Приложение 1

Ручное шифрование

Определю открытый текст для шифрования

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

open_text = "evgeny"

# Предствлю 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", sig
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": "11001010111011001100111011001010110111001111
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, φ(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
```

Шаг 4.1 Вычислю значение закрытого ключа вручную

q	r	x	y	a	b	x_2	x_1	y_2	y_1
-	-	-	-	19320	223	1	0	0	1
86	142	1	-86	223	142	0	1	1	-86
1	81	-1	87	142	81	1	-1	-86	87
1	61	2	-173	81	61	-1	2	87	-173
1	20	-3	260	61	20	2	-3	-173	260
3	1	11	-953	20	1	-3	11	260	-953
20	0	-223	19320	1	0	11	-223	***-953***	19320
------	------	------	------	------	------	------	------	------	------

$$y_2 = -953(mod 19320) = 18367(mod 19320)$$

Что совпадает с предыдущими вычислениями

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

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, block_3_int, block_2_int, block_1_int
```

Out[9]: (25, 5990, 7573, 11897)

In [10]:
```
# Зашифруем блоки возведя в экспоненту e по модулю n

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[10]: (16690, 17436, 9865, 13431)

Произведу возведение в степень блока номер 1 вручную:

$$11897^{223} mod 19667 = 11897^{1+222} mod 19667 = 11897 * 11897^{222} mod 19667 =$$
$$11897 * 14877^{111} mod 19667 = 11897 * 14877^{3*37} mod 19667 =$$
$$11897 * 5385^{37} mod 19667 = 11897 * 5385^{36+1} mod 19667 =$$

$$11897 * 5385 * 5385^{36} mod 19667 = 11897 * 5385 * 5385^{3*3*2*2} mod 19667 =$$
$$11897 * 5385 * 12301^{3*2*2} mod 19667 = 11897 * 5385 * 16624^{2*2} mod 19667 =$$
$$11897 * 5385 * 16359^{2} mod 19667 = 11897 * 5385 * 8012 mod 19667 = \textbf{\textit{13431}}$$

In [11]:
```
# 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[11]:
```
('100000100110010', '100010000011100', '10011010001001', '110100011
10111')
```

In [12]:
```
# Проверим длину блоков

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

Out[12]:
```
(15, 15, 14, 14)
```

In [13]:
```
# Блок 1 и 2 имеют длину больше чем остальные.
# Выровняем 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), le
```

Out[13]:
```
(15, 15, 15, 15)
```

In [14]:
```
# 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), leng

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

# Прочитаем зашифрованные байты

!cat encrypted_text.txt
```

```
✓&Q☎♌D�w
```

In [15]:
```
# Расшифруем зашифрованный текст

# Преобразуем байты в биты

encrypted_text_bin = bin(int.from_bytes(encrypted_text_bytes, byteor
encrypted_text_bin
```

Out[15]:
```
'1000001001100101000100000111000100110100010010110100011101111'
```

In [16]:
```
# Измерим длину битовой строки

len(encrypted_text_bin)
```

Out[16]:
```
60
```

In [17]:
```
# Делится ли на нашу увеличенную длину блока нацело?

len(encrypted_text_bin) / block_size_full
```

Out[17]:
```
4.0
```

In [18]:
```
# 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[18]:
```
(25, 5990, 7573, 11897)
```

In [19]:
```
# 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_
    (block_4_int, block_3_int, block_2_int, block_1_int)
)
```

Out[19]:
```
True
```

In [20]:
```
# Для наглядности продолжу преобразования до получения исходной стро
# добавляю нолики в начале блока до исходного размера блока

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

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

Out[20]:
```
b'evgeny'
```

In [21]:
```
# Вуаля!
```

In [22]:
```
# 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 3120
-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 27 18:14 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  26456 Apr 27 18:13 manual.ipynb
-rw-r--r-- 1 evgeny evgeny      7 Apr 27 12:44 manual_open.txt
-rw-r--r-- 1 evgeny evgeny 315972 Apr 26 17:17 manual.pdf
-rw-r--r-- 1 evgeny evgeny     11 Apr 27 18:14 manual.public
-rw-r--r-- 1 evgeny evgeny     13 Apr 27 18:14 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 339238 Apr 26 17:23 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
drwxr-xr-x 6 evgeny evgeny   4096 Apr 15 15:18 venv
```

In [23]:
```
# Создам объект класса RSA для зашифрования и расшифрования

from rsa import RSA

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

Out[23]:
```
<rsa.RSA at 0x7f21d17bea50>
```

In [24]:
```
# Создам файл с текстов "evgeny"

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

```
total 3120
-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 27 18:14 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  26456 Apr 27 18:13 manual.ipynb
-rw-r--r-- 1 evgeny evgeny      7 Apr 27 18:14 manual_open.txt
-rw-r--r-- 1 evgeny evgeny 315972 Apr 26 17:17 manual.pdf
-rw-r--r-- 1 evgeny evgeny     11 Apr 27 18:14 manual.public
-rw-r--r-- 1 evgeny evgeny     13 Apr 27 18:14 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 339238 Apr 26 17:23 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
drwxr-xr-x 6 evgeny evgeny   4096 Apr 15 15:18 venv
```

In [25]:
```
# Зашифрую

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

```
total 3124
-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 27 18:14 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 27 18:14 manual_encrypted.txt
-rw-r--r-- 1 evgeny evgeny  26456 Apr 27 18:13 manual.ipynb
-rw-r--r-- 1 evgeny evgeny      7 Apr 27 18:14 manual_open.txt
-rw-r--r-- 1 evgeny evgeny 315972 Apr 26 17:17 manual.pdf
-rw-r--r-- 1 evgeny evgeny     11 Apr 27 18:14 manual.public
-rw-r--r-- 1 evgeny evgeny     13 Apr 27 18:14 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 339238 Apr 26 17:23 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
drwxr-xr-x 6 evgeny evgeny   4096 Apr 15 15:18 venv
```

In [26]:
```
rsa.decrypt("manual_encrypted.txt", "manual_decrypted.txt")
!cat manual_decrypted.txt
```

```
evgeny
```

In [27]:
```
# Вуаля еще раз
```