RSA CIPHER

ECRYPT PROJECT 2023

Maria Koilalou

Georgia Bousmpoukea

The purpose of this project is the software implementation of the RSA cipher in two ciphering modes: ECB, CBC. RSA is a public-key cipher, widely used for secure data transmission.

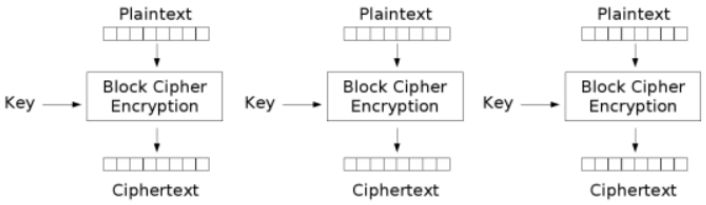
To generate the public and private keys the following process is used:

- Generate two large prime numbers p, q
- Calculate n=p* q and φ[n]=[p-1]* [q-1]
- Choose a random number e, coprime to φ[n]
- Calculate d as the modular inverse of e modulo φ[n]
- The public key is [n, e] and the private key is [n, d]

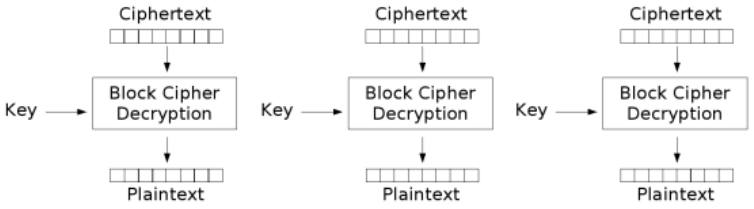
The cyphertext of the transmitted message m is calculated as: $c=m^e[mod n]$ and the decrypted message is $m=c^d[mod n]$.

ECB mode

In this mode the transmitted message for the encryption is divided into blocks. Each block then is encrypted using the above RSA algorithm. For given keys, the result of the algorithm is deterministic. In the decryption each block of the ciphertext is used to generate the corresponding plaintext message block.



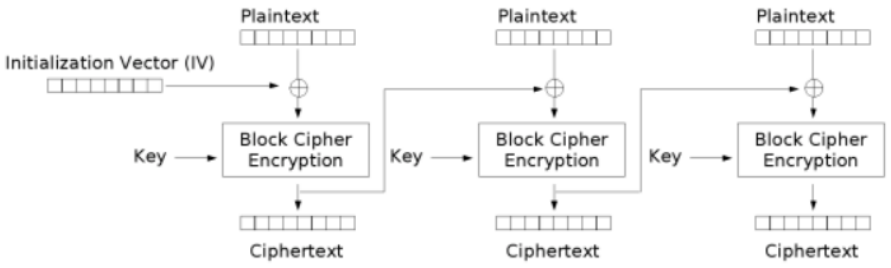
Electronic Codebook (ECB) mode encryption



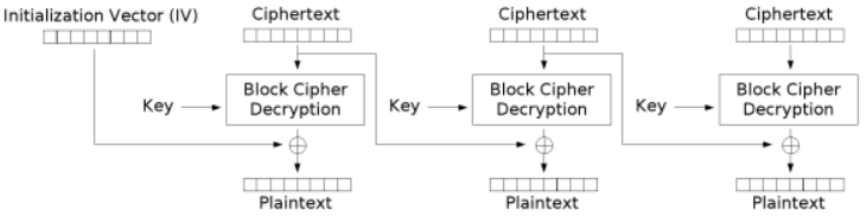
Electronic Codebook (ECB) mode decryption

CBC mode

This is a more secure mode for RSA. Now before the encryption the blocks of the message are XORed with the previous block of the ciphertext. This way, each ciphertext block is dependent on all plaintext blocks up to that point, introducing randomness, since, for different initialization vector, the ciphertext is different. In the decryption a similar process is followed.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Description of algorithm

Our code take as an input a file [string input, the name of the file], calculates the public and private keys, performs encryption and decryption in ECB and CBC modes and gives as an output the ciphertext and the decrypted text in each mode.

- We import the necessary libraries:

```
import math
import random
import sympy
import os
```

- Key generation:

We use the function ***generate_prime()*** to generate a prime of 1024 bits and the function ***generate_keys()***, which returns two tuples with the public and private keys.

```python
# Function to generate a prime number
ntua-19211
def generate_prime():
    # Generate a random number with 1024 bits.
    p = random.getrandbits(1024)
    if is_prime(p):
        return p
    else:
        t = sympy.nextprime(p)
        return t
```

```python
# Function to check if number is prime with Miller-Rabin primality test.
ntua-19211
def is_prime(n, k=128):
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 2 == 0:
        return False
    r, s = 0, n - 1
    while s % 2 == 0:
        r += 1
        s //= 2
    for _ in range(k):
        a = random.randrange(2, n - 1)
        x = pow(a, s, n)
        if x == 1 or x == n - 1:
            continue
        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True
```

```python
# Function to generate the public and private keys
ntua-19211
def generate_keys():
    # Generate two large prime numbers
    p = generate_prime()
    q = generate_prime()

    # Calculate n and phi(n)
    n = p * q
    phi = (p - 1) * (q - 1)

    # Choose a random number e that is coprime to phi(n)
    e = random.randrange(1, phi)
    g = math.gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = math.gcd(e, phi)

    # Calculate the modular multiplicative inverse of e modulo phi(n)
    d = mod_inverse(e, phi)

    # Return the public and private keys
    return (n, e), (n, d)
```

```python
# Function to calculate the modular multiplicative inverse of a modulo m using the extended Euclidean algorithm
ntua-19211
def mod_inverse(a, m):
    # Compute the modular inverse of a modulo m using the extended Euclidean algorithm
    x, y, lastx, lasty = 0, 1, 1, 0
    while m:
        a, (q, m) = m, divmod(a, m)
        lastx, x = x, lastx - q * x
        lasty, y = y, lasty - q * y
    return lastx
```

- Padding:

We use function **pad_message[]**, that takes as an input a message in byte form and the desired block size and pads the message, so that its length is multiple of the block size (if the size of the message is equal to the block size a block is added). Each byte added contains the size of the padding (i.e. if we need a padding o 5 bytes the padding is '\x03\x03\x03').

The function ***unpad_message[]***, takes as input a message in byte form and removes the padding, by removing number of bytes equal to the value of the last byte (since the last byte always includes the size of the padding, that's why when the size of the message is equal to the block size we add one extra block).

```
# Function to pad the message
ntua-19211
def pad_message(msg, block_size):
    pad_size = block_size - len(msg) % block_size
    padding = bytes([pad_size] * pad_size)
    padded_msg = msg + padding
    return padded_msg


# Function to unpad the message
ntua-19211
def unpad_message(padded_message):
    padding_length = padded_message[-1]
    return padded_message[:-padding_length]
```

- ECB mode:

Encryption is performed by the ***encrypt_ecb[]*** function, which takes as input the public key and the message. The block size is chosen according to the size of n. The message is transformed in byte format and then it is padded and divided into blocks. For each block the encryption is performed: convert the byte of the block into an integer and calculate the corresponding ciphertext ($c = m^e \pmod n$). A string of the joined ciphertexts in byte format is returned.

Decryption is performed by the ***decrypt_ecb()*** function, which takes as input the private key and the ciphertext. Each block of the ciphertext is converted into an integer and decryption is performed [$m=c^d[mod n]$]. The result is transformed into byte format, then it's unpadded and finally it is transformed back to string using the UTF-8 encoding.

```python
# Function that performs encryption with ECB mode
ntua-19211
def encrypt_ecb(public_key, msg):
    # Extracts public key values
    n, e = public_key
    # Calculates the block size needed based on the key length
    block_size = (n.bit_length() + 7) // 8 - 1
    # Pads the message to be a multiple of the block size
    padded_msg = pad_message(msg.encode(), block_size)
    # Breaks the padded message into blocks
    blocks = [padded_msg[i:i + block_size] for i in range(0, len(padded_msg), block_size)]
    cipher_blocks = []
    # Encrypts each block and appends it to a list
    for block in blocks:
        m = int.from_bytes(block, byteorder='big')
        c = pow(m, e, n)
        cipher_blocks.append(c.to_bytes(block_size + 1, byteorder='big'))
    # Joins the encrypted blocks into a single byte string
    return b''.join(cipher_blocks)
```

```python
# Function that performs decryption with ECB mode
ntua-19211
def decrypt_ecb(private_key, ciphertext):
    # Extracts private key values
    n, d = private_key
    # Calculates the block size needed based on the key length
    block_size = (n.bit_length() + 7) // 8 - 1
    # Breaks the ciphertext into blocks
    blocks = [ciphertext[i:i + block_size + 1] for i in range(0, len(ciphertext), block_size + 1)]
    # Decrypts each block and appends it to a list
    plaintext_blocks = []
    for block in blocks:
        c = int.from_bytes(block, byteorder='big')
        m = pow(c, d, n)
        plaintext_blocks.append(m.to_bytes(block_size, byteorder='big'))
    # Joins the decrypted blocks into a single byte string
    plaintext = unpad_message(b''.join(plaintext_blocks))
    # Removes any leading null bytes and returns the plaintext as a string
    plaintext = plaintext.lstrip(b'\x00')
    return plaintext.decode('utf-8')
```

- CBC mode:

The general idea is the same, but now a random initialisation vector is used as the first block of the ciphertext. The following blocks of the ciphertext are calculated by XORing the corresponding block of the message with the previous ciphertext block. In similar way the decrypted message is calculated.

```python
# Function that performs encryption with CBC mode
ntua-19211
def encrypt_cbc(public_key, msg):
    # Extracts public key values
    n, e = public_key
    # Calculates the block size needed based on the key length
    block_size = (n.bit_length() + 7) // 8 - 1
    # Generates a random initialization vector
    iv = os.urandom(block_size)
    # Pads the message to be a multiple of the block size
    padded_msg = pad_message(msg.encode(), block_size)
    # Breaks the padded message into blocks and XORs each block with the previous ciphertext block
    blocks = [padded_msg[i:i + block_size] for i in range(0, len(padded_msg), block_size)]
    cipher_blocks = [iv]
    for block in blocks:
        m = int.from_bytes(block, byteorder='big')
        c = pow(m ^ int.from_bytes(cipher_blocks[-1], byteorder='big'), e, n)
        cipher_blocks.append(c.to_bytes(block_size + 1, byteorder='big'))
    # Joins the initialization vector and the encrypted blocks into a single byte string
    ciphertext = iv + b''.join(cipher_blocks[1:])
    return ciphertext
```

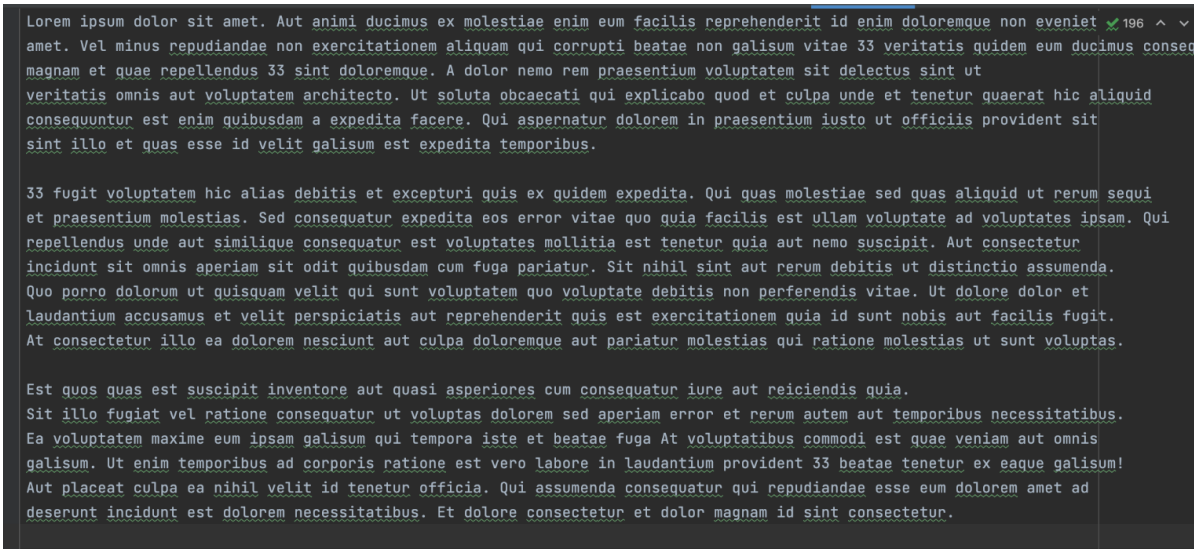
```python
# Function that performs decryption with CBC mode
ntua-19211
def decrypt_cbc(private_key, ciphertext):
    # Extracts private key values
    n, d = private_key
    # Calculates the block size needed based on the key length
    block_size = (n.bit_length() + 7) // 8 - 1
    iv = ciphertext[:block_size]
    # Breaks the ciphertext into blocks
    blocks = [ciphertext[i:i + block_size + 1] for i in range(block_size, len(ciphertext), block_size + 1)]
    # Decrypts each block and appends it to a list
    plaintext_blocks = []
    prev_block = iv
    for block in blocks:
        c = int.from_bytes(block, byteorder='big')
        m = pow(c, d, n) ^ int.from_bytes(prev_block, byteorder='big')
        plaintext_blocks.append(m.to_bytes(block_size, byteorder='big'))
        prev_block = block
    # Joins the decrypted blocks into a single byte string
    plaintext = unpad_message(b''.join(plaintext_blocks))
    # Removes any leading null bytes and returns the plaintext as a string
    plaintext = plaintext.lstrip(b'\x00')
    return plaintext.decode()
```

Testing

We created 7 files with different data (ex. Chinese or Greek characters, numbers, emojis) and we want to check if the result of the encryption followed by the decryption is the same as the initial text.

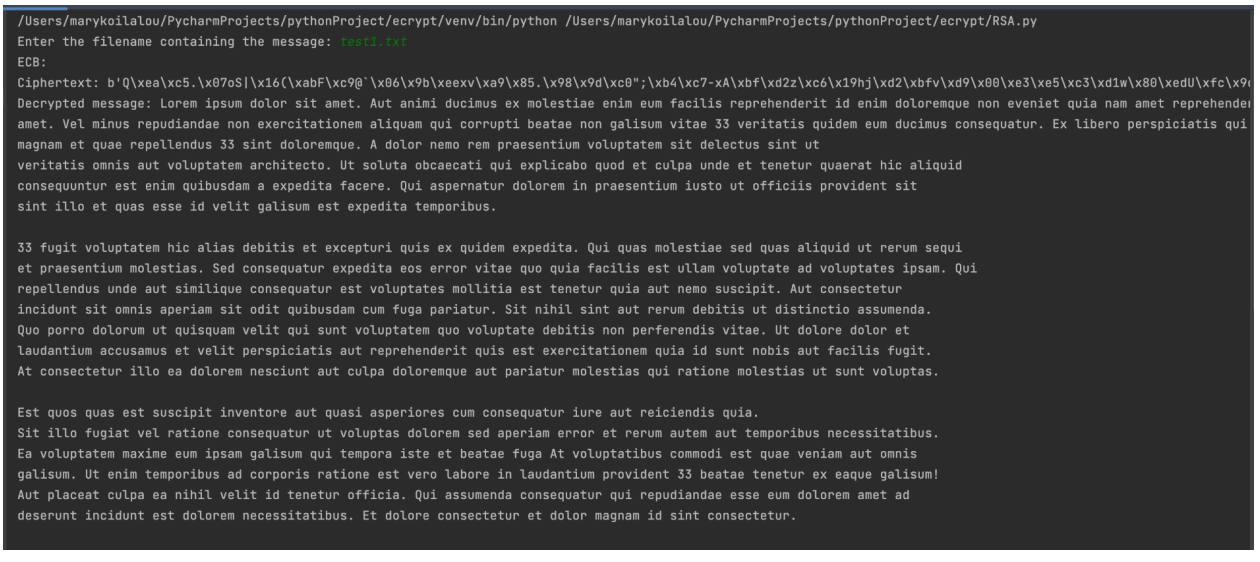
- test1.txt

Input:



```
Lorem ipsum dolor sit amet. Aut animi ducimus ex molestiae enim eum facilis reprehenderit id enim doloremque non eveniet
amet. Vel minus repudiandae non exercitationem aliquam qui corrupti beatae non galisum vitae 33 veritatis quidem eum ducimus consequ
magnam et quae repellendus 33 sint doloremque. A dolor nemo rem praesentium voluptatem sit delectus sint ut
veritatis omnis aut voluptatem architecto. Ut soluta obcaecati qui explicabo quod et culpa unde et tenetur quaerat hic aliquid
consequuntur est enim quibusdam a expedita facere. Qui aspernatur dolorem in praesentium iusto ut officiis provident sit
sint illo et quas esse id velit galisum est expedita temporibus.

33 fugit voluptatem hic alias debitis et excepturi quis ex quidem expedita. Qui quas molestiae sed quas aliquid ut rerum sequi
et praesentium molestias. Sed consequatur expedita eos error vitae quo quia facilis est ullam voluptate ad voluptates ipsam. Qui
repellendus unde aut similique consequatur est voluptates mollitia est tenetur quia aut nemo suscipit. Aut consectetur
incidunt sit omnis aperiam sit odit quibusdam cum fuga pariatur. Sit nihil sint aut rerum debitis ut distinctio assumenda.
Quo porro dolorum ut quisquam velit qui sunt voluptatem quo voluptate debitis non perferendis vitae. Ut dolore dolor et
laudantium accusamus et velit perspiciatis aut reprehenderit quis est exercitationem quia id sunt nobis aut facilis fugit.
At consectetur illo ea dolorem nesciunt aut culpa doloremque aut pariatur molestias qui ratione molestias ut sunt voluptas.

Est quos quas est suscipit inventore aut quasi asperiores cum consequatur iure aut reiciendis quia.
Sit illo fugiat vel ratione consequatur ut voluptas dolorem sed aperiam error et rerum autem aut temporibus necessitatibus.
Ea voluptatem maxime eum ipsam galisum qui tempora iste et beatae fuga At voluptatibus commodi est quae veniam aut omnis
galisum. Ut enim temporibus ad corporis ratione est vero labore in laudantium provident 33 beatae tenetur ex eaque galisum!
Aut placeat culpa ea nihil velit id tenetur officia. Qui assumenda consequatur qui repudiandae esse eum dolorem amet ad
deserunt incidunt est dolorem necessitatibus. Et dolore consectetur et dolor magnam id sint consectetur.
```

Output:



```
/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test1.txt
ECB:
Ciphertext: b'Q\xea\xc5.\x07oS|\x16(\xabF\xc9@`\x06\x9b\xeexv\xa9\x85.\x98\x9d\xc0";\xb4\xc7-xA\xbf\xd2z\xc6\x19hj\xd2\xbfv\xd9\x00\xe3\xe5\xc3\xd1w\x80\xedU\xfc\x9
Decrypted message: Lorem ipsum dolor sit amet. Aut animi ducimus ex molestiae enim eum facilis reprehenderit id enim doloremque non eveniet quia nam amet reprehende
amet. Vel minus repudiandae non exercitationem aliquam qui corrupti beatae non galisum vitae 33 veritatis quidem eum ducimus consequatur. Ex libero perspiciatis qui
magnam et quae repellendus 33 sint doloremque. A dolor nemo rem praesentium voluptatem sit delectus sint ut
veritatis omnis aut voluptatem architecto. Ut soluta obcaecati qui explicabo quod et culpa unde et tenetur quaerat hic aliquid
consequuntur est enim quibusdam a expedita facere. Qui aspernatur dolorem in praesentium iusto ut officiis provident sit
sint illo et quas esse id velit galisum est expedita temporibus.

33 fugit voluptatem hic alias debitis et excepturi quis ex quidem expedita. Qui quas molestiae sed quas aliquid ut rerum sequi
et praesentium molestias. Sed consequatur expedita eos error vitae quo quia facilis est ullam voluptate ad voluptates ipsam. Qui
repellendus unde aut similique consequatur est voluptates mollitia est tenetur quia aut nemo suscipit. Aut consectetur
incidunt sit omnis aperiam sit odit quibusdam cum fuga pariatur. Sit nihil sint aut rerum debitis ut distinctio assumenda.
Quo porro dolorum ut quisquam velit qui sunt voluptatem quo voluptate debitis non perferendis vitae. Ut dolore dolor et
laudantium accusamus et velit perspiciatis aut reprehenderit quis est exercitationem quia id sunt nobis aut facilis fugit.
At consectetur illo ea dolorem nesciunt aut culpa doloremque aut pariatur molestias qui ratione molestias ut sunt voluptas.

Est quos quas est suscipit inventore aut quasi asperiores cum consequatur iure aut reiciendis quia.
Sit illo fugiat vel ratione consequatur ut voluptas dolorem sed aperiam error et rerum autem aut temporibus necessitatibus.
Ea voluptatem maxime eum ipsam galisum qui tempora iste et beatae fuga At voluptatibus commodi est quae veniam aut omnis
galisum. Ut enim temporibus ad corporis ratione est vero labore in laudantium provident 33 beatae tenetur ex eaque galisum!
Aut placeat culpa ea nihil velit id tenetur officia. Qui assumenda consequatur qui repudiandae esse eum dolorem amet ad
deserunt incidunt est dolorem necessitatibus. Et dolore consectetur et dolor magnam id sint consectetur.
```

```
CBC:
Ciphertext: b'*\x90\xb4\x86\x17\xa5\xbe\x14'...
Decrypted message: Lorem ipsum dolor sit amet. Aut animi ducimus ex molestiae enim eum facilis reprehenderit id enim doloremque non eveniet quia nam amet reprehender
amet. Vel minus repudiandae non exercitationem aliquam qui corrupti beatae non galisum vitae 33 veritatis quidem eum ducimus consequatur. Ex libero perspiciatis qui
magnam et quae repellendus 33 sint doloremque. A dolor nemo rem praesentium voluptatem sit delectus sint ut
veritatis omnis aut voluptatem architecto. Ut soluta obcaecati qui explicabo quod et culpa unde et tenetur quaerat hic aliquid
consequuntur est enim quibusdam a expedita facere. Qui aspernatur dolorem in praesentium iusto ut officiis provident sit
sint illo et quas esse id velit galisum est expedita temporibus.

33 fugit voluptatem hic alias debitis et excepturi quis ex quidem expedita. Qui quas molestiae sed quas aliquid ut rerum sequi
et praesentium molestias. Sed consequatur expedita eos error vitae quo quia facilis est ullam voluptate ad voluptates ipsam. Qui
repellendus unde aut similique consequatur est voluptates mollitia est tenetur quia aut nemo suscipit. Aut consectetur
incidunt sit omnis aperiam sit odit quibusdam cum fuga pariatur. Sit nihil sint aut rerum debitis ut distinctio assumenda.
Quo porro dolorum ut quisquam velit qui sunt voluptatem quo voluptate debitis non perferendis vitae. Ut dolore dolor et
laudantium accusamus et velit perspiciatis aut reprehenderit quis est exercitationem quia id sunt nobis aut facilis fugit.
At consectetur illo ea dolorem nesciunt aut culpa doloremque aut pariatur molestias qui ratione molestias ut sunt voluptas.

Est quos quas est suscipit inventore aut quasi asperiores cum consequatur iure aut reiciendis quia.
Sit illo fugiat vel ratione consequatur ut voluptas dolorem sed aperiam error et rerum autem aut temporibus necessitatibus.
Ea voluptatem maxime eum ipsam galisum qui tempora iste et beatae fuga At voluptatibus commodi est quae veniam aut omnis
galisum. Ut enim temporibus ad corporis ratione est vero labore in laudantium provident 33 beatae tenetur ex eaque galisum!
Aut placeat culpa ea nihil velit id tenetur officia. Qui assumenda consequatur qui repudiandae esse eum dolorem amet ad
deserunt incidunt est dolorem necessitatibus. Et dolore consectetur et dolor magnam id sint consectetur.
```

- test2.txt

Input:

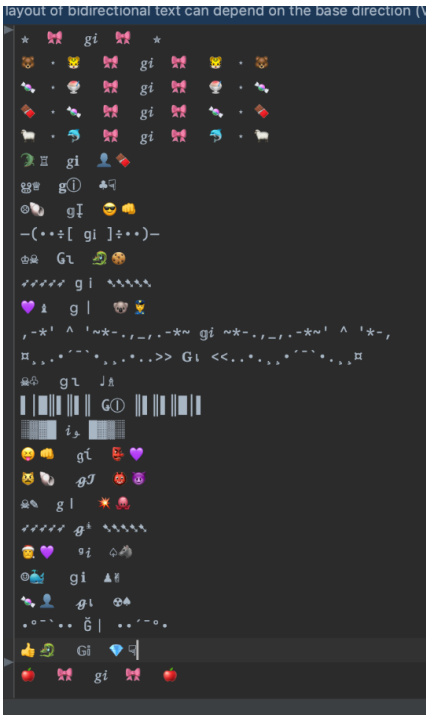
```
レの尺モᄊ ノア丂ひᄊ りのレの尺 丂ノイ ム爪モイ. ムひイ ム刀ᄊノ りひㄷノᄊひ丂 モメ ᄊのレモ丂イ/ムモ モ刀ノᄊ モひᄊ ヰムㄷルノ丂 尺モア尺モん乇刀り乇尺ノイ
ノり モ刀ノᄊ りのレの尺モᄊぁひモ 刀の刀 モV乇刀ノ乇イ ぁひ/ム 刀ム爪 ム爪モイ 尺モア尺モん乇刀り乇尺ノイ. V乇レ のᄊ刀ノ丂 ムひイ乇ᄊ ムひイ モレノム乇刀りノ
りノム刀ノ丂丂ノᄊのち ひイ 刀モ丂ㄷノひ刀イ
ム爪モイ.
```

Output:

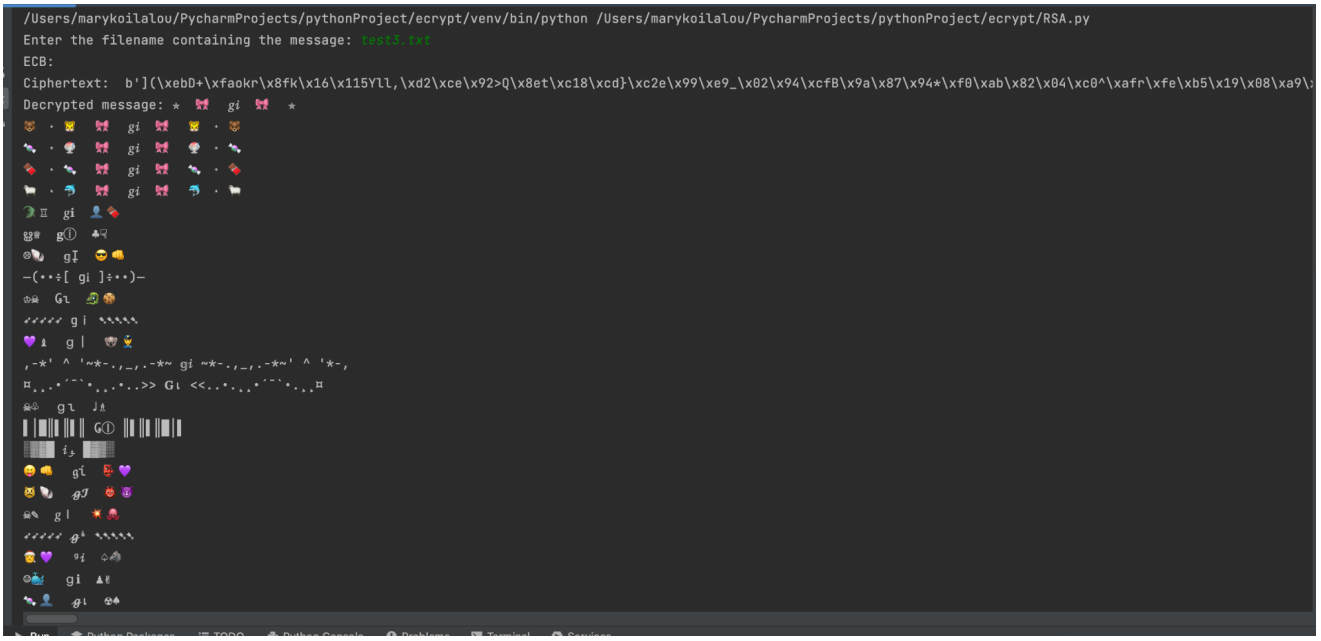
```
/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test2.txt
ECB:
Ciphertext: b'-\xc1\xa3\xca\xdd%{\xd1B\x8c%\x96\x0c{\xce\x11\xe6I\xca\xec?)\xc7GA\x84"\xc5\xbaq\xf1<K\x10\xb5;]\xfc\x96\xc0\xab\x86\x98<\xa9a\xbe\x08]\x9b\xcc\x02\x
Decrypted message: レの尺モᄊ ノア丂ひᄊ りのレの尺 丂ノイ ム爪モイ. ムひイ ム刀ᄊノ りひㄷノᄊひ丂 モメ ᄊのレモ丂イ/ムモ モ刀ノᄊ モひᄊ ヰムㄷルノ丂 尺モア尺モん乇刀り乇尺ノイ
ノり モ刀ノᄊ りのレの尺モᄊぁひモ 刀の刀 モV乇刀ノ乇イ ぁひ/ム 刀ム爪 ム爪モイ 尺モア尺モん乇刀り乇尺ノイ. V乇レ のᄊ刀ノ丂 ムひイ乇ᄊ ムひイ モレノム乇刀りノ
りノム刀ノ丂丂ノᄊのち ひイ 刀モ丂ㄷノひ刀イ
ム爪モイ.
CBC:
Ciphertext:  b'B\xb8DW\xea\x98\x9bT\x82\x17H#\xfb\xdb\x12=\x8b\xf6\xb2\xe8\xd8+\xd2\xb9\x18^k\xf9\x88\x02\x05\xf2lN\x83w`\xb2U\x0b`\xf80x\x8b\x85W\xe5T#~\xd1\x91\x9
Decrypted message: レの尺モᄊ ノア丂ひᄊ りのレの尺 丂ノイ ム爪モイ. ムひイ ム刀ᄊノ りひㄷノᄊひ丂 モメ ᄊのレモ丂イ/ムモ モ刀ノᄊ モひᄊ ヰムㄷルノ丂 尺モア尺モん乇刀り乇尺ノイ
ノり モ刀ノᄊ りのレの尺モᄊぁひモ 刀の刀 モV乇刀ノ乇イ ぁひ/ム 刀ム爪 ム爪モイ 尺モア尺モん乇刀り乇尺ノイ. V乇レ のᄊ刀ノ丂 ムひイ乇ᄊ ムひイ モレノム乇刀りノ
りノム刀ノ丂丂ノᄊのち ひイ 刀モ丂ㄷノひ刀イ
ム爪モイ.
```

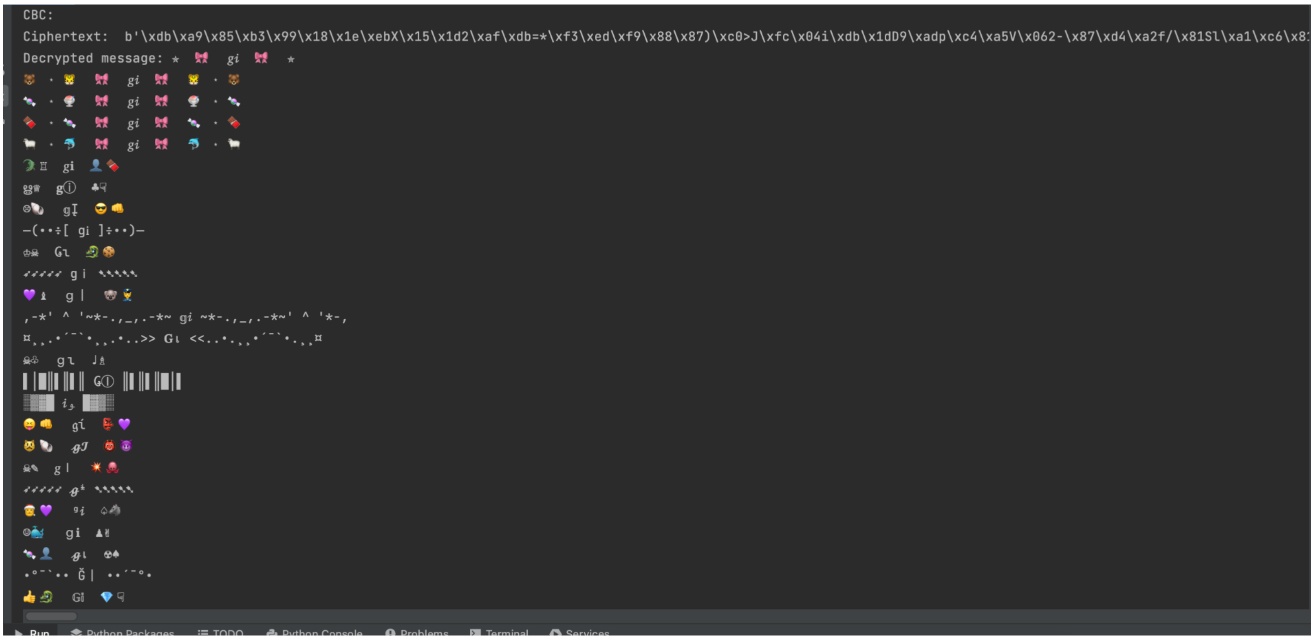
- test3.txt

Input:



Output:





- test4.txt

Input:

```
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
```

Output:

```
/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test4.txt
ECB:
Ciphertext:  b'3#0\xb7\xe3\x17\xbe\xf7\x98\x10J6\x90\xd2\x99!t\xc6\xee\x85\xdf\xbe\xd7\\U\xa3-\'\x83\x91\xcc\x82\xb2[\t\xc5\x924; X\xd8\x06#\xdd \xa3\xd3\xb3\xef\xea
Decrypted message: 78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
```

```
CBC:
Ciphertext:  b'\xb7\x82m\x8c\t\xc7M-\x88\xa9\xe8\x865\x9d\xf8\x93\xdf\xd6;\x8dYH\x83\xd4\x14O\x9e\x1bi]\r\xfa\xd1d\xa8o\x0c\xe0\xbb^\xac\x9c\x98\xee\x86\xb1z:<#\x14\
Decrypted message: 78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982
78,877,325,319,198,790,505,267,261,125,965,041,816,439,817,849,518,
229,771,036,394,183,187,681,915,304,982,380,283,225,703,329,624,982

Process finished with exit code 0
```

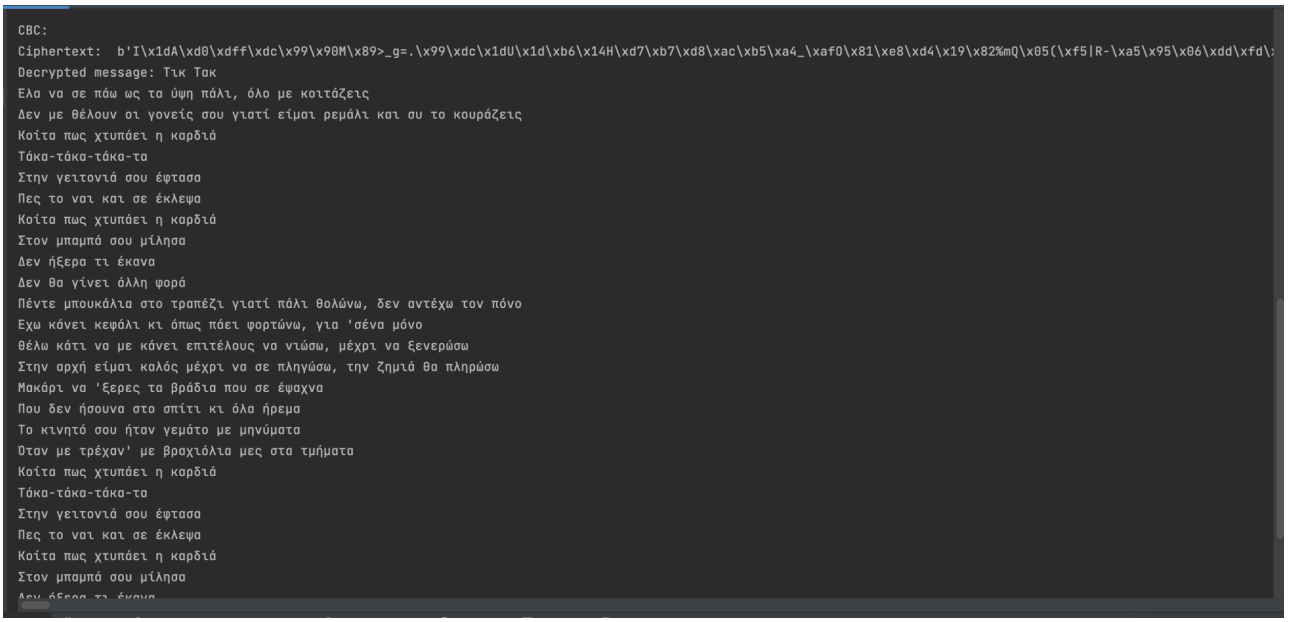
- tests.txt

Input:

```
Τικ Τακ
Ελα να σε πάω ως τα ύψη πάλι, όλο με κοιτάζεις
Δεν με θέλουν οι γονείς σου γιατί είμαι ρεμάλι και συ το κουράζεις
Κοίτα πως χτυπάει η καρδιά
Τάκα-τάκα-τάκα-τα
Στην γειτονιά σου έφτασα
Πες το ναι και σε έκλεψα
Κοίτα πως χτυπάει η καρδιά
Στον μπαμπά σου μίλησα
Δεν ήξερα τι έκανα
Δεν θα γίνει άλλη φορά
Πέντε μπουκάλια στο τραπέζι γιατί πάλι θολώνω, δεν αντέχω τον πόνο
Εχω κάνει κεφάλι κι όπως πάει φορτώνω, για 'σένα μόνο
θέλω κάτι να με κάνει επιτέλους να νιώσω, μέχρι να ξενερώσω
Στην αρχή είμαι καλός μέχρι να σε πληγώσω, την ζημιά θα πληρώσω
Μακάρι να 'ξερες τα βράδια που σε έψαχνα
Που δεν ήσουνα στο σπίτι κι όλα ήρεμα
Το κινητό σου ήταν γεμάτο με μηνύματα
Οταν με τρέχαν' με βραχιόλια μες στα τμήματα
Κοίτα πως χτυπάει η καρδιά
Τάκα-τάκα-τάκα-τα
Στην γειτονιά σου έφτασα
Πες το ναι και σε έκλεψα
Κοίτα πως χτυπάει η καρδιά
Στον μπαμπά σου μίλησα
Δεν ήξερα τι έκανα
Δεν θα γίνει άλλη φορά
```

Output:

```
/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test5.txt
ECB:
Ciphertext:  b'\x02\xf6c\xc2R9~{yw\xcd\xb9\xdc\x91\xda_#!R\xc3\xaf\x8e\xfc\x91\xd2\x9bN\x91\x7fv\x17(\xda\xc4\xd6\xa4\xb7K\xe1\x7fe\xc7}Oe\x04\xd7\xcdma\xe3<\xa71\x
Decrypted message: Τικ Τακ
Ελα να σε πάω ως τα ύψη πάλι, όλο με κοιτάζεις
Δεν με θέλουν οι γονείς σου γιατί είμαι ρεμάλι και συ το κουράζεις
Κοίτα πως χτυπάει η καρδιά
Τάκα-τάκα-τάκα-τα
Στην γειτονιά σου έφτασα
Πες το ναι και σε έκλεψα
Κοίτα πως χτυπάει η καρδιά
Στον μπαμπά σου μίλησα
Δεν ήξερα τι έκανα
Δεν θα γίνει άλλη φορά
Πέντε μπουκάλια στο τραπέζι γιατί πάλι θολώνω, δεν αντέχω τον πόνο
Εχω κάνει κεφάλι κι όπως πάει φορτώνω, για 'σένα μόνο
θέλω κάτι να με κάνει επιτέλους να νιώσω, μέχρι να ξενερώσω
Στην αρχή είμαι καλός μέχρι να σε πληγώσω, την ζημιά θα πληρώσω
Μακάρι να 'ξερες τα βράδια που σε έψαχνα
Που δεν ήσουνα στο σπίτι κι όλα ήρεμα
Το κινητό σου ήταν γεμάτο με μηνύματα
Οταν με τρέχαν' με βραχιόλια μες στα τμήματα
Κοίτα πως χτυπάει η καρδιά
Τάκα-τάκα-τάκα-τα
Στην γειτονιά σου έφτασα
Πες το ναι και σε έκλεψα
Κοίτα πως χτυπάει η καρδιά
```

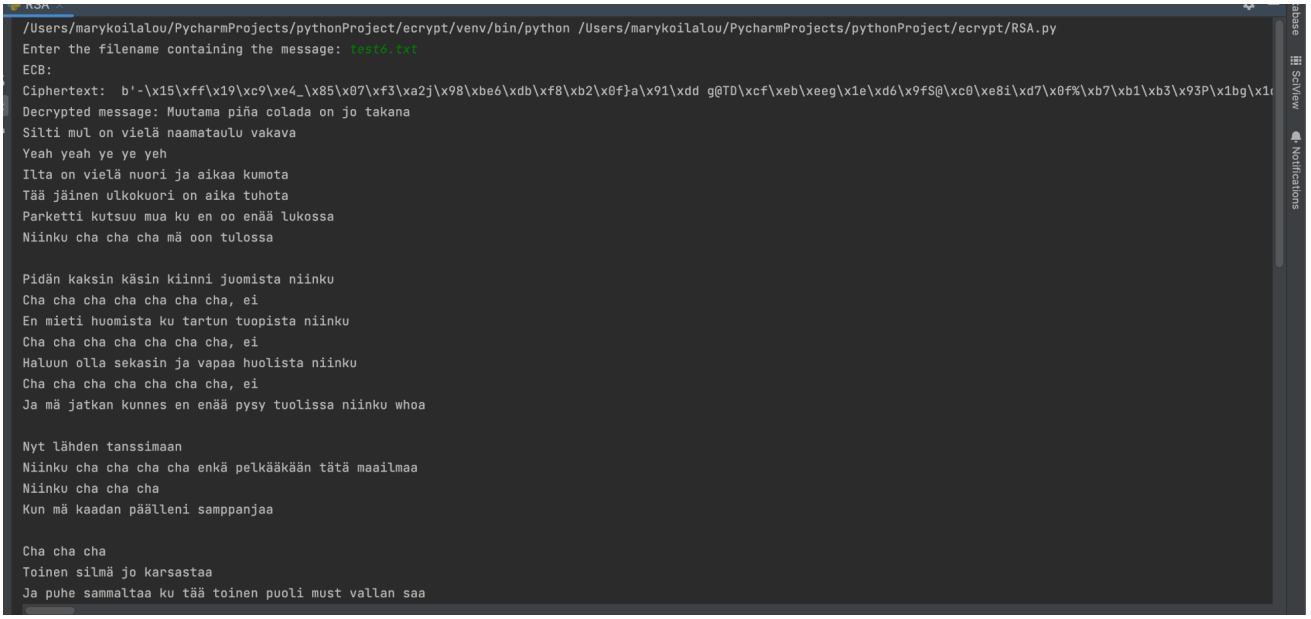


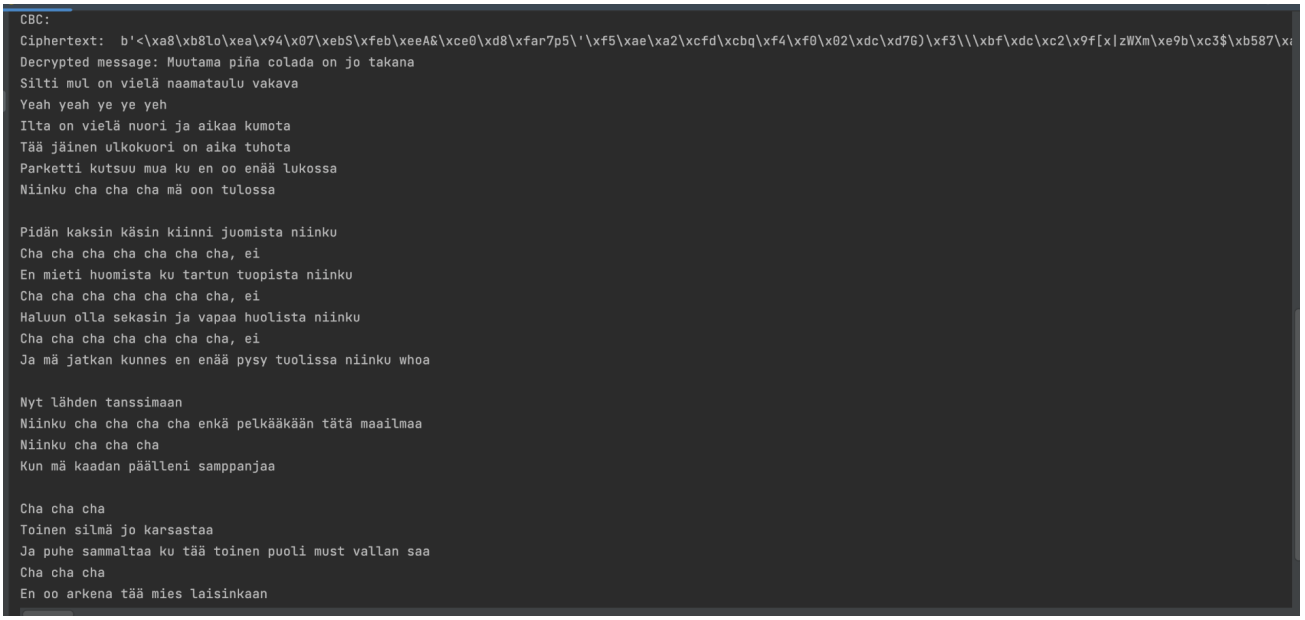
- `testo.txt`

Input:

```
Muutama piña colada on jo takana
Silti mul on vielä naamataulu vakava
Yeah yeah ye ye yeh
Ilta on vielä nuori ja aikaa kumota
Tää jäinen ulkokuori on aika tuhota
Parketti kutsuu mua ku en oo enää lukossa
Niinku cha cha cha mä oon tulossa

Pidän kaksin käsin kiinni juomista niinku
Cha cha cha cha cha cha cha, ei
En mieti huomista ku tartun tuopista niinku
Cha cha cha cha cha cha cha, ei
Haluun olla sekasin ja vapaa huolista niinku
Cha cha cha cha cha cha cha, ei
Ja mä jatkan kunnes en enää pysy tuolissa niinku whoa

Nyt lähden tanssimaan
Niinku cha cha cha cha enkä pelkääkään tätä maailmaa
Niinku cha cha cha
Kun mä kaadan päälleni samppanjaa

Cha cha cha
Toinen silmä jo karsastaa
Ja puhe sammaltaa ku tää toinen puoli must vallan saa
Cha cha cha
En oo arkena tää mies laisinkaan
En oo mut tänään oon se mies, tänään oon se mies
```

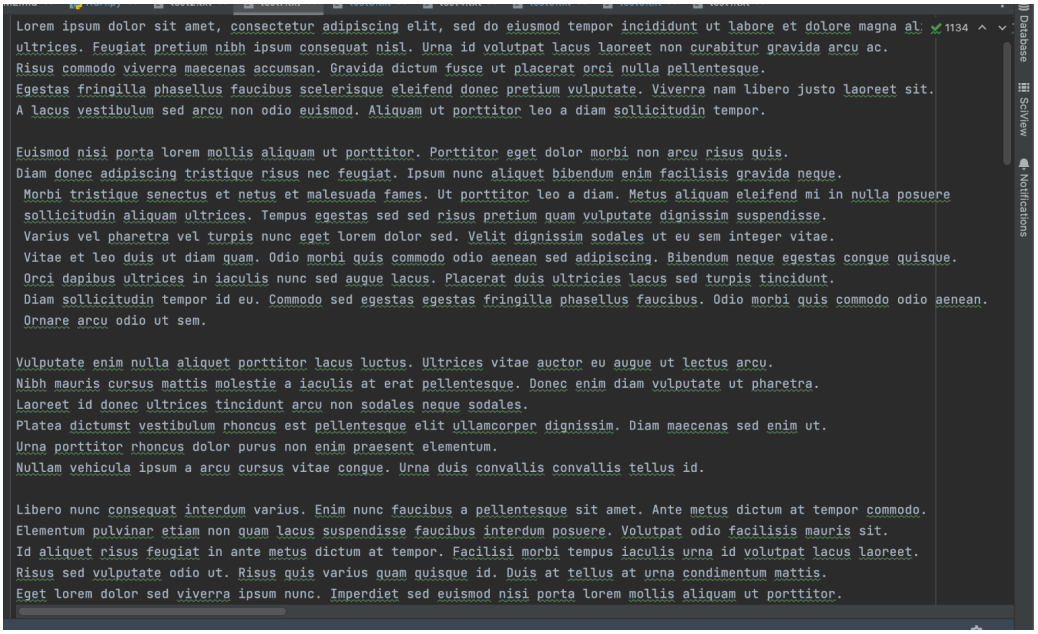
Output:


/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test6.txt
ECB:
Ciphertext: b'-\x15\xff\x19\xc9\xe4_\x85\x07\xf3\xa2j\x98\xbe6\xdb\xf8\xb2\x0f}a\x91\xdd g@TD\xcf\xeb\xeeg\x1e\xd6\x9fS@\xc0\xe8i\xd7\x0f%\xb7\xb1\xb3\x93P\x1bg\x1
Decrypted message: Muutama piña colada on jo takana
Silti mul on vielä naamataulu vakava
Yeah yeah ye ye yeh
Ilta on vielä nuori ja aikaa kumota
Tää jäinen ulkokuori on aika tuhota
Parketti kutsuu mua ku en oo enää lukossa
Niinku cha cha cha mä oon tulossa
Pidän kaksin käsin kiinni juomista niinku
Cha cha cha cha cha cha cha, ei
En mieti huomista ku tartun tuopista niinku
Cha cha cha cha cha cha cha, ei
Haluun olla sekasin ja vapaa huolista niinku
Cha cha cha cha cha cha cha, ei
Ja mä jatkan kunnes en enää pysy tuolissa niinku whoa
Nyt lähden tanssimaan
Niinku cha cha cha cha enkä pelkääkään tätä maailmaa
Niinku cha cha cha
Kun mä kaadan päälleni samppanjaa
Cha cha cha
Toinen silmä jo karsastaa
Ja puhe sammaltaa ku tää toinen puoli must vallan saa


CBC:
Ciphertext: b'<\xa8\xb8lo\xea\x94\x07\xebS\xfeb\xeeA&\xce0\xd8\xfar7p5\'\xf5\xae\xa2\xcfd\xcbq\xf4\xf0\x02\xdc\xd7G)\xf3\\\xbf\xdc\xc2\x9f[x|zWXm\xe9b\xc3$\xb587\x
Decrypted message: Muutama piña colada on jo takana
Silti mul on vielä naamataulu vakava
Yeah yeah ye ye yeh
Ilta on vielä nuori ja aikaa kumota
Tää jäinen ulkokuori on aika tuhota
Parketti kutsuu mua ku en oo enää lukossa
Niinku cha cha cha mä oon tulossa
Pidän kaksin käsin kiinni juomista niinku
Cha cha cha cha cha cha cha, ei
En mieti huomista ku tartun tuopista niinku
Cha cha cha cha cha cha cha, ei
Haluun olla sekasin ja vapaa huolista niinku
Cha cha cha cha cha cha cha, ei
Ja mä jatkan kunnes en enää pysy tuolissa niinku whoa
Nyt lähden tanssimaan
Niinku cha cha cha cha enkä pelkääkään tätä maailmaa
Niinku cha cha cha
Kun mä kaadan päälleni samppanjaa
Cha cha cha
Toinen silmä jo karsastaa
Ja puhe sammaltaa ku tää toinen puoli must vallan saa
Cha cha cha
En oo arkena tää mies laisinkaan

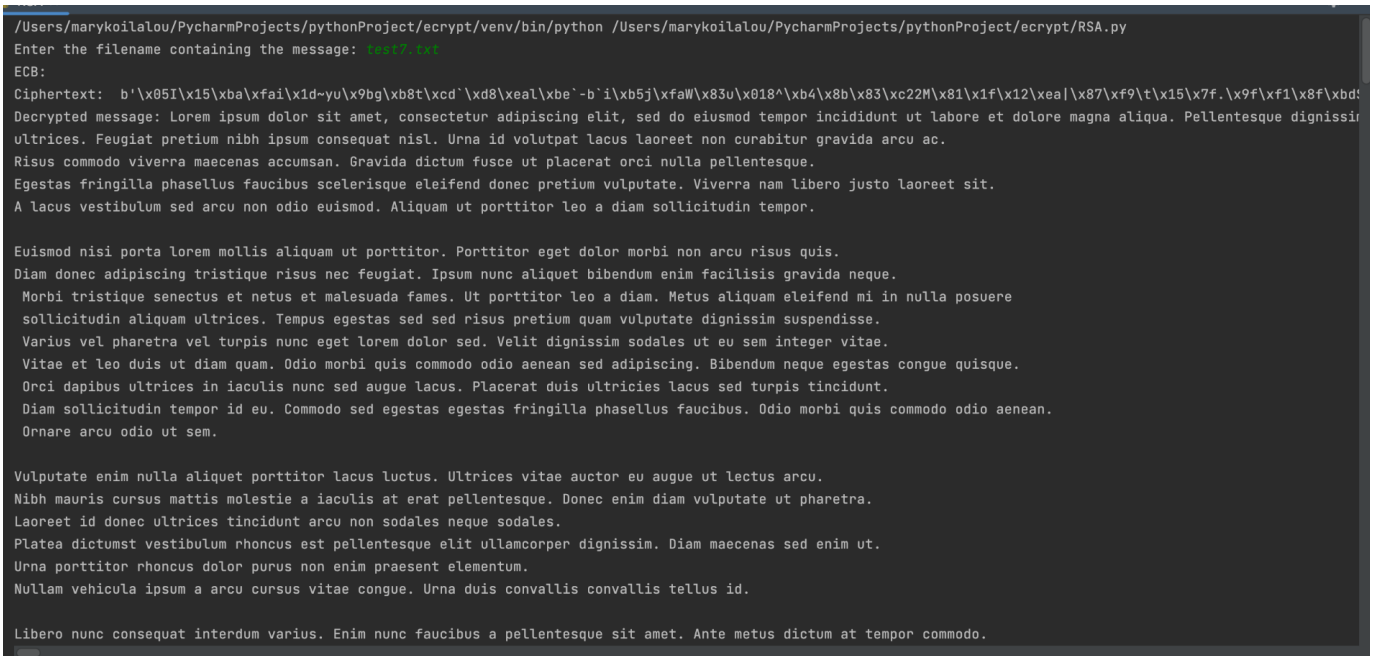
- test7.txt

Input:

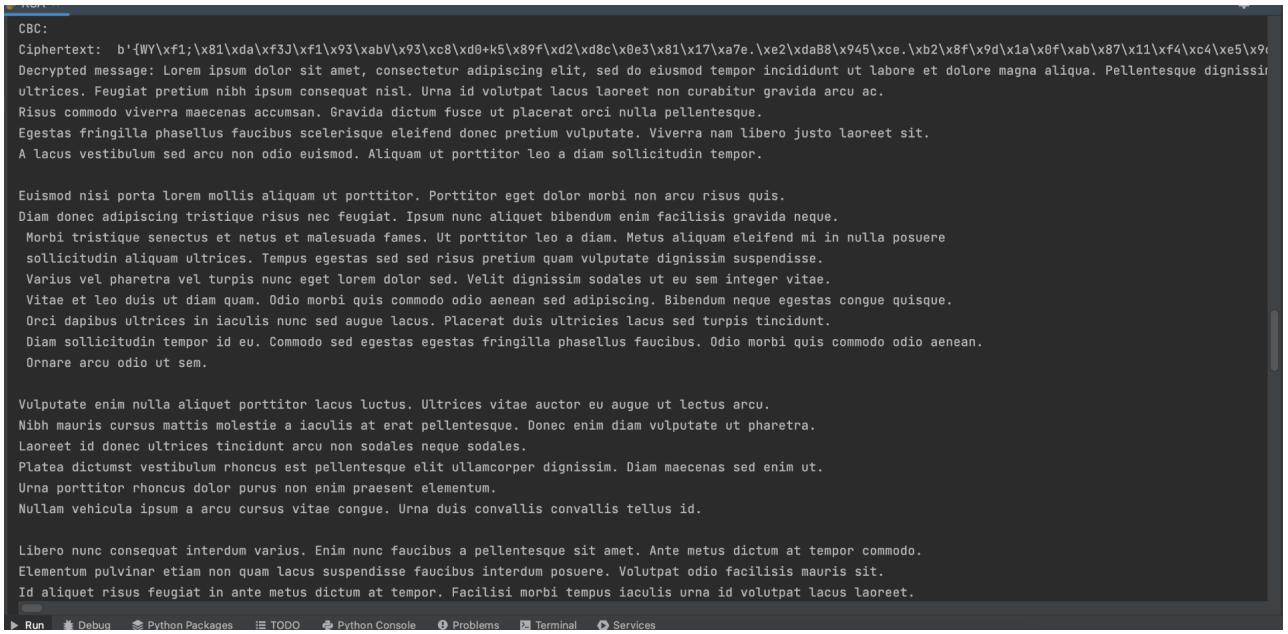


```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna al
ultrices. Feugiat pretium nibh ipsum consequat nisl. Urna id volutpat lacus laoreet non curabitur gravida arcu ac.
Risus commodo viverra maecenas accumsan. Gravida dictum fusce ut placerat orci nulla pellentesque.
Egestas fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate. Viverra nam libero justo laoreet sit.
A lacus vestibulum sed arcu non odio euismod. Aliquam ut porttitor leo a diam sollicitudin tempor.

Euismod nisi porta lorem mollis aliquam ut porttitor. Porttitor eget dolor morbi non arcu risus quis.
Diam donec adipiscing tristique risus nec feugiat. Ipsum nunc aliquet bibendum enim facilisis gravida neque.
 Morbi tristique senectus et netus et malesuada fames. Ut porttitor leo a diam. Metus aliquam eleifend mi in nulla posuere
 sollicitudin aliquam ultrices. Tempus egestas sed sed risus pretium quam vulputate dignissim suspendisse.
 Varius vel pharetra vel turpis nunc eget lorem dolor sed. Velit dignissim sodales ut eu sem integer vitae.
 Vitae et leo duis ut diam quam. Odio morbi quis commodo odio aenean sed adipiscing. Bibendum neque egestas congue quisque.
 Orci dapibus ultrices in iaculis nunc sed augue lacus. Placerat duis ultricies lacus sed turpis tincidunt.
 Diam sollicitudin tempor id eu. Commodo sed egestas egestas fringilla phasellus faucibus. Odio morbi quis commodo odio aenean.
 Ornare arcu odio ut sem.

Vulputate enim nulla aliquet porttitor lacus luctus. Ultrices vitae auctor eu augue ut lectus arcu.
Nibh mauris cursus mattis molestie a iaculis at erat pellentesque. Donec enim diam vulputate ut pharetra.
Laoreet id donec ultrices tincidunt arcu non sodales neque sodales.
Platea dictumst vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Diam maecenas sed enim ut.
Urna porttitor rhoncus dolor purus non enim praesent elementum.
Nullam vehicula ipsum a arcu cursus vitae congue. Urna duis convallis convallis tellus id.

Libero nunc consequat interdum varius. Enim nunc faucibus a pellentesque sit amet. Ante metus dictum at tempor commodo.
Elementum pulvinar etiam non quam lacus suspendisse faucibus interdum posuere. Volutpat odio facilisis mauris sit.
Id aliquet risus feugiat in ante metus dictum at tempor. Facilisi morbi tempus iaculis urna id volutpat lacus laoreet.
Risus sed vulputate odio ut. Risus quis varius quam quisque id. Duis at tellus at urna condimentum mattis.
Eget lorem dolor sed viverra ipsum nunc. Imperdiet sed euismod nisi porta lorem mollis aliquam ut porttitor.
```

Output:



```
/Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/venv/bin/python /Users/marykoilalou/PycharmProjects/pythonProject/ecrypt/RSA.py
Enter the filename containing the message: test7.txt
ECB:
Ciphertext:  b'\x05I\x15\xba\xfai\x1d~yu\x9bg\xb8t\xcd`\xd8\xeal\xbe`-b`i\xb5j\xfaW\x83u\x018^\xb4\x8b\x83\xc22M\x81\x1f\x12\xea|\x87\xf9\t\x15\x7f.\x9f\xf1\x8f\xbd
Decrypted message: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Pellentesque dignissi
ultrices. Feugiat pretium nibh ipsum consequat nisl. Urna id volutpat lacus laoreet non curabitur gravida arcu ac.
Risus commodo viverra maecenas accumsan. Gravida dictum fusce ut placerat orci nulla pellentesque.
Egestas fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate. Viverra nam libero justo laoreet sit.
A lacus vestibulum sed arcu non odio euismod. Aliquam ut porttitor leo a diam sollicitudin tempor.

Euismod nisi porta lorem mollis aliquam ut porttitor. Porttitor eget dolor morbi non arcu risus quis.
Diam donec adipiscing tristique risus nec feugiat. Ipsum nunc aliquet bibendum enim facilisis gravida neque.
 Morbi tristique senectus et netus et malesuada fames. Ut porttitor leo a diam. Metus aliquam eleifend mi in nulla posuere
 sollicitudin aliquam ultrices. Tempus egestas sed sed risus pretium quam vulputate dignissim suspendisse.
 Varius vel pharetra vel turpis nunc eget lorem dolor sed. Velit dignissim sodales ut eu sem integer vitae.
 Vitae et leo duis ut diam quam. Odio morbi quis commodo odio aenean sed adipiscing. Bibendum neque egestas congue quisque.
 Orci dapibus ultrices in iaculis nunc sed augue lacus. Placerat duis ultricies lacus sed turpis tincidunt.
 Diam sollicitudin tempor id eu. Commodo sed egestas egestas fringilla phasellus faucibus. Odio morbi quis commodo odio aenean.
 Ornare arcu odio ut sem.

Vulputate enim nulla aliquet porttitor lacus luctus. Ultrices vitae auctor eu augue ut lectus arcu.
Nibh mauris cursus mattis molestie a iaculis at erat pellentesque. Donec enim diam vulputate ut pharetra.
Laoreet id donec ultrices tincidunt arcu non sodales neque sodales.
Platea dictumst vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Diam maecenas sed enim ut.
Urna porttitor rhoncus dolor purus non enim praesent elementum.
Nullam vehicula ipsum a arcu cursus vitae congue. Urna duis convallis convallis tellus id.

Libero nunc consequat interdum varius. Enim nunc faucibus a pellentesque sit amet. Ante metus dictum at tempor commodo.
```



References

https://delta.cs.cinvestav.mx/~francisco/cripto/modes.htm