**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.

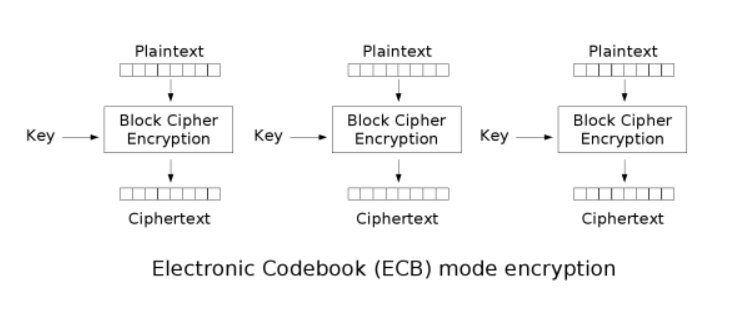
To generate the public and private keys the following process is used:

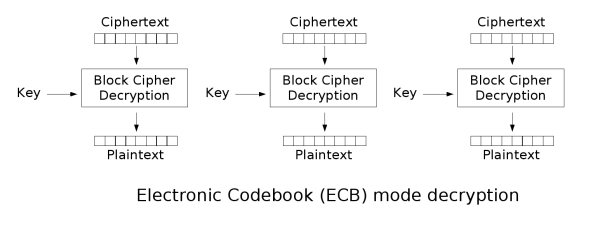
* Generate two large prime numbers p, q
* Calculate n=p\*q and φ(n)=(p-1)\*(q-1)
* Choose a random number e, coprime to φ(n)
* Calculate d as the modular inverse of e modulo φ(n)
* The public key is (n, e) and the private key is (n, d)

The cyphertext of the transmitted message m is calculated as: c=me(modn) and the decrypted message is m=cd(modn).

**ECB mode**

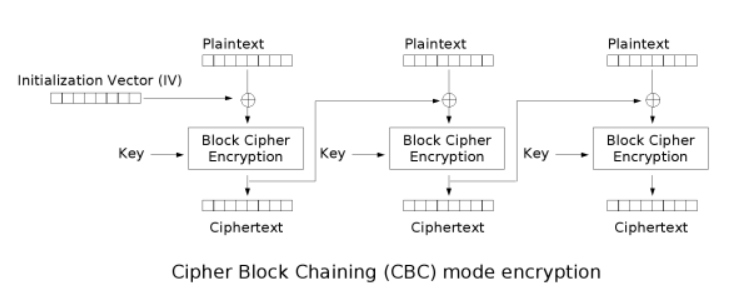
In this mode the transmitted message for the encryption is divided into blocks. Each block then is encrypted using the above RSA algorithm. For given keys, the result of the algorithm is deterministic. In the decryption each block of the ciphertext is used to generate the corresponding plaintext message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

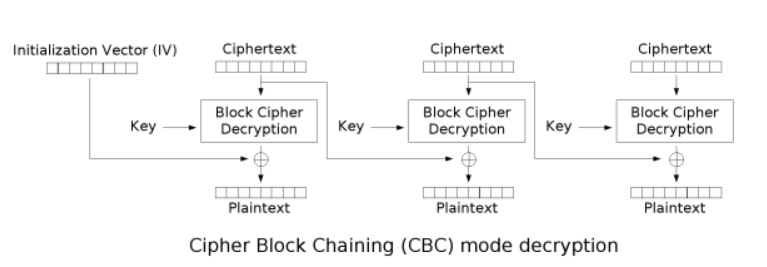




**CBC mode**

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.





**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.

* 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.

* 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).

* 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= me(modn)). 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=cd(modn)). The result is transformed into byte format, then it’s unpadded and finally it is transformed back to string using the UTF-8 encoding.

* CBC mode:

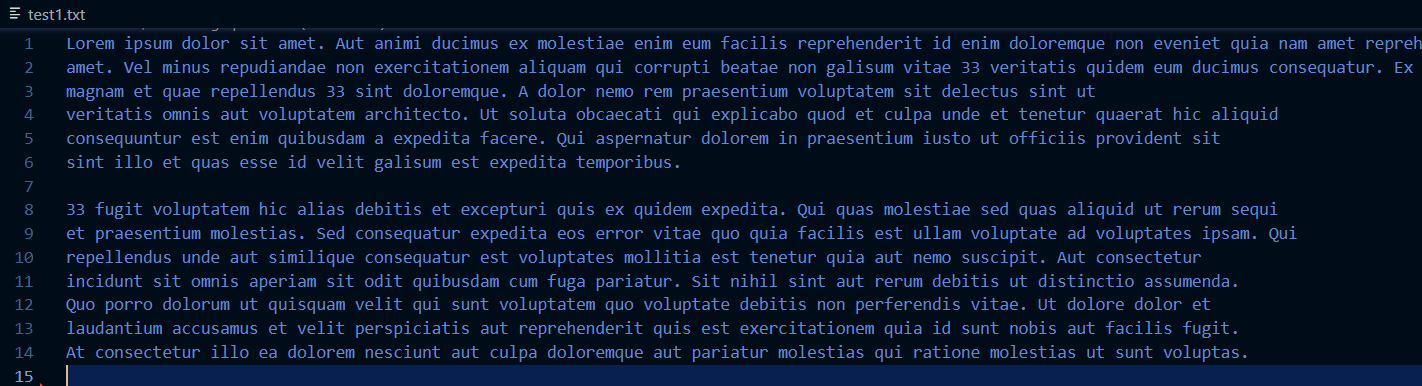
The general idea is the same, but now a random initialisation vector is used as the first block of the ciphertext. The following blocks of the ciphertext are calculated by XORing the corresponding block of the message with the previous ciphertext block. In similar way the decrypted message is calculated.

**Testing**

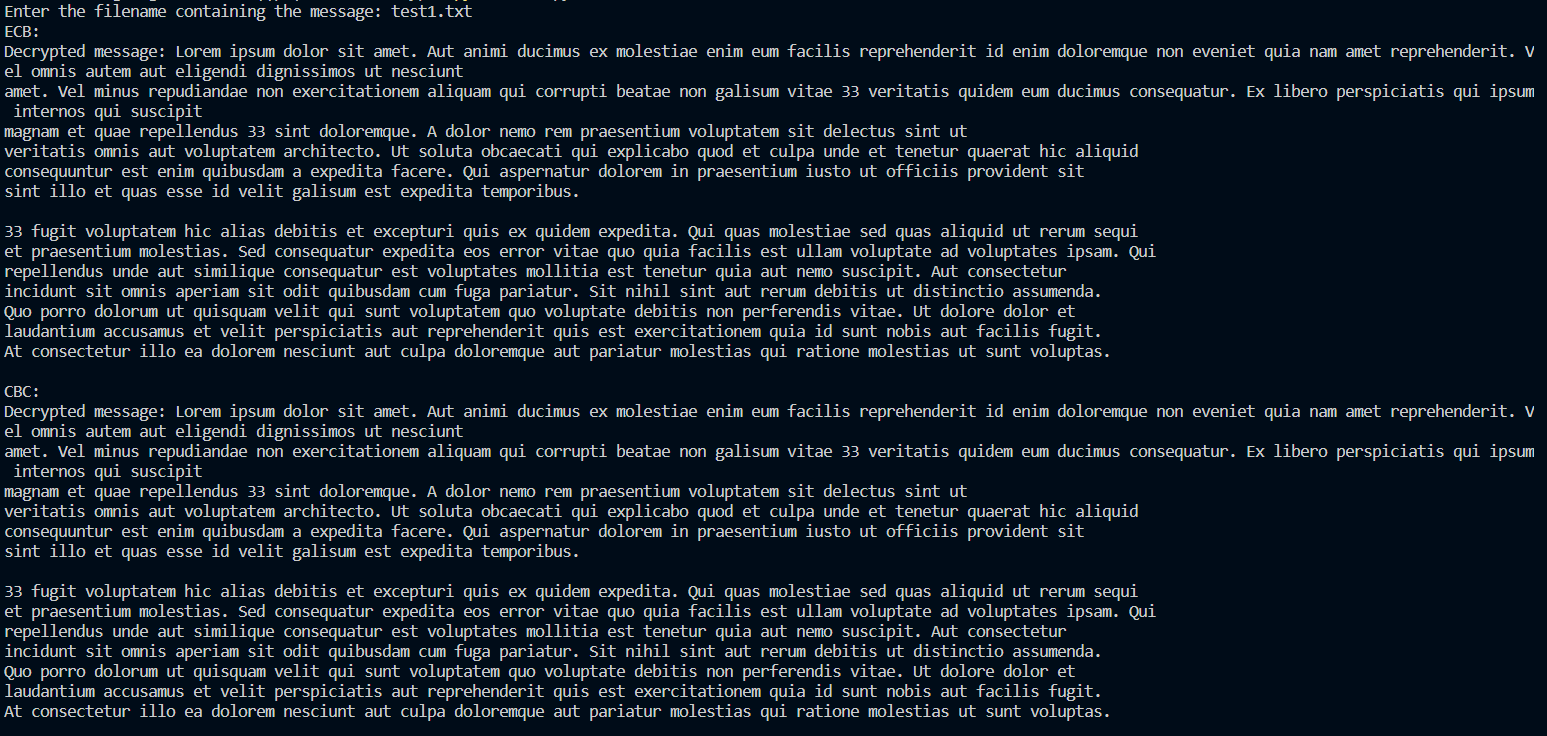
We created 7 files with different data (ex. Chinese or Greek characters, numbers, emojis) and we want to check if the result of the encryption followed by the decryption is the same as the initial text.

* test1.txt

Input:



Output:

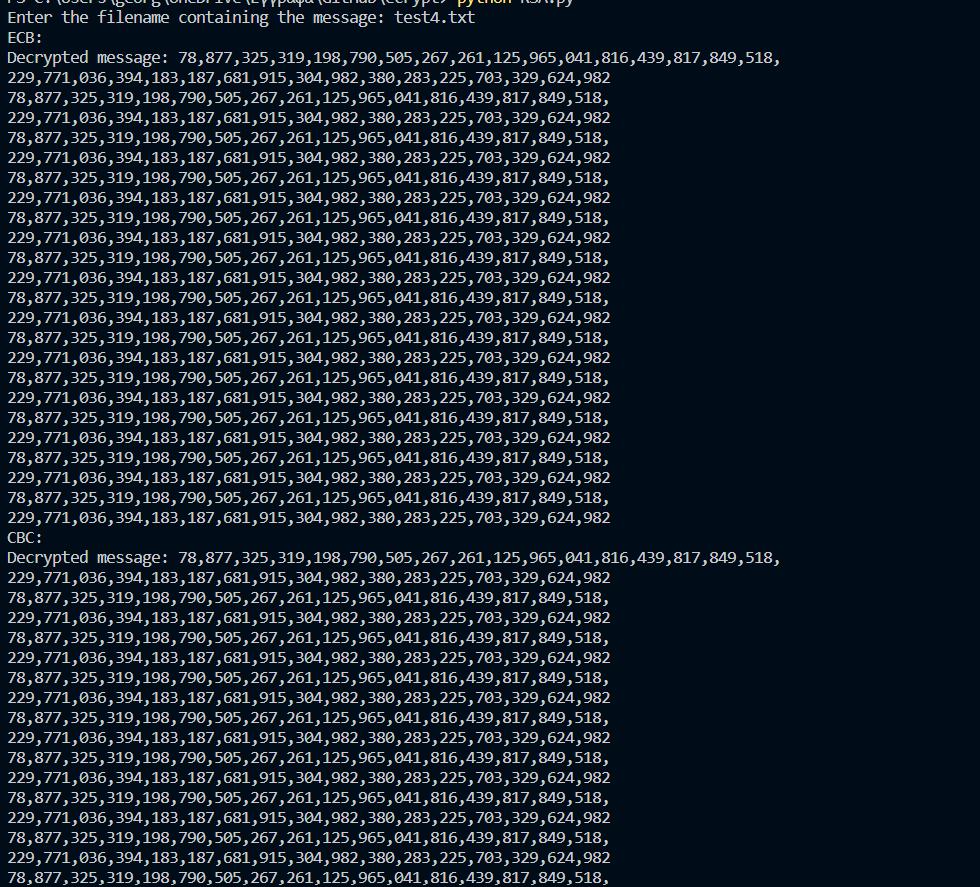


* test4.txt

Input:



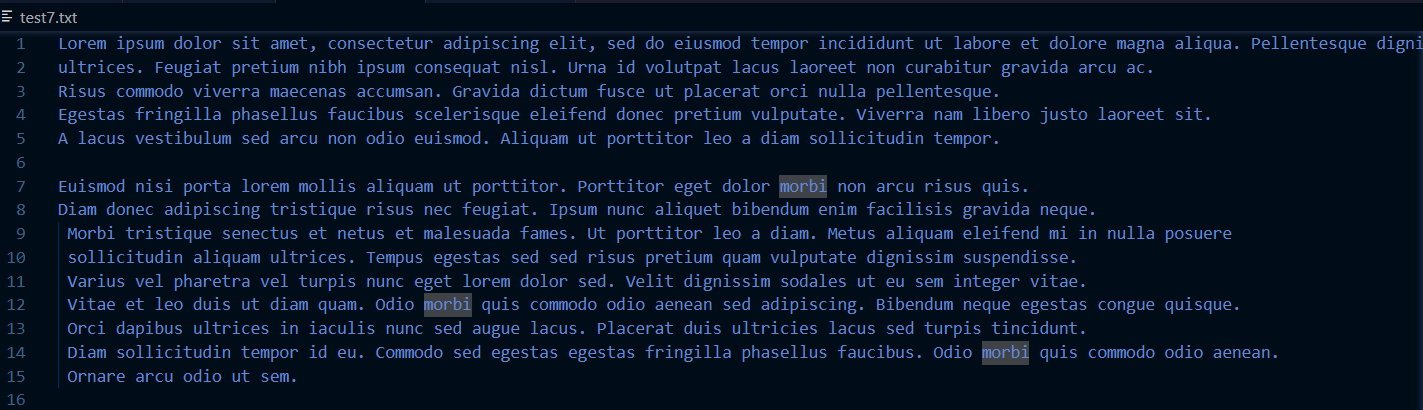
Output:



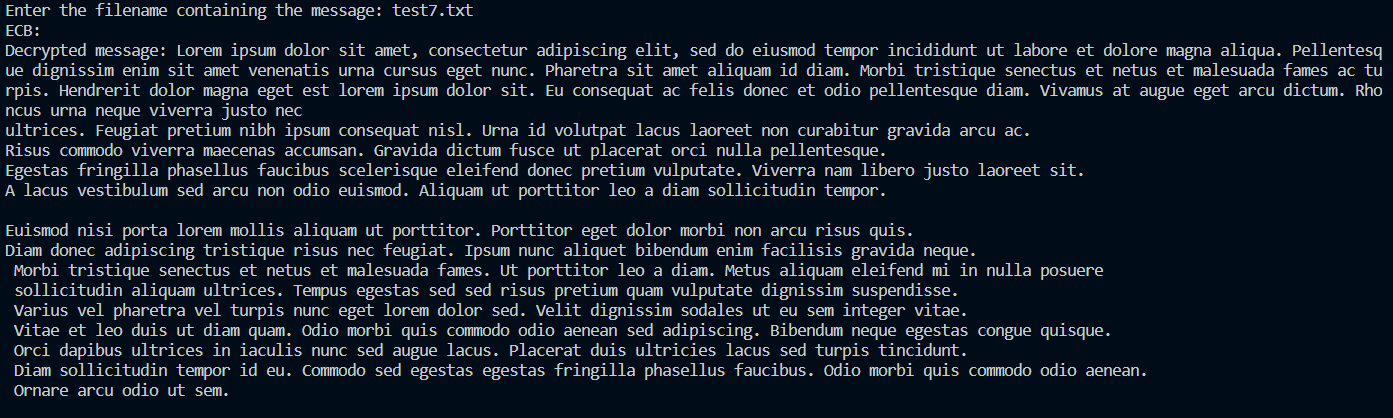
* test7.txt

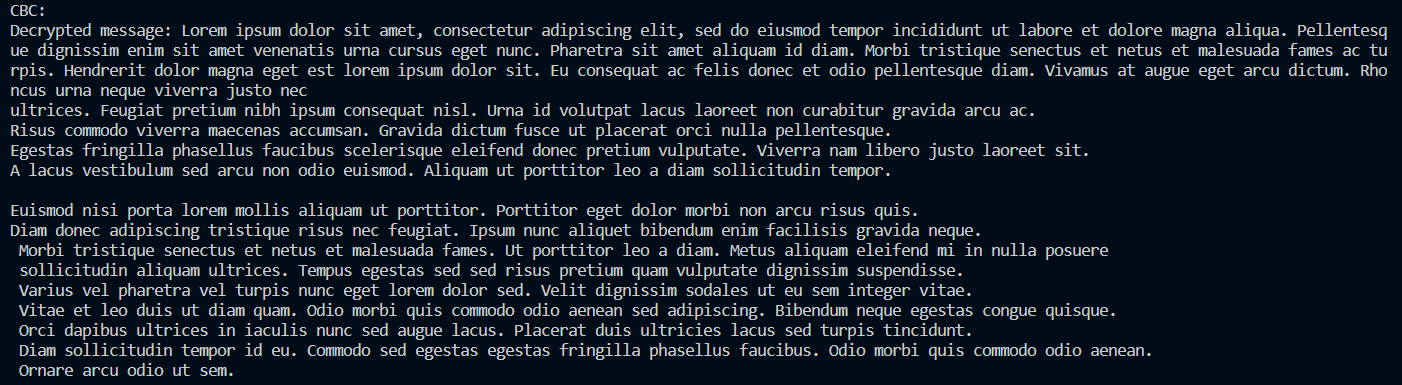
(is a big text, only parts of input and output are shown)

Input:



Output:





References

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