Преобразуем байты в биты

encrypted_text_bin = bin(int.from_bytes(encrypted_text_bytes, byteorder='big')).replace('0b', '')
encrypted_text_bin
```

Out[14]: '100000100110010100010000011100010011010001001011010001110111'

```
In [15]: # Измерим длину битовой строки

len(encrypted_text_bin)
```

Out[15]: 60

```
In [16]: # Делится ли на нашу увеличенную длину блока нацело?

len(encrypted_text_bin) / block_size_full
```

Out[16]: 4.0

```
In [17]: # Да, делится нацело!
# Значит идём по блокам и расшифровываем. Для этого
# нужно преобразовать каждый блок в целочисл значение
# затем возвести в экспоненту расшифрования d по модулю n

encrypted_block_4 = "100000100110010"
encrypted_block_3 = "100010000011100"
encrypted_block_2 = "010011010001001"
encrypted_block_1 = "011010001110111"

encrypted_block_4_int = int(encrypted_block_4, 2)
encrypted_block_3_int = int(encrypted_block_3, 2)
encrypted_block_2_int = int(encrypted_block_2, 2)
encrypted_block_1_int = int(encrypted_block_1, 2)

decrypted_block_4_int = encrypted_block_4_int ** d % n
decrypted_block_3_int = encrypted_block_3_int ** d % n
decrypted_block_2_int = encrypted_block_2_int ** d % n
decrypted_block_1_int = encrypted_block_1_int ** d % n

decrypted_block_4_int, decrypted_block_3_int, decrypted_block_2_int,
```

```
Out[17]: (25, 5990, 7573, 11897)
```

```
In [18]: # Если я правильно расшифровал то целочисленные значения
# расшифрованных блоков и блоков открытого текста до шифрования
# должны совпасть

bool(
    (decrypted_block_4_int, decrypted_block_3_int, decrypted_block_2_int,
     decrypted_block_1_int) == (block_4_int, block_3_int, block_2_int, block_1_int)
)
```

```
Out[18]: True
```

```
In [19]: # Для наглядности продолжу преобразования до получения исходной строки
# добавляю нолики в начале блока до исходного размера блока

decrypted_block_4_bin = bin(decrypted_block_4_int)[2:].zfill(block_size)
decrypted_block_3_bin = bin(decrypted_block_3_int)[2:].zfill(block_size)
decrypted_block_2_bin = bin(decrypted_block_2_int)[2:].zfill(block_size)
decrypted_block_1_bin = bin(decrypted_block_1_int)[2:].zfill(block_size)

decrypted_bin = decrypted_block_4_bin + decrypted_block_3_bin + decrypted_block_2_bin + decrypted_block_1_bin
decrypted_int = int(decrypted_bin, 2)
int.to_bytes(decrypted_int, 6, byteorder="big", signed=False)
```

```
Out[19]: b'evgeny'
```

```
In [20]: # Вуаля!
```

```
In [21]: # Теперь натравим программу на ту же строку

# Создам небольшого размера пару ключей RSA
# Видим пару ключей с префиксом manual в репозитории

from keygen import KeyGenerator

KeyGenerator(8, 8, "manual")
!ls -l
```

```
total 2808
-rw-r--r-- 1 evgeny evgeny 431919 Apr 25 19:45 analyze.ipynb
-rw-r--r-- 1 evgeny evgeny 634672 Apr 25 17:07 analyze.pdf
-rw-r--r-- 1 evgeny evgeny  87080 Apr 25 17:06 demo.ipynb
-rw-r--r-- 1 evgeny evgeny 314541 Apr 25 17:08 demo.pdf
-rw-r--r-- 1 evgeny evgeny      8 Apr 26 17:16 encrypted_text.txt
-rw-r--r-- 1 evgeny evgeny  35117 Apr 23 16:57 image.png
-rw-r--r-- 1 evgeny evgeny   4172 Apr 26 17:00 keygen.py
-rw-r--r-- 1 evgeny evgeny  24080 Apr 26 17:15 manual.ipynb
-rw-r--r-- 1 evgeny evgeny    11 Apr 26 17:16 manual.public
-rw-r--r-- 1 evgeny evgeny    13 Apr 26 17:16 manual.secret
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 17:06 mayday_decrypted.png
-rw-r--r-- 1 evgeny evgeny 328461 Apr 25 17:06 mayday_encrypted.png
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 16:42 mayday.png
-rw-r--r-- 1 evgeny evgeny   408 Apr 20 15:36 open.txt
drwxr-xr-x 2 evgeny evgeny   4096 Apr 26 17:09 __pycache__
-rw-r--r-- 1 evgeny evgeny 339695 Apr 26 15:24 report.docx
-rw-r--r-- 1 evgeny evgeny   5100 Apr 25 19:02 rsa.py
-rw-r--r-- 1 evgeny evgeny    13 Apr 25 17:06 secret.crack
-rw-r--r-- 1 evgeny evgeny    11 Apr 25 17:06 small.public
-rw-r--r-- 1 evgeny evgeny    13 Apr 25 17:06 small.secret
drwxr-xr-x 6 evgeny evgeny   4096 Apr 15 15:18 venv
```

```
In [22]: # Создам объект класса RSA для зашифрования и расшифрования

from rsa import RSA

rsa = RSA("manual.public", "manual.secret")
rsa
```

```
Out[22]: <rsa.RSA at 0x7f152017c750>
```

In [23]: *# Создам файл с текстов "evgeny"*

```
!echo "evgeny" >> manual_open.txt
!ls -l
```

```
total 2812
-rw-r--r-- 1 evgeny evgeny 431919 Apr 25 19:45 analyze.ipynb
-rw-r--r-- 1 evgeny evgeny 634672 Apr 25 17:07 analyze.pdf
-rw-r--r-- 1 evgeny evgeny 87080 Apr 25 17:06 demo.ipynb
-rw-r--r-- 1 evgeny evgeny 314541 Apr 25 17:08 demo.pdf
-rw-r--r-- 1 evgeny evgeny 8 Apr 26 17:16 encrypted_text.txt
-rw-r--r-- 1 evgeny evgeny 35117 Apr 23 16:57 image.png
-rw-r--r-- 1 evgeny evgeny 4172 Apr 26 17:00 keygen.py
-rw-r--r-- 1 evgeny evgeny 24080 Apr 26 17:15 manual.ipynb
-rw-r--r-- 1 evgeny evgeny 7 Apr 26 17:16 manual_open.txt
-rw-r--r-- 1 evgeny evgeny 11 Apr 26 17:16 manual.public
-rw-r--r-- 1 evgeny evgeny 13 Apr 26 17:16 manual.secret
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 17:06 mayday_decrypted.png
-rw-r--r-- 1 evgeny evgeny 328461 Apr 25 17:06 mayday_encrypted.png
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 16:42 mayday.png
-rw-r--r-- 1 evgeny evgeny 408 Apr 20 15:36 open.txt
drwxr-xr-x 2 evgeny evgeny 4096 Apr 26 17:09 __pycache__
-rw-r--r-- 1 evgeny evgeny 339695 Apr 26 15:24 report.docx
-rw-r--r-- 1 evgeny evgeny 5100 Apr 25 19:02 rsa.py
-rw-r--r-- 1 evgeny evgeny 13 Apr 25 17:06 secret.crack
-rw-r--r-- 1 evgeny evgeny 11 Apr 25 17:06 small.public
-rw-r--r-- 1 evgeny evgeny 13 Apr 25 17:06 small.secret
drwxr-xr-x 6 evgeny evgeny 4096 Apr 15 15:18 venv
```

In [24]: *# Зашифрую*

```
rsa.encrypt("manual_open.txt", "manual_encrypted.txt")
!ls -l
```

```
total 2816
-rw-r--r-- 1 evgeny evgeny 431919 Apr 25 19:45 analyze.ipynb
-rw-r--r-- 1 evgeny evgeny 634672 Apr 25 17:07 analyze.pdf
-rw-r--r-- 1 evgeny evgeny 87080 Apr 25 17:06 demo.ipynb
-rw-r--r-- 1 evgeny evgeny 314541 Apr 25 17:08 demo.pdf
-rw-r--r-- 1 evgeny evgeny 8 Apr 26 17:16 encrypted_text.txt
-rw-r--r-- 1 evgeny evgeny 35117 Apr 23 16:57 image.png
-rw-r--r-- 1 evgeny evgeny 4172 Apr 26 17:00 keygen.py
-rw-r--r-- 1 evgeny evgeny 8 Apr 26 17:16 manual_encrypted.txt
-rw-r--r-- 1 evgeny evgeny 24080 Apr 26 17:15 manual.ipynb
-rw-r--r-- 1 evgeny evgeny 7 Apr 26 17:16 manual_open.txt
-rw-r--r-- 1 evgeny evgeny 11 Apr 26 17:16 manual.public
-rw-r--r-- 1 evgeny evgeny 13 Apr 26 17:16 manual.secret
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 17:06 mayday_decrypted.png
-rw-r--r-- 1 evgeny evgeny 328461 Apr 25 17:06 mayday_encrypted.png
-rw-r--r-- 1 evgeny evgeny 306562 Apr 25 16:42 mayday.png
-rw-r--r-- 1 evgeny evgeny 408 Apr 20 15:36 open.txt
drwxr-xr-x 2 evgeny evgeny 4096 Apr 26 17:09 __pycache__
-rw-r--r-- 1 evgeny evgeny 339695 Apr 26 15:24 report.docx
-rw-r--r-- 1 evgeny evgeny 5100 Apr 25 19:02 rsa.py
-rw-r--r-- 1 evgeny evgeny 13 Apr 25 17:06 secret.crack
-rw-r--r-- 1 evgeny evgeny 11 Apr 25 17:06 small.public
-rw-r--r-- 1 evgeny evgeny 13 Apr 25 17:06 small.secret
drwxr-xr-x 6 evgeny evgeny 4096 Apr 15 15:18 venv
```

```
In [25]: rsa.decrypt("manual_encrypted.txt", "manual_decrypted.txt")  
!cat manual_decrypted.txt
```

evgeny

```
In [26]: # Вуаля еще раз
```