

Tarea 1:

1. En esta tarea vamos a practicar cómo se cifra un texto con el algoritmo AES, el estándar de cifrado simétrico actual. Para ello vamos a utilizar Python 3. Como sugerencia puedes utilizar el IDE PyCharm con Python 3.x. Puedes obtener la versión community (gratis y open source) en este enlace <https://www.jetbrains.com/pycharm/download/#section=windows>.

Vamos a utilizar la librería criptográfica llamada "cryptography".

Esta librería se instala con el comando `$ pip install cryptography`.

En este enlace se pueden encontrar más detalles sobre su instalación:

<https://cryptography.io/en/latest/installation/>

a) El ejercicio consiste en cifrar el mensaje "a secret message" con la clave '12345678901234567890123456789012'.

Utiliza el modo de cifrado CBC. Descifra el criptograma y comprueba que obtienes el texto original. Explica cómo has desarrollado el ejercicio y aporta capturas de pantalla que apoyen tu explicación. Además, aporta el fichero .py que has generado.

Aquí podemos ver el código:

```
# Apartado A: Cifrado modo CBC
backend = default_backend()
key = b'12345678901234567890123456789012'
iv = os.urandom(16)

cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=backend)
encryptor = cipher.encryptor()

ct = encryptor.update(b"a secret message") + encryptor.finalize()
decryptor = cipher.decryptor()

result = decryptor.update(ct) + decryptor.finalize()

print("Mensaje encriptado con modo CBC {}".format(ct.hex()))
print("Mensaje original {}".format(result.decode("utf-8")))
```

Donde le pasamos la key indicada en bytes, damos un valor random al vector de inicialización.

Posteriormente llamamos al método Cipher y le pasamos la clave, el modo y el vector de inicialización.

Usando el método encryptor encriptamos el mensaje, pasado en bytes, con los parámetros metidos en el objeto cipher.

Llamando al método decryptor obtenemos el mensaje.

Por último, se escribe en pantalla el mensaje encriptado y el mensaje original:

```
Mensaje encriptado con modo CBC 2234db03bf0fd1212656d6d63bac4c07
Mensaje original a secret message
```

b) En este apartado vamos a ver la diferencia entre los diferentes modos de cifrado. Para ello se va a repetir el ejercicio anterior, utilizando los modos de cifrado OFB, CFB, ECB. Compara el criptograma obtenido en cada uno de los casos.

Si ejecutas el programa varias veces, ¿el resultado de los textos cifrados es siempre el mismo o varía? ¿A qué se debe este fenómeno?

El código usado para usar los otros módulos es el mismo o muy parecido.

```
# Apartado B.1: Cifrado modo OFB

key = b'12345678901234567890123456789012'
iv = os.urandom(16)

cipher = Cipher(algorithms.AES(key), modes.OFB(iv), backend=backend)
encryptor = cipher.encryptor()

ct = encryptor.update(b"a secret message") + encryptor.finalize()
decryptor = cipher.decryptor()

result = decryptor.update(ct) + decryptor.finalize()
```

```
# Apartado B.2: Cifrado modo CFB

key = b'12345678901234567890123456789012'
iv = os.urandom(16)

cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=backend)
encryptor = cipher.encryptor()

ct = encryptor.update(b"a secret message") + encryptor.finalize()
decryptor = cipher.decryptor()

result = decryptor.update(ct) + decryptor.finalize()

print("Mensaje encriptado con modo CFB {}".format(ct.hex()))
print("Mensaje original {}".format(result.decode("utf-8")))
```

```
# Apartado B.3: Cifrado modo ECB

key = b'12345678901234567890123456789012'

cipher = Cipher(algorithms.AES(key), modes.ECB(), backend=backend)
encryptor = cipher.encryptor()

ct = encryptor.update(b"a secret message") + encryptor.finalize()
decryptor = cipher.decryptor()

result = decryptor.update(ct) + decryptor.finalize()

print("Mensaje encriptado con modo ECB {}".format(ct.hex()))
print("Mensaje original {}".format(result.decode("utf-8")))
```

En el modo ECB no es necesario el vector de inicialización.

Los resultados en la primera ejecución son los siguientes:

```
Mensaje encriptado con modo CBC 2234db03bf0fd1212656d6d63bac4c07
Mensaje original a secret message
Mensaje encriptado con modo OFB 343b22bb3e101a3ef6036a5c14d91f81
Mensaje original a secret message
Mensaje encriptado con modo CFB 60709c88ed26779f2a1bdf0dcb8c56c4
Mensaje original a secret message
Mensaje encriptado con modo ECB b66218d747307ef7d64550f56fa5dc16
Mensaje original a secret message
```

En la segunda son estos:

```
Mensaje encriptado con modo CBC 7f2684b949332d094683f390843d3df8
Mensaje original a secret message
Mensaje encriptado con modo OFB f12b5af9c6d021903665915b9dfce5a4
Mensaje original a secret message
Mensaje encriptado con modo CFB fcb4eeb96b8daa7552b2e415cde92896
Mensaje original a secret message
Mensaje encriptado con modo ECB b66218d747307ef7d64550f56fa5dc16
Mensaje original a secret message
```

Como se puede observar en todos los modos el mensaje encriptado tiene la misma longitud, pero es diferente en todos ellos.

También el mensaje encriptado cambia en todos los modos, excepto en el modo ECB. Esto es debido a que estamos pasando en el resto de modos un vector de inicialización random y es lo que hace que el resultado cambie.

En el modo ECB no usamos este vector y por ello siempre obtenemos el mismo resultado, ante el mismo mensaje de entrada.

2. En este ejercicio vamos a programar una función para calcular el hash de un fichero. Esto se puede usar para comprobar la integridad de un fichero descargado.

Haciendo uso de la función hash MD5 de la librería “cryptography”, programa una función que calcule el hash MD5 de los ficheros “WinMD5.exe” y “WinMD5_2.exe”. Tienes los ejecutables en el fichero adjunto.

¿Cuáles son sus hashes? Teniendo en cuenta que el hash del fichero original es 944a1e869969dd8a4b64ca5e6ebc209a, ¿Cuál de los ficheros es correcto?

Si tienes problemas, que te impidan avanzar, con Python coméntamelo y realizamos la tarea sin la necesidad de programar.

Aquí el código empleado para comprobar que el fichero WinMD5 es el original:

```
digest = hashes.Hash(hashes.MD5(), default_backend())
with open('WinMD5.exe', 'rb') as f:
    contents = f.read()

digest.update(contents)
result_1 = digest.finalize().hex()
print('El hash del fichero WinMD5.exe es: {}'.format(result_1))

digest_2 = hashes.Hash(hashes.MD5(), default_backend())
with open('WinMD5_2.exe', 'rb') as f:
    contents = f.read()

digest_2.update(contents)
result_2 = digest_2.finalize().hex()
print('El hash del fichero WinMD5_2.exe es: {}'.format(result_2))

# WinMD5.exe es el fichero original.
```

En ambos casos se sigue el mismo proceso. Se usa el modulo Hash de la librería y se le pasa el contenido del fichero. A este se le aplica la función hash para obtener el hash de dicho contenido.

Estos son los resultados obtenidos:

```
El hash del fichero WinMD5.exe es: 944a1e869969dd8a4b64ca5e6ebc209a
El hash del fichero WinMD5_2.exe es: d1284c8060db2d9e8960696b6b8ccfc4
```

El fichero WinMD5.exe es el fichero original.

3. En este apartado vamos a trabajar sobre criptomonedas.

Consultando <https://coinmarketcap.com/> analiza la criptomoneda Monero.

En cada apartado recuerda explicar cómo has desarrollado el ejercicio y aporta capturas de pantalla que apoyen tu explicación

a) Explica brevemente algunas características sobre esta criptomoneda.

b) ¿Qué valor tiene en este momento? Indica la fecha y hora de la consulta.

c) ¿Qué volumen tuvo el 1 de Abril de 2020?

MONERO:

Es una divisa digital (P2P) descentralizada, segura, privada y no rastreable. Lanzada en 2014.

Su símbolo es XMR y como curiosidad Monero en Esperanto significa moneda.

Es minada mediante PoW. Aquí una guía de cómo hacerlo con CPU.

Es una moneda fungible, es decir, todas las monedas son iguales y valen lo mismo.

Tienes tres tecnologías muy importantes para permitir esa privacidad:

“Stealth Addresses” Se genera automáticamente para cada transacción una clave pública de manera aleatoria.

<https://www.getmonero.org/resources/moneropedia/stealthaddress.html>

“Ring Signatures” Tipo de firma de digital con un grupo de posibles firmantes, donde es imposible determinar quien del grupo ha firmado la transacción.

<https://www.getmonero.org/resources/moneropedia/ringsignatures.html>

“RingCT” Ring Confidential Transactions. Es como las transacciones son ocultadas en Monero.

Las claves en Monero:

Cuando se crea una cuenta de Monero se crean tres claves (Una más de lo habitual): la clave privada, la clave privada para las transacciones y la clave pública.

La clave privada se utiliza para mostrar las transacciones entrantes en tu cuenta.

La clave privada para transacciones se usa para enviar transacciones.

La clave pública se utiliza para recibir pagos.

El proyecto es mantenido por un grupo de desarrolladores y por la comunidad.

Precio en 27/03/2021 9:21hrs:

223,35\$

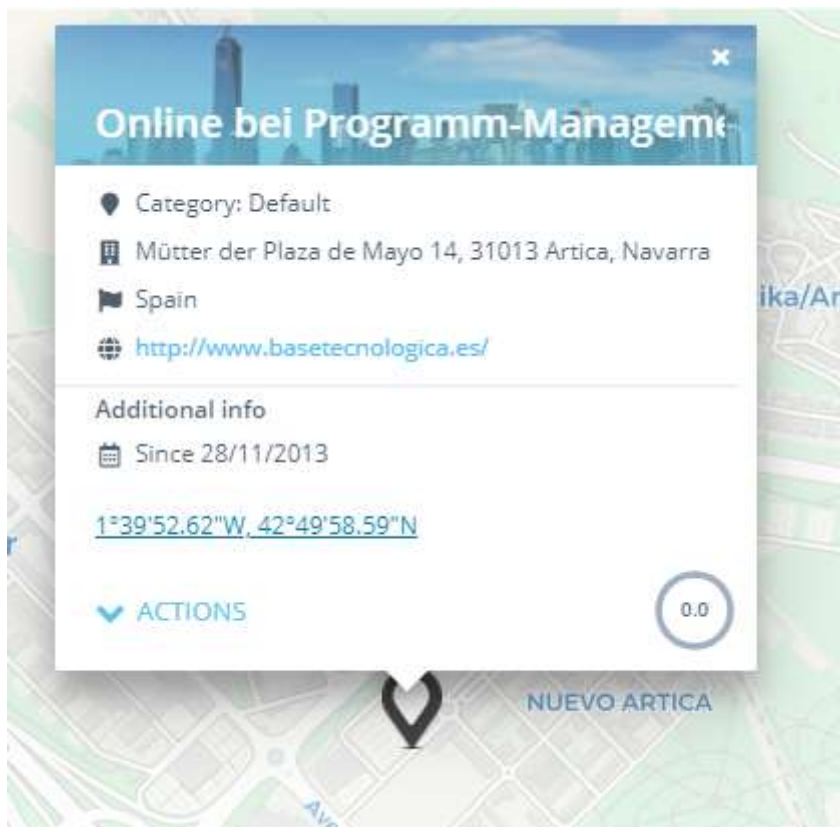


Volumen en 01/04/2020:

118.501.640\$

d) Localiza al menos dos comercios (preferiblemente en tu ciudad) que admitan pago en bitcoins. ¿Cuáles son y cómo los has encontrado? Si no hay comercios en tu ciudad, puedes indicar los de una ciudad cercana.

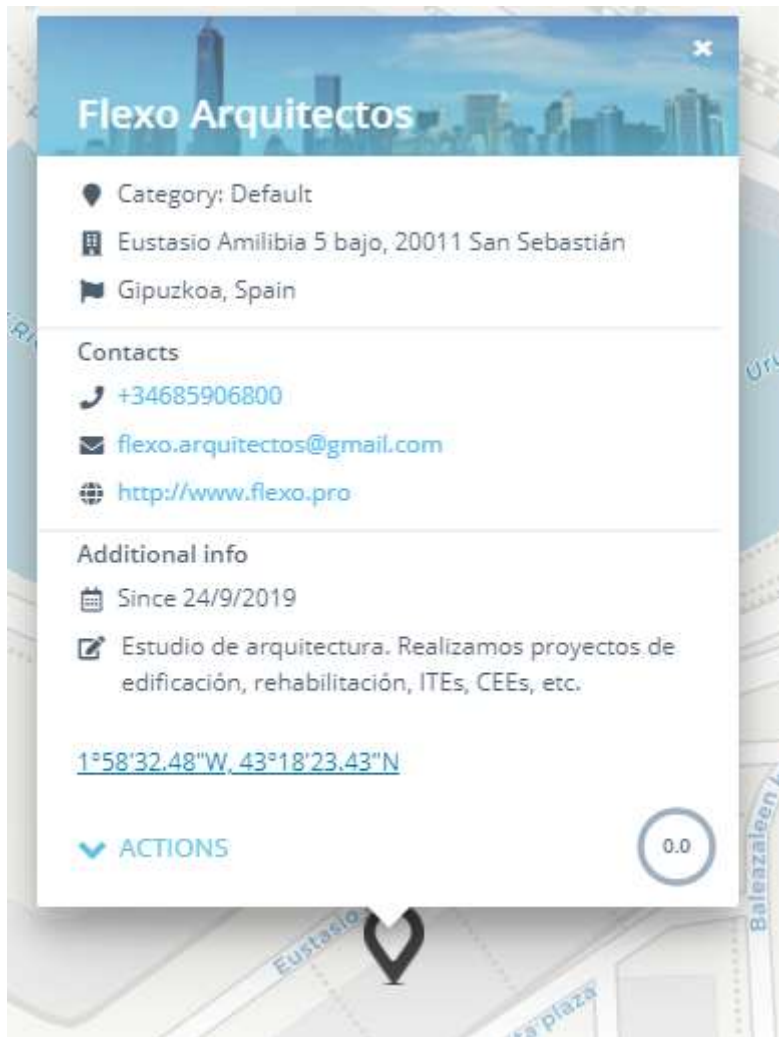
He usado <https://coinmap.org/> para localizarlos. En Pamplona solamente me sale un comercio:



Pero no se si funciona ya que la web no funciona...

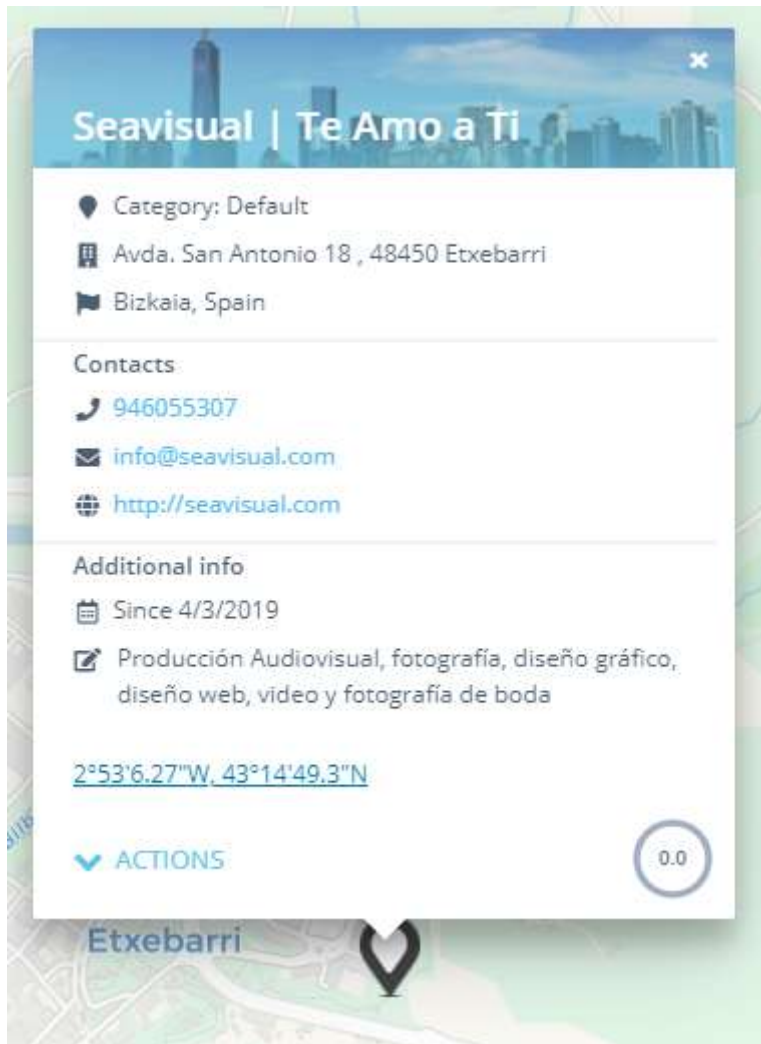
Así que me he ido a San Sebastian donde he encontrado este:

<https://www.flexo.pro/>



Y en Bilbao he encontrado este otro:

<http://www.seavisual.com/?ref=coinmap.org>



e) Créate una cuenta en el monedero de blockchain.info. Da alguna de tus direcciones públicas (¡ten cuidado de no compartir tu clave privada!). Explica el proceso que has seguido para conseguirla. No olvides aportar una captura de pantalla indicando cómo has obtenido la dirección pública.

1. En la sección de wallet en https://www.blockchain.com/wallet?utm_campaign=dcomnav_wallet he pulsado el botón de Create Your Wallet:

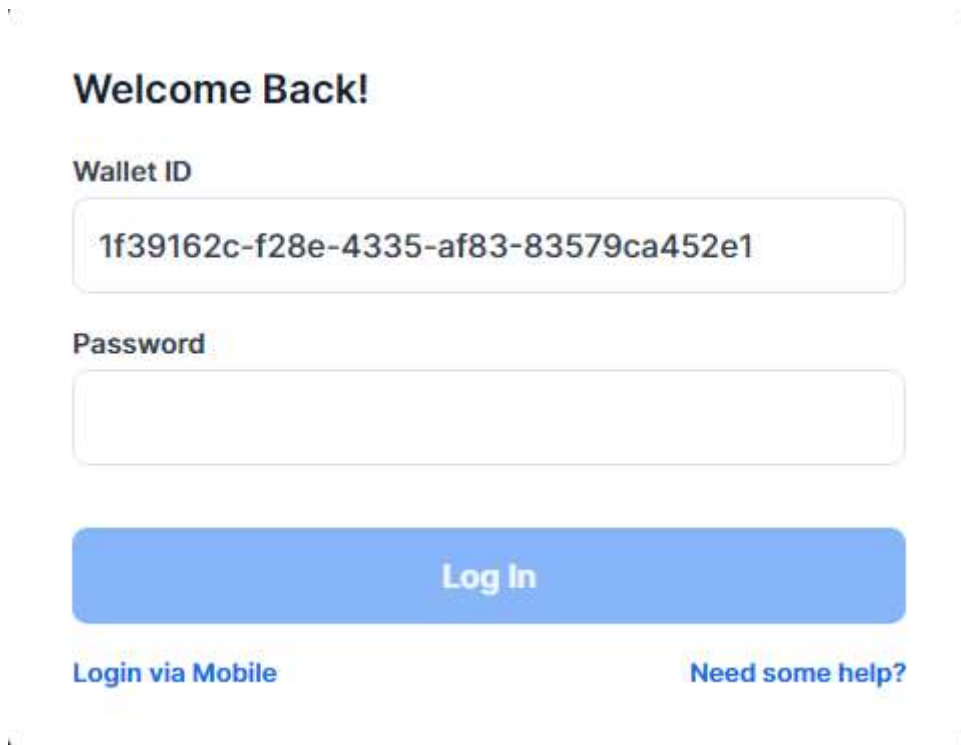
Be Your Own Bank[®]

The safest and most popular wallet for investing and storing cryptocurrencies

Create Your Wallet

2. Te pide email + contraseña. Posteriormente hay que verificar el email.

3. El wallet es creado:



Welcome Back!

Wallet ID

1f39162c-f28e-4335-af83-83579ca452e1

Password

Log In

[Login via Mobile](#) [Need some help?](#)

4. Me pide una confirmación de login via correo electrónico:

Authorize Log In Attempt

A login attempt to your Blockchain wallet was made from an unknown browser. Please make sure the following details are correct before authorizing the login:

Time: 2021-03-27 08:26:21 GMT
IP Address: 185.153.150.79
Browser: Chrome 8
Operating System: Windows 10
Location: ES

If the above details are correct, click the following link to approve the request.

5. Una vez pulsado aceptar, de nuevo confirmar desde el navegador el log in:



Login attempt from another browser

Someone, hopefully you, is attempting to login to your wallet from a different browser.

Your Device vs. Requesting Device

- ✓ Browser: ✓ Chrome 8
- ✓ IP Address: ✓ 185.153.150.79
- ✓ Country of Origin: ✓ Spain

Accept

Or

Reject

6. Login Approved



Login Approved!

Please return to your previous
tab to view your wallet.

7. El wallet:

Blockchain.com
Send
Request
Swap
Buy/Sell Crypto
Earn Interest
Borrow

Total Balance
\$0,00

Dashboard

Bitcoin
Ether
Bitcoin Cash
Stellar
Wrapped-DGLD NEW

Increase your limits
Continue your verification to become Gold level and increase your limits and payment m

Total Balance
\$0,00

TotalWalletHardware

Bitcoin \$0,00
0 BTC

Ether \$0,00
0 ETH

8. Podemos ver nuestras claves públicas pulsando en request en cualquiera de los activos digitales disponibles.

Esta es mi clave pública para BTC: 1P3yuiF7jFpDi4WV6w2gZ7ppEjaXvoxZM Mediante la cual puedo recibir transacciones del mismo.

Request Bitcoin


×

Currency:

Bitcoin

Receive To:

My Bitcoin Wallet (0 BTC)



Address:

1P3yuiF7jFpDi4WV6w2gZ7ppEjaXvoxZM

📄

Done

Create Shareable Request Link

f) ¿Cuál es el hash correspondiente al bloque número 650217 de Bitcoin Cash? Menciona diferencias entre Bitcoin y Bitcoin Cash. ¿Cuál es el tamaño del bloque y cuántas transacciones tiene?

Explorador de bloques: <https://explorer.bitcoin.com/bch>

Hash bloque número 650217 de Bitcoin Cash:

[illegible]

Block #650217

[illegible]

BCH es un fork de BTC realizado el 1 de agosto de 2017. Surge por discrepancias en la comunidad en como resolver el problema de la escalabilidad de BTC.

La principal diferencia entre BCH y BTC es que el primero ha aumentado el tamaño de bloque a los 32MB (empezó en 8MB) Siendo el tamaño de bloque en BTC de 1MB.

BTC considera a SegWit como la solución a los problemas de escalado junto con aplicaciones como Lightning Network y Atomic Swaps.