SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES

Computer Security
Semester 2024-I
Impartido por Prof. Cristhian Iza, PhD

UNIVERSIDAD YACHAY TECH

LAB 2: Symmetric Key

Dilan Coral[1]
School of Mathematical and Computational Science
Universidad Yachay Tech, Urcuquí - Ecuador

March 08, 2024

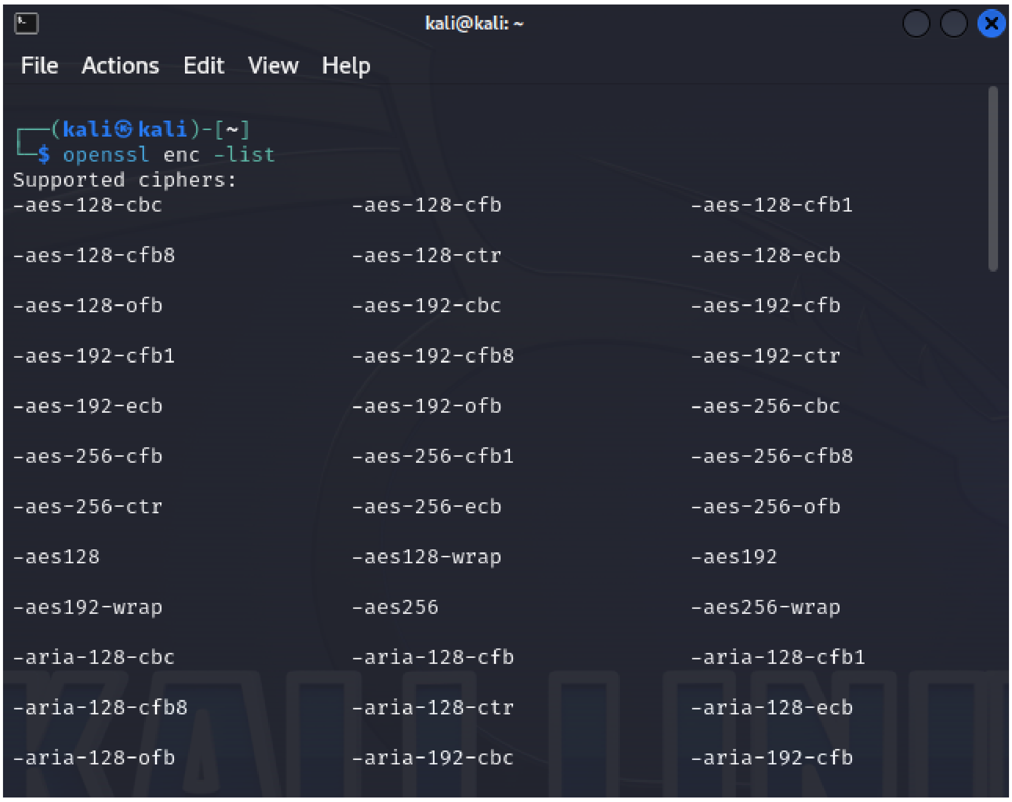
You can find the code for this project on GitHub. Here is the repository link:

- **Repository**: https://github.com/Lelis10/Computer-Security---YT.git

1 Exercise 1: Base-64 encoding, hexadecimal representation, and modulus operator.

1. Outline five encryption methods supported by OpenSSL

```
openssl enc -list
```

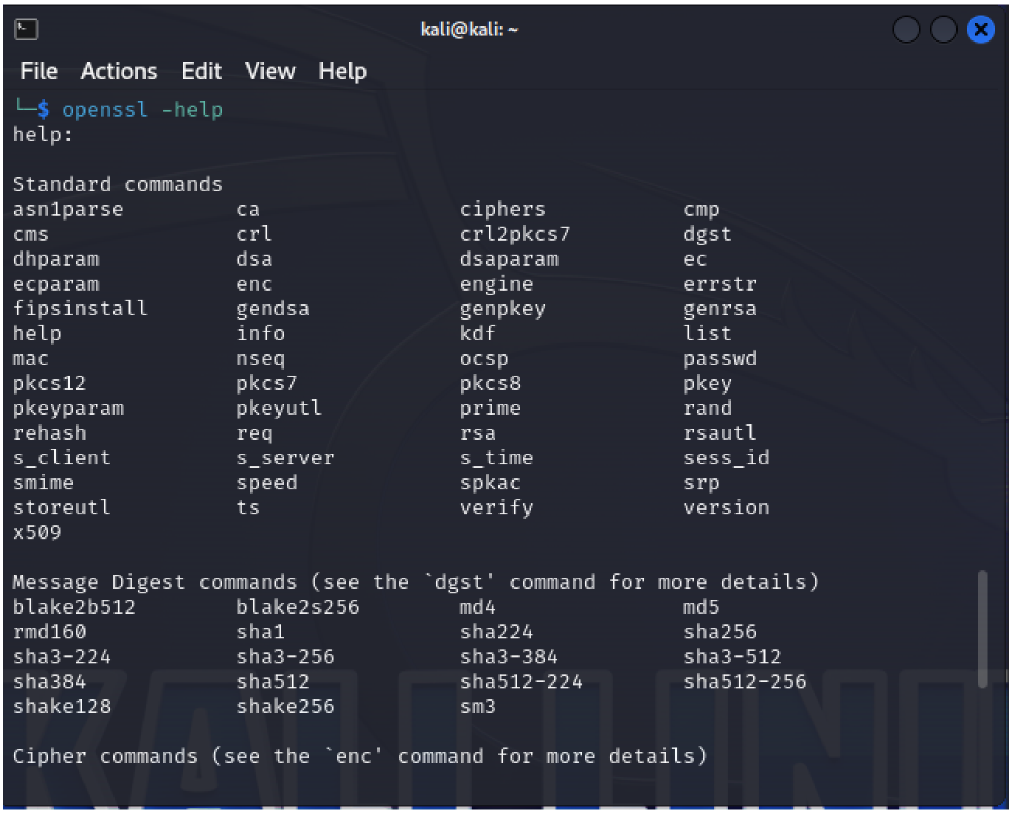


2. Outline the version of OpenSSL

```
openssl version
```

3. Outline the help of OpenSSL

```
openssl -help
```



4. Checking if given numbers are prime

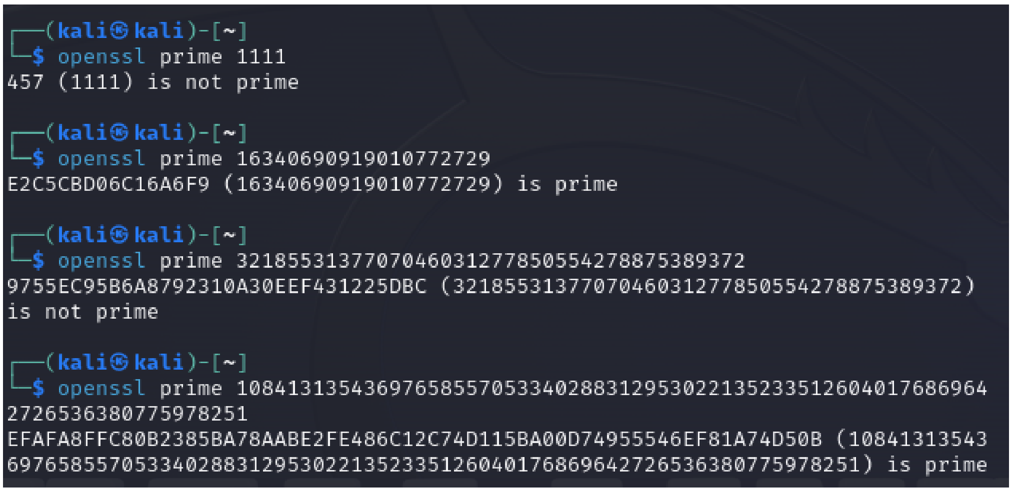
```
openssl prime 1111
```

check if the following are prime numbers:

(a) 16340690919010772729

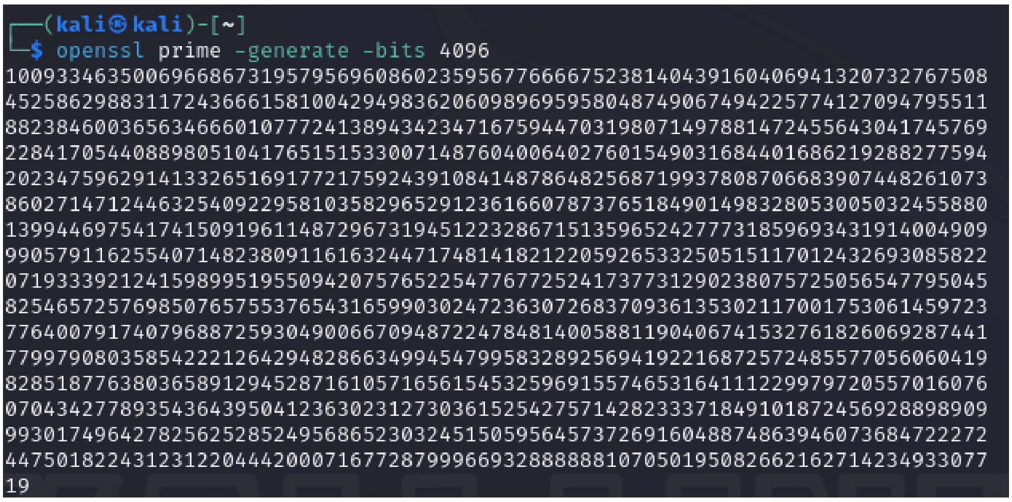
(b) 3218553137707046031277850554278875389372

(c) 108413135436976585570533402883129530221352335126040176869642726536380775978251



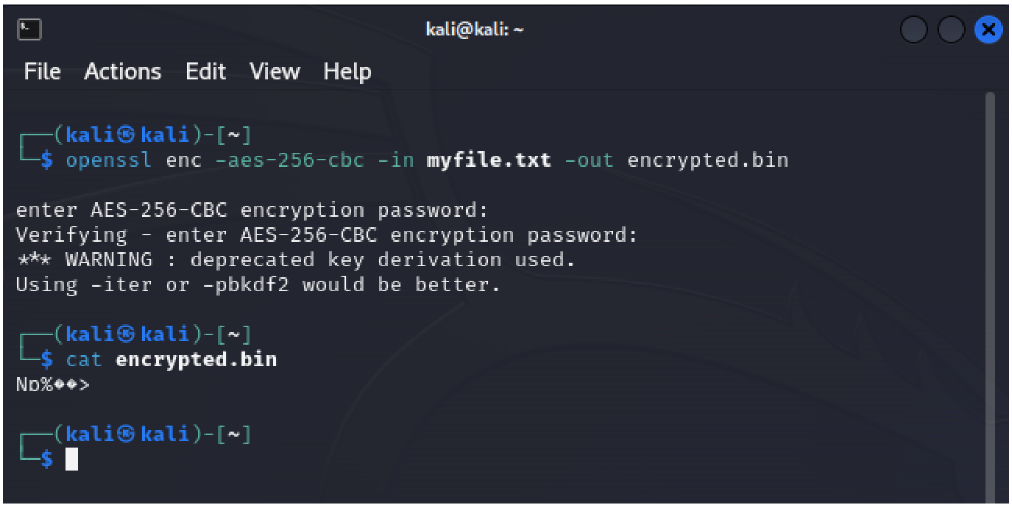
5. Generating a random prime number of 4096 bits

```
openssl prime -generate -bits 4096
```



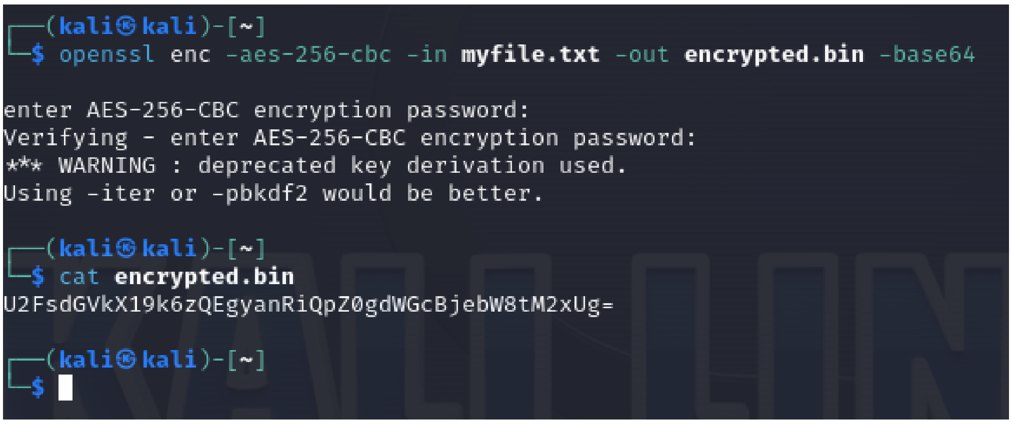
6. Encrypting a file with aes-256-cbc

```
openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin
```



7. Encrypting a file with aes-256-cbc and base64 encoding

```
openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin -base64
```

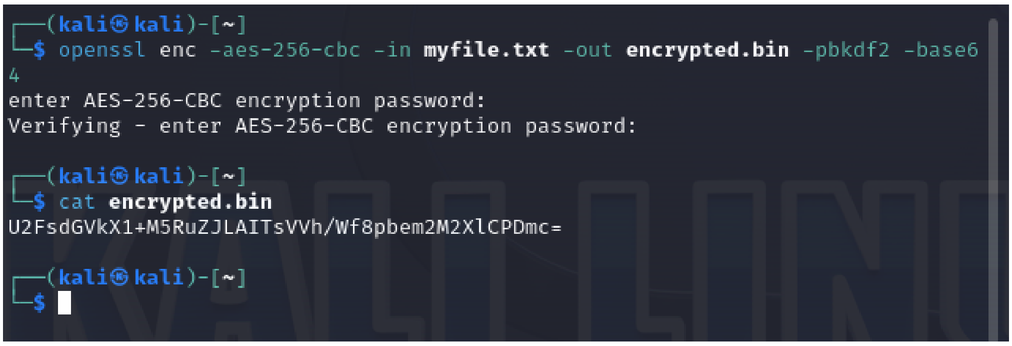


8. Encrypting a file with aes-256-cbc, pbkdf2, and base64 encoding

```
openssl enc -aes-256-cbc -in myfile.txt -out encrypted.bin -pbkdf2 -base64
```

(b) Has the output changed? [Yes][No]

(c) Why has it changed?



(b) Yes, the output has changed.

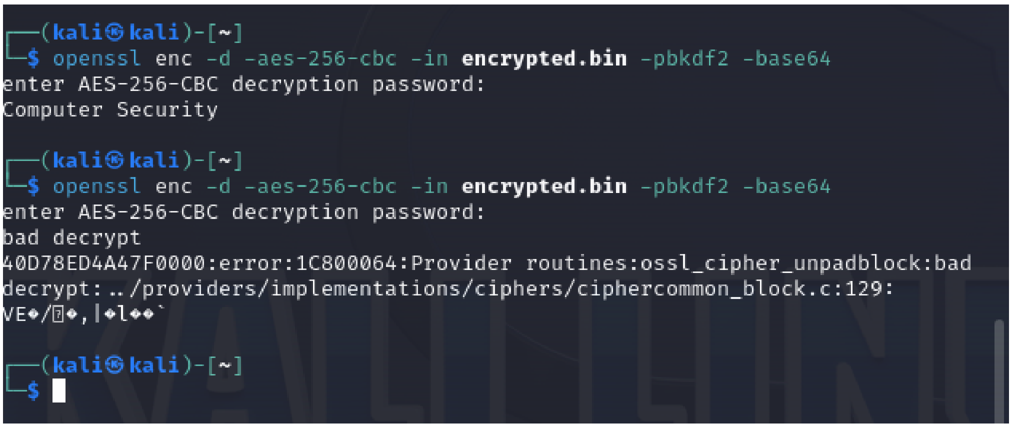
(c) The output has changed because using PBKDF2 for key derivation introduces additional security measures compared to the default key derivation method. PBKDF2 (Password-Based Key Derivation Function 2) is a key stretching algorithm that iterates a cryptographic hash function multiple times to generate a strong, derived key from a password. This increases the computational complexity, making it harder for attackers to perform brute-force attacks against the encrypted data. Additionally, base64 encoding converts binary data into ASCII text, which can change the representation of the encrypted output.

9. Decrypting the encrypted file

```
openssl enc -d -aes-256-cbc -in encrypted.bin -pbkdf2 -base64
```

(b) Has the output been decrypted correctly? Yes, it decrypt correctly.

(c) What happens when you use the wrong password? The command prompt throws an error.



10. Encrypting a file with Blowfish and attempting decryption

```
# Encrypt with Blowfish
openssl enc -bf -in myfile.txt -out encrypted_blowfish.bin

# Attempt decryption
openssl enc -d -bf -in encrypted_blowfish.bin -out decrypted_blowfish.txt
```

(a) Did you manage to decrypt the file? Yes

11. Encrypting a file with 3DES and attempting decryption

```
# Encrypt with 3DES
openssl enc -des3 -in myfile.txt -out encrypted_3des.bin

# Attempt decryption
openssl enc -d -des3 -in encrypted_3des.bin -out decrypted_3des.txt
```

(a) Did you manage to decrypt the file? Yes

12. Encrypting a file with RC2 and attempting decryption

```
# Encrypt with RC2
openssl enc -rc2 -in myfile.txt -out encrypted_rc2.bin

# Attempt decryption
openssl enc -d -rc2 -in encrypted_rc2.bin -out decrypted_rc2.txt
```

(a) Did you manage to decrypt the file? Yes

2 Padding (AES)

2.1 Installation of Cryptographic Libraries for AES-256 Encryption

Install the necessary cryptographic libraries for AES-256 encryption.

2.2 Block Size for AES-256 Encryption

With AES which uses a 256-bit key, what is the normal block size (in bytes).

1. Block size (bytes): 16 bytes
2. Number of hex characters for block size: 32 hex characters (each byte represented by 2 hex characters)

2.3 Implementation of AES-256 Encryption with Different Padding Schemes

Demonstrate the implementation of AES-256 encryption with different padding schemes using Python.

2.4 Encryption and Decryption Tests

Perform encryption and decryption tests using sample data and various padding schemes.

Lab2 > exercise_2.py > aes_256_decrypt

```python

def aes_256_encrypt(data, key, padding_method):
    cipher = AES.new(key, AES.MODE_CBC)
    if padding_method == 'zero':
        padded_data = zero_pad(data.encode(), AES.block_size)
    else:
        padded_data = pad(data.encode(), AES.block_size, style=padding_method)
    ciphertext = cipher.encrypt(padded_data)
    return ciphertext, cipher.iv

def aes_256_decrypt(ciphertext, key, iv, padding_method):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = cipher.decrypt(ciphertext)
    if padding_method == 'zero':
        unpadded_data = zero_unpad(decrypted_data)
    else:
        unpadded_data = unpad(decrypted_data, AES.block_size, style=padding_method)
    return unpadded_data.decode()
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
Padding method: PKCS7
Encrypted data: b0c51a24ac4f7235f816e8fb688e36a2
Decrypted data: Hello, World!

Padding method: X923
Encrypted data: caa48486eabac9372b9a9e8baa9bdf52
Decrypted data: Hello, World!

Padding method: ISO7816
Encrypted data: 276cf995bc3e84752115fc5b7e628c99
Decrypted data: Hello, World!

Padding method: ZERO
Encrypted data: f396f5e6d5bf2877c83a48d31953cdea
Decrypted data: Hello, World!

PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security>
```

3 Padding (DES)

3.1 With DES which uses a 64-bit key, what is the normal block size (in bytes):

(a) Block size (bytes): 8 bytes

(b) Number of hex characters for block size: 16 hex characters (each byte represented by 2 hex characters)

3.2 Demonstrate the implementation of DES encryption with different padding schemes using Python.

3.3 Perform encryption and decryption tests using sample data and various padding schemes.

Lab2 > exercise_3.py > des_encrypt

```python
from Crypto.Cipher import DES
from Crypto.Util.Padding import pad
from Crypto.Util.Padding import unpad
import binascii

def des_encrypt(data, key, padding_method):
    cipher = DES.new(key, DES.MODE_ECB)
    if padding_method == 'zero':
        padded_data = zero_pad(data.encode(), DES.block_size)
    else:
        padded_data = pad(data.encode(), DES.block_size, style=padding_method)
    ciphertext = cipher.encrypt(padded_data)
    return ciphertext

def des_decrypt(ciphertext, key, padding_method):
    cipher = DES.new(key, DES.MODE_ECB)
    decrypted_data = cipher.decrypt(ciphertext)
    if padding_method == 'zero':
        unpadded_data = zero_unpad(decrypted_data)
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
ts/YT/Informatic Security/Lab2/exercise_3.py"
Padding method: PKCS7
Encrypted data: 5b22ead96e11ff28ebc50c405e0ae518
Decrypted data: Hello, DES!

Padding method: X923
Encrypted data: 5b22ead96e11ff28a2764f52fae3c67f
Decrypted data: Hello, DES!

Padding method: ISO7816
Encrypted data: 5b22ead96e11ff285fc68e06bc4efc46
Decrypted data: Hello, DES!

Padding method: ZERO
Encrypted data: 5b22ead96e11ff28f5e72c004e8ea62b
Decrypted data: Hello, DES!

PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security>
```

4 Python Coding (Encrypting)

4.1 . Update the code so that you can enter a string and the program will show the cipher text. The format will be something like:

python cipher01.py hello mykey

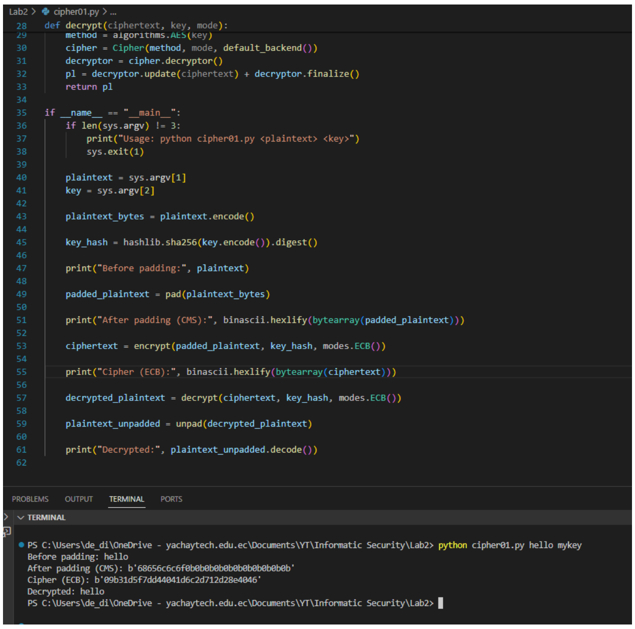


Figura 1: Enter Caption

4.2 Now determine the cipher text for the following (the first example has already been completed):

Message	Key CMS	Cipher
hello	hello123	0a7ec77951291795bac6690c9e7f4c0d
Security	orange	2e5b2dc7f30b60fa64afd3a6fd478797
YachayTech	university	6c7316eed2c54f7ee50d8e7d5ca81d76
Ecuador	emerald	50f58241232c031f8b0cf7d862e58067

4.3 Copy your code and modify it so that it implements 64-bit DES and complete the table

Message	Key CMS	Cipher
hello	hello123	0a7ec77951291795bac6690c9e7f4c0d
Security	orange	b5ff774b731cc863e63dc05c64f27a62
YachayTech	university	0461efcfa6cb5b03c838ead7ef8e4dea
Ecuador	emerald	4c0e14b8c6d78bc0

4.4 Modify the code so that the user can enter the values from the keyboard, such as with:

cipher=raw_input('Enter cipher:') # password=raw_input('Enter password:')

Lab2 > cipher03.py > ...

```python
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.backends import default_backend

import hashlib
import binascii

def pad(data, size=128):
    padder = padding.PKCS7(size).padder()
    padded_data = padder.update(data)
    padded_data += padder.finalize()
    return padded_data

def unpad(data, size=128):
    padder = padding.PKCS7(size).unpadder()
    unpadded_data = padder.update(data)
    unpadded_data += padder.finalize()
    return unpadded_data
```

PROBLEMS OUTPUT TERMINAL PORTS

TERMINAL

```
PS C:\Users\de_di\O& C:/Users/de_di/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/de_d
er03.py"
Enter plaintext: DilanCoral
Enter key: hola
Before padding: DilanCoral
After padding (CMS): b'44696c616e436f72616c060606060606'
Cipher (ECB): b'5710a7d04dea51416c7c7332a3c7c99c'
Decrypted: DilanCoral
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2>
```

5 Python Coding (Decrypting)

5.1 Modify your coding for 256-bit AES ECB encryption, so that you can enter the cipher text, and an encryption key, and the code will decrypt to provide the result. You should use CMS for padding. With this, determine the plaintext for the following:

CMS Cipher (256-bit AES ECB)	key	Plain text
b436bd84d16db330359edebf49725c62	hello	germany
4bb2eb68fccd6187ef8738c40de12a6b	ankle	spain
029c4dd71cdae632ec33e2be7674cc14	changeme	england
d8f11e13d25771e83898efdbad0e522c	123456	scotland

Tabla 1: CMS Cipher (256-bit AES ECB) Key and Plain text

```
Enter the cipher text (hexadecimal): b436bd84d16db330359edebf49725c62
Enter the encryption key: hello
Decrypted: germany
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:/
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrpyt01.py"
Enter the cipher text (hexadecimal): 4bb2eb68fccd6187ef8738c40de12a6b
Enter the encryption key: ankle
Decrypted: spain
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:/
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrpyt01.py"
Enter the cipher text (hexadecimal): 029c4dd71cdae632ec33e2be7674cc14
Enter the encryption key: changeme
Decrypted: england
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:/
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrpyt01.py"
Enter the cipher text (hexadecimal): d8f11e13d25771e83898efdbad0e522c
Enter the encryption key: 123456
Decrypted: scotland
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2>
```

5.2 Modify your coding for 64-bit DES ECB encryption, so that you can enter the cipher text, and an encryption key, and the code will decrypt to provide the result. You should use CMS for padding. With this, determine the plaintext for the following:

CMS Cipher (64-bit DES ECB)	key	Plain text
f37ee42f2267458d	hello	Germany
67b7d1162394b868	ankle	France
ac9feb702ba2ecc0	changeme	Norway
de89513fbd17d0dc	123456	England

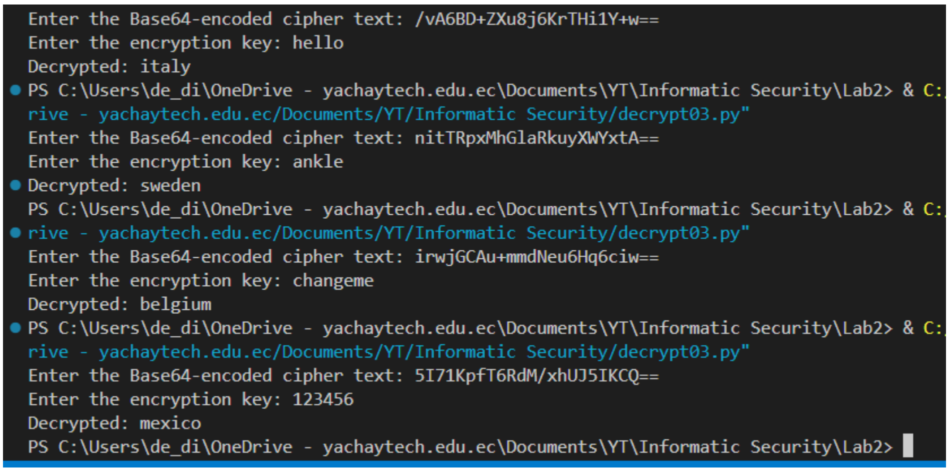
Tabla 2: CMS Cipher (64-bit DES ECB) Key and Plain text

```
Enter the cipher text (hexadecimal): f37ee42f2267458d
Enter the encryption key: hello
Decrypted: Germany
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> &
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt02.py"
Enter the cipher text (hexadecimal): 67b7d1162394b868
Enter the encryption key: ankle
Decrypted: France
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> &
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt02.py"
Enter the cipher text (hexadecimal): ac9feb702ba2ecc0
Enter the encryption key: changeme
Decrypted: Norway
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> &
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt02.py"
Enter the cipher text (hexadecimal): de89513fbd17d0dc
Enter the encryption key: 123456
Decrypted: England
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2>
```

5.3 Update your program, so that it takes a cipher string in Base-64 and converts it to a hex string and then decrypts it. From this now decrypt the following Base-64 encoded cipher streams (Remember to add import base64).

CMS Cipher (256-bit AES ECB)	key	Plain text
/vA6BD+ZXu8j6KrTHi1Y+w==	hello	italy
nitTRpxMhGlaRkuyXWYxtA==	ankle	sweden
irwjGCAu+mmdNeu6Hq6ciw==	changeme	belgium
5I71KpfT6RdM/xhUJ5IKCQ==	123456	mexico

Tabla 3: CMS Cipher (64-bit DES ECB) Key and Plain text



```
Enter the Base64-encoded cipher text: /vA6BD+ZXu8j6KrTHi1Y+w==
Enter the encryption key: hello
Decrypted: italy
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt03.py"
Enter the Base64-encoded cipher text: nitTRpxMhGlaRkuyXWYxtA==
Enter the encryption key: ankle
Decrypted: sweden
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt03.py"
Enter the Base64-encoded cipher text: irwjGCAu+mmdNeu6Hq6ciw==
Enter the encryption key: changeme
Decrypted: belgium
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2> & C:
rive - yachaytech.edu.ec/Documents/YT/Informatic Security/decrypt03.py"
Enter the Base64-encoded cipher text: 5I71KpfT6RdM/xhUJ5IKCQ==
Enter the encryption key: 123456
Decrypted: mexico
PS C:\Users\de_di\OneDrive - yachaytech.edu.ec\Documents\YT\Informatic Security\Lab2>
```

6 Catching exceptions

6.1 Implement a Python program which will try various keys for a cipher text input, and show the decrypted text. The keys tried should be: [hello,ankle,changeme,123456]

6.2 Run the program and try to crack: 1jDmCTD1IfbXbyyHgAyrdg==

6.3 What is the password?

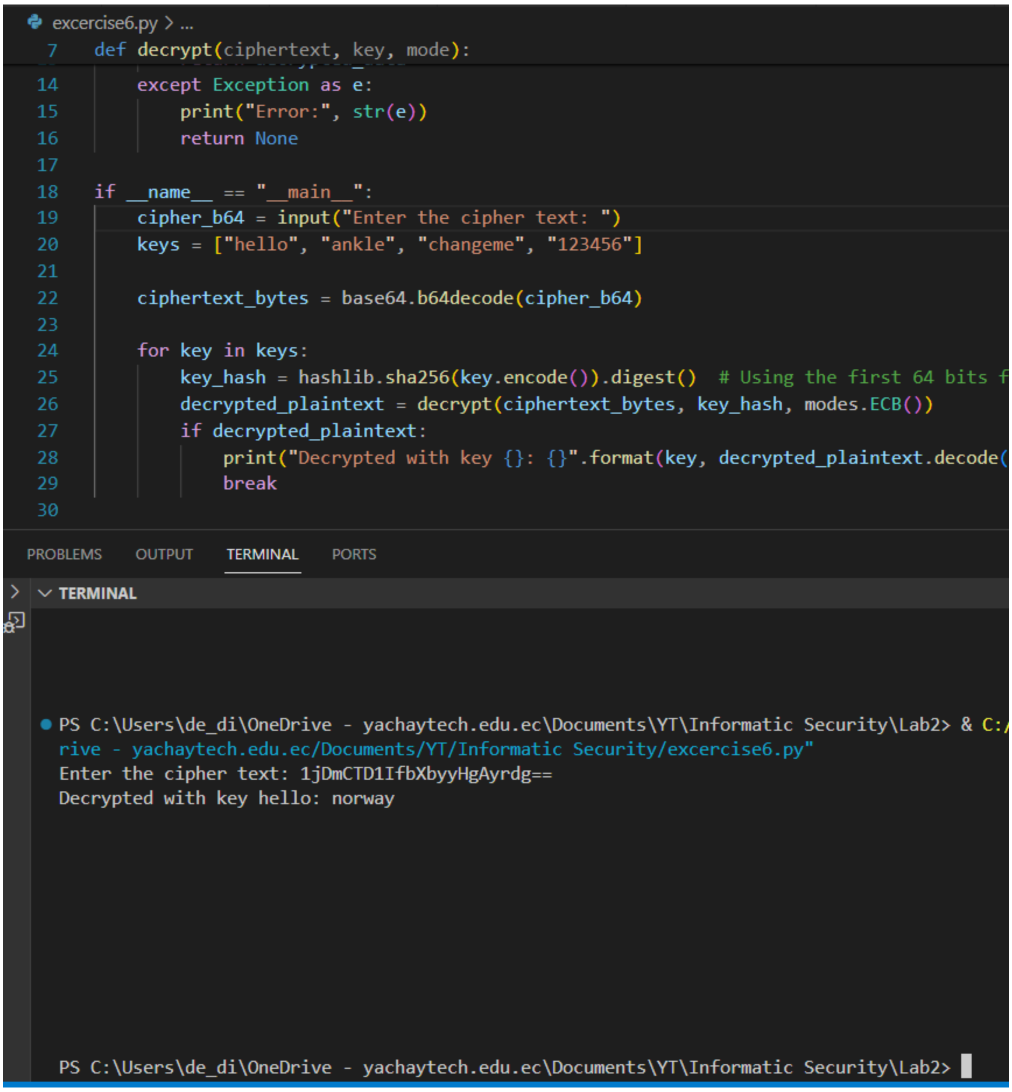


Figura 2: Enter Caption

7 Stream Ciphers

7.1 Develop an application in Python to implement the ChaCha20 stream cipher.

7.2 If we use a key of qwerty, nonce = 0x0000000000000000 can you find the well-known fruits (in lower case) of the following ChaCha20 cipher streams:

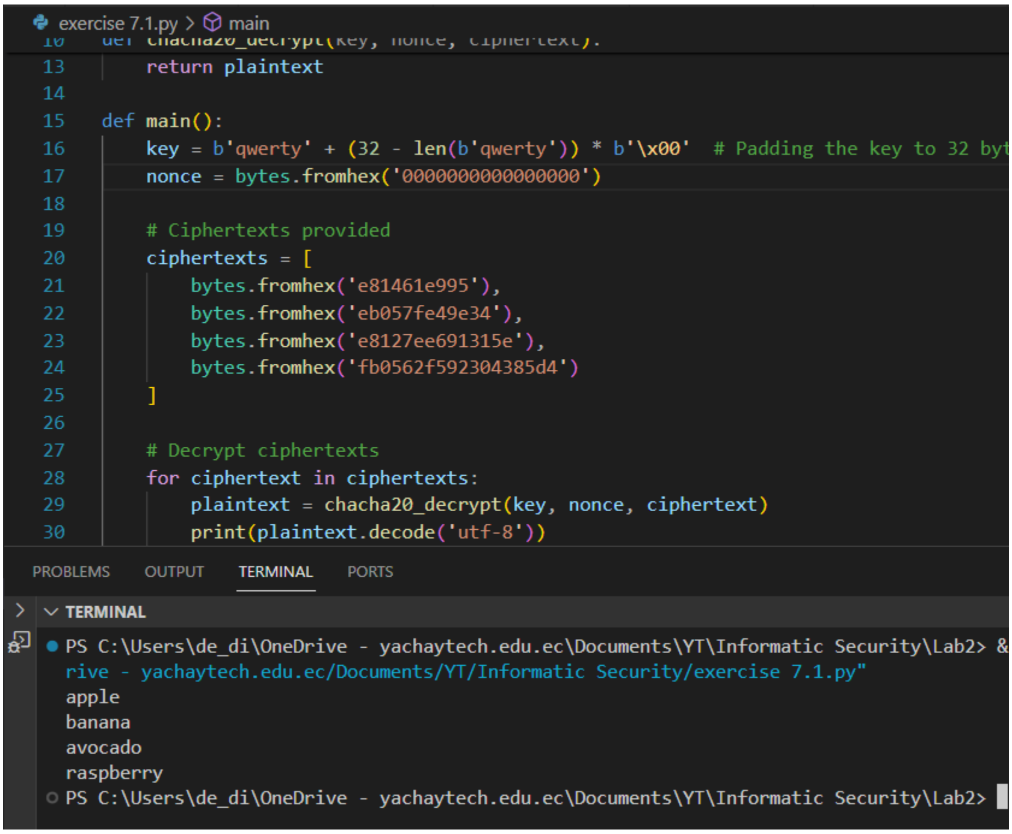


Figura 3: Enter Caption

7.3 RC4 is a standard stream cipher and can be used for light-weight cryptography. Develop an application in Python to implement the RC4 stream cipher.