PRACTICAL 4

Name:	Harsh Shah	**Semester:**	VII	**Division:**	6
Roll No.:	21BCP359	**Date:**	22-08-24	**Batch:**	G11
Aim:	To Implement Digital Signature in any programming language				

Digital Signature

A digital signature is a mathematical scheme that is used to verify the integrity and authenticity of digital messages and documents. It may be considered as a digital version of the handwritten signature or stamped seal. The digital signatures use asymmetric cryptography i.e., also known as public key cryptography.

Program

```
import hashlib
import os
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import rsa, padding
import base64

PRIVATE_KEY_PATH = "example-rsa.pem"
PUBLIC_KEY_PATH = "example-rsa.pub"
PRIVATE_KEY_PASS = b"my$ecretp@$$word"

def generate_private_key():
    private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)

    pem_private_key = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.BestAvailableEncryption(PRIVATE_KEY_PASS),
    )

    with open(PRIVATE_KEY_PATH, "wb") as private_key_file:
        private_key_file.write(pem_private_key)

    return private_key

def generate_public_key(private_key):
    pem_public_key = private_key.public_key().public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo,
```

```
    )
    with open(PUBLIC_KEY_PATH, "wb") as public_key_file:
        public_key_file.write(pem_public_key)
    return pem_public_key

def get_key_pairs():
    if os.path.exists(PRIVATE_KEY_PATH) and os.path.exists(PUBLIC_KEY_PATH):
        with open(PRIVATE_KEY_PATH, "rb") as private_key_file:
            keydata = private_key_file.read()
            private_key = serialization.load_pem_private_key(
                keydata, password=PRIVATE_KEY_PASS
            )
        with open(PUBLIC_KEY_PATH, "rb") as public_key_file:
            keydata = public_key_file.read()
            public_key = serialization.load_pem_public_key(keydata)

        return private_key, public_key
    else:
        private_key = generate_private_key()
        generate_public_key(private_key)
        return private_key

def sign_message(private_key, message):
    signature = private_key.sign(message, padding.PKCS1v15(), hashes.SHA512())
    return base64.b64encode(signature).decode("utf-8")

def verify_signature(public_key, message, signature):
    decoded_signature = base64.b64decode(signature)
    try:
        public_key.verify(
            decoded_signature, message, padding.PKCS1v15(), hashes.SHA512()
        )
        return True
    except:
        return False

if __name__ == "__main__":
    private_key, public_key = get_key_pairs()
    print("What do you want to do?\n 1-Sign\n 2-Verify")
    choice = int(input("Enter your choice [1/2]: "))

    if choice == 1:
        message_input = input("Enter the message to sign: ").encode("utf-8")
        signature = sign_message(private_key, message_input)
```

```
        print("Signature:", signature)

    elif choice == 2:
        message_input = input("Enter the message to verify: ").encode("utf-8")
        signature_input = input("Enter the signature: ")
        is_valid = verify_signature(public_key, message_input, signature_input)
        print(f"Signature is {'valid' if is_valid else 'invalid'}")
    else:
        print("Enter correct choice!")
```

Output

```
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 7\Blockchain\Blockchain Lab\practical5> python -u "c:\Users\harsh\OneDrive - p
dpu.ac.in\HARSH\_PDEU\SEM 7\Blockchain\Blockchain Lab\practical4\digital_signature.py"
What do you want to do?
 1-Sign
 2-Verify
Enter your choice [1/2]: 1
Enter the message to sign: This is a seceret message which is being signed
Signature: wsMUBmwgWOQdXzUdBsTYvi/awwLRwK9XuEyK8cOyDeaHJ1Q2QCP5sVOexdLP/nNnqGEMDWJzHSC69G/VcUimZ73nIGiMUrHP/cSyBrxo5aqLFxKNsGKyfChAH7E
BjDdvwZkqT3I075/vQHcQV+1Fym966qxjkq67BFGFShnJYzGSiq2EHZrP14ydYlu4xUio2DISlaS0/ZIPdcA3LNaFalDK8MwRotxS4HEocKdegQLtxqrlNYhx6SSuY/Z2bRWF9
WxE05NmrACGvTPdtdvhOB4GD0wlMDqkiF4RBaZsFhMS5jcB4jgVziYVYOLyIVaEAySN77dG6iowS2Iu3yOjBA==
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 7\Blockchain\Blockchain Lab\practical5> python -u "c:\Users\harsh\OneDrive - p
dpu.ac.in\HARSH\_PDEU\SEM 7\Blockchain\Blockchain Lab\practical4\digital_signature.py"
What do you want to do?
 1-Sign
 2-Verify
Enter your choice [1/2]: 2
Enter the message to verify: This is a not seceret message which is being signed
Enter the signature: wsMUBmwgWOQdXzUdBsTYvi/awwLRwK9XuEyK8cOyDeaHJ1Q2QCP5sVOexdLP/nNnqGEMDWJzHSC69G/VcUimZ73nIGiMUrHP/cSyBrxo5aqLFxKNs
GKyfChAH7EBjDdvwZkqT3I075/vQHcQV+1Fym966qxjkq67BFGFShnJYzGSiq2EHZrP14ydYlu4xUio2DISlaS0/ZIPdcA3LNaFalDK8MwRotxS4HEocKdegQLtxqrlNYhx6SS
uY/Z2bRWF9WxE05NmrACGvTPdtdvhOB4GD0wlMDqkiF4RBaZsFhMS5jcB4jgVziYVYOLyIVaEAySN77dG6iowS2Iu3yOjBA==
Signature is invalid
PS C:\Users\harsh\OneDrive - pdpu.ac.in\HARSH\_PDEU\SEM 7\Blockchain\Blockchain Lab\practical5>
```