```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import pss
from Crypto.Hash import SHA256

def crear_RSAKey():
    key = RSA.generate(2048)
    return key

def guardar_RSAKey_Privada(fichero, key, password):
    key_cifrada = key.export_key(passphrase=password, pkcs=8, protection="scryptAndAES128-CBC")
    file_out = open(fichero, "wb")
    file_out.write(key_cifrada)
    file_out.close()

def cargar_RSAKey_Privada(fichero, password):
    key_cifrada = open(fichero, "rb").read()
    key = RSA.import_key(key_cifrada, passphrase=password)
    return key

def guardar_RSAKey_Publica(fichero, key):
    key_pub = key.publickey().export_key()
    file_out = open(fichero, "wb")
    file_out.write(key_pub)
    file_out.close()

def cargar_RSAKey_Publica(fichero):
    keyFile = open(fichero, "rb").read()
    key_pub = RSA.import_key(keyFile)
    return key_pub

def cifrarRSA_OAEP(cadena, key):
    datos = cadena.encode("utf-8")
    engineRSACifrado = PKCS1_OAEP.new(key)
    cifrado = engineRSACifrado.encrypt(datos)
    return cifrado

def descifrarRSA_OAEP(cifrado, key):
    engineRSADescifrado = PKCS1_OAEP.new(key)
    datos = engineRSADescifrado.decrypt(cifrado)
    cadena = datos.decode("utf-8")
    return cadena

def firmarRSA_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 comprobarRSA_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
```

```
61
62  #Crea par de claves para Alice y Bob
63
64
65  RSA_A = crear_RSAKey()
66  RSA_B = crear_RSAKey()
67
68  password = "1234"
69
70  #guarda las claves en ficheros distintos
71  guardar_RSAKey_Publica("A_pub.pkcs",RSA_A)
72  guardar_RSAKey_Publica("B_pub.pkcs",RSA_B)
73  guardar_RSAKey_Privada("A_priv.pkcs",RSA_A,password)
74  guardar_RSAKey_Privada("B_priv.pkcs",RSA_B,password)
75
76
77
78
```