

# UNIVERSIDAD NACIONAL DE QUILMES

## **SEGURIDAD INFORMÁTICA**

### LABORATORIO N°2: *Criptografía*



- **Alumna:**
  - Ruiz Camila
- **Profesores:**
  - Cipriano Marcelo
  - Sliafertas Matías

# Consigna

---

- La modalidad de entrega y de trabajo es individual
- **Objetivo:**
  - realizar un programa que tome por entrada un archivo cualquiera y devuelva el mismo archivo cifrado con el algoritmo AES-256.
  - Se podrá utilizar el lenguaje de programación preferido por el alumno.
  - Deberá entregarse un breve informe explicando el funcionamiento del código.
  - El informe deberá contener un anexo explicando cómo se deberá efectuar la ejecución.
- **Presentación:**
  - Deberá entregarse un archivo .zip con:
    - El código fuente y el ejecutable.
    - El informe en formato PDF.
- **Fecha de entrega:**
  - 27/11/2021

## INTRODUCCIÓN

Lenguaje elegido para mi programa: Python.

El programa va a tener el fin de recibir como parámetro el nombre de un archivo, encriptarlo y guardar el resultado en otro archivo. Para verificar que se encripte de forma correcta también se puede utilizar el programa para desencriptar el archivo encriptado, y validar que la desencriptación genere un archivo idéntico al original sin encriptar.

El programa hecho recibe 3 parámetros:

- "encrypt" o "decrypt" dependiendo de si se quiere encriptar o desencriptar respectivamente un archivo.
- el nombre del archivo a encriptar / desencriptar.
- la contraseña con la cual encriptar / desencriptar.

## EXPLICACIÓN DEL CÓDIGO

### ACCIÓN DE ENCRYPTAR:

Si la acción es encriptar se lee los bytes del archivo, se los encripta y por último se los escribe en un archivo nuevo con nombre "encrypted".

### ACCIÓN DE DESENCRIPTAR:

Si la acción es desencriptar se lee los bytes del archivo, se los desencripta y se los escribe en un archivo nuevo con nombre "decrypted".

Para esto, el código tiene 4 funciones:

#### 1. *read\_file\_bytes*

Recibe el nombre del archivo a leer. Después abre el archivo, lee los bytes y los retorna.

```
def read_file_bytes(filename):  
    in_file = open(filename, "rb")  
    bytes_read = in_file.read()  
    in_file.close()  
    return bytes_read
```

#### 2. *write\_file\_bytes*

Recibe el nombre del archivo a escribir y también recibe los bytes a ser escritos. Después abre el archivo y escribe los bytes en el archivo.

```
def write_file_bytes(filename, bytes_to_write):  
    out_file = open(filename, "wb")  
    out_file.write(bytes_to_write)  
    out_file.close()
```

### 3. *encrypt*

Recibe los bytes a encriptar y también recibe la contraseña con la cual encriptar.

```
def encrypt(bytes_to_encrypt, key):  
    key_256 = hashlib.sha256(key.encode()).digest()  
    iv = Random.new().read(AES.block_size)  
    aes = AES.new(key_256, AES.MODE_OFB, iv)  
    return iv + aes.encrypt(bytes_to_encrypt)
```

Primero se genera a partir de la contraseña una key de longitud 256bits (con el método de hash SHA256). La key debe ser de 256bits, porque AES256 depende de que la key sea de esa longitud.

```
key_256 = hashlib.sha256(key.encode()).digest()
```

Segundo se genera un IV (vector de inicialización) que son bytes generados aleatoriamente ya que es un parámetro necesario para la encriptación con AES junto a la key. La longitud del IV es del tamaño de los bloques utilizados por AES (128bits). Los bloques son como se dividen los bytes para ser encriptados.

```
iv = Random.new().read(AES.block_size)
```

Tercero se inicializa al objeto que realiza la encriptación pasándole los parámetros necesarios: key, IV, y el modo. El modo afecta a como se va realizando encadenando la encriptación de los distintos bloques.

```
aes = AES.new(key_256, AES.MODE_OFB, iv)
```

Por último, se encripta los bytes y se retorna el resultado agregándole al principio los bytes del IV, ya que este es un valor público que tiene que acompañar a los datos encriptados, para poder realizar la descriptación.

```
return iv + aes.encrypt(bytes_to_encrypt)
```

#### 4. *decrypt*

Recibe los bytes a descriptar y también recibe la contraseña con la cual descriptar.

```
def decrypt(bytes_to_decrypt, key):  
    key_256 = hashlib.sha256(key.encode()).digest()  
    iv = bytes_to_decrypt[:AES.block_size]  
    aes = AES.new(key_256, AES.MODE_OFB, iv)  
    return aes.decrypt(bytes_to_decrypt[AES.block_size:])
```

Primero se genera a partir de la contraseña una key de longitud 256bits (con el método de hash SHA256). Esta va a ser igual a la key utilizada en la encriptación ya que el hash de una misma contraseña da el mismo resultado.

```
key_256 = hashlib.sha256(key.encode()).digest()
```

Segundo se obtiene el IV agarrando los primeros 128bits de los bytes a descriptar ya que los habíamos guardado al principio de los bytes encriptados y con esa longitud.

```
iv = bytes_to_decrypt[:AES.block_size]
```

Tercero se inicializa al objeto que realiza la encriptación pasándole los parámetros necesarios: key, IV, y el modo. Todos los parámetros tienen que ser idénticos a los utilizados para la encriptación.

```
aes = AES.new(key_256, AES.MODE_OFB, iv)
```

Por último, se descripta los bytes (agarrando el resto de los bytes después del IV) y se retorna el resultado.

```
return aes.decrypt(bytes_to_decrypt[AES.block_size:])
```

## ANEXO

### Instrucciones para correr el programa:

- Tener instalado Python 3
- Instalar el paquete con la librería utilizada para la encriptación con AES con el comando "pip install pycryptodome"
- Correr el programa con el comando "python main.py <encrypt/decrypt> <nombre\_archivo> <contraseña>"

### EJEMPLO DE ENCRIPCIÓN:

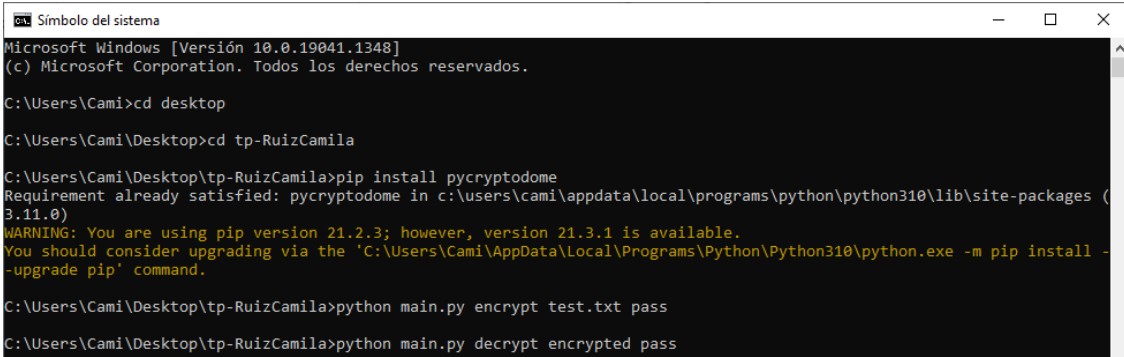
"python main.py encrypt test.txt pass"

Encripta el archivo "test.txt" con la contraseña "pass" y genera el archivo "encrypted" con el resultado.

### EJEMPLO DE DESENCRIPTACIÓN:

"python main.py decrypt encrypted pass"

Desencripta el archivo "encrypted" con la contraseña "pass" y genera el archivo "decrypted" con el resultado.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Cami>cd desktop

C:\Users\Cami\Desktop>cd tp-RuizCamila

C:\Users\Cami\Desktop\tp-RuizCamila>pip install pycryptodome
Requirement already satisfied: pycryptodome in c:\users\cami\appdata\local\programs\python\python310\lib\site-packages (3.11.0)
WARNING: You are using pip version 21.2.3; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\Cami\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.

C:\Users\Cami\Desktop\tp-RuizCamila>python main.py encrypt test.txt pass

C:\Users\Cami\Desktop\tp-RuizCamila>python main.py decrypt encrypted pass
```