```python
# La criptografía de curvas elípticas (o ECC) es una variante de la criptografía asimétrica basada
# en las matemáticas de las curvas elípticas. Al igual que RSA, esta clase de criptografía permite
# tanto realizar operaciones de cifrado (aun no implementadas en pycryptodome) como de firma.
# Se pide implementar en el fichero ecc.py las funciones indicadas en el apéndice B
# <<Crear una clave pública y una clave privada RSA de 2048 bits para Bob. Guardar cada clave en un
# fichero>> utilizando criptografía de curvas elípticas. Para ello, se deberá consultar la documentación
# de la librería pycryptodome:
# Ver https://pycryptodome.readthedocs.io/en/latest/src/public_key/ecc.html
# Ver https://pycryptodome.readthedocs.io/en/latest/src/signature/dsa.html

from Crypto.PublicKey import ECC
from Crypto.Hash import SHA256
from Crypto.Signature import pss


def crear_ECCKey():
    key = ECC.generate(curve='P-256')
    return key

def guardar_ECCKey_Privada(fichero, key, password):
    key_cifrada = key.export_key(format = 'PEM',passphrase=password, use_pkcs8 = True, protection="scryptAndAES128-CBC")
    file_out = open(fichero, "wt")
    file_out.write(key_cifrada)
    file_out.close()

def cargar_ECCKey_Privada(fichero, password):
    key_cifrada = open(fichero, "rt").read()
    key = ECC.import_key(key_cifrada, passphrase=password)
    return key

def guardar_ECCKey_Publica(fichero, key):

    key_pub = key.public_key().export_key(format = 'PEM')
    file_out = open(fichero, "wt")
    file_out.write(key_pub)
    file_out.close()

def cargar_ECCKey_Publica(fichero):
    keyFile = open(fichero, "rt").read()
    key_pub = ECC.import_key(keyFile)
    return key_pub

# def cifrarECC_OAEP(cadena, key):
    # El cifrado con ECC (ECIES) aun no está implementado
    # Por lo tanto, no se puede implementar este método aun en la versión 3.9.7
    # return cifrado

# def descifrarECC_OAEP(cifrado, key):
    # El cifrado con ECC (ECIES) aun no está implementado
    # Por lo tanto, no se puede implementar este método aun en la versión 3.9.7
    # return cadena

def firmarECC_PSS(texto, key_private):
# La firma se realiza sobre el hash del texto (h)
    h = SHA256.new(texto.encode("utf-8"))
    print(h.hexdigest())
```

```
    signature = pss.new(key_private).sign(h)
    return signature

def comprobarECC_PSS(texto, firma, key_public):
# Comprobamos que la firma coincide con el hash (h)
    h = SHA256.new(texto.encode("utf-8"))
    print(h.hexdigest())
    verifier = pss.new(key_public)
    try:
        verifier.verify(h, firma)
        return True
    except (ValueError, TypeError):
        return False

#Crea clave pública y privada de BOB

ECC_B = crear_ECCKey()

password = "1234"

#guarda las claves en ficheros distintos

guardar_ECCKey_Publica("B_pub.pem",ECC_B)

guardar_ECCKey_Privada("B_priv.pem",ECC_B,password)

```