Walchand College of Engineering, Sangli ***(Government Aided Autonomous Institute)***	
AY 2025-26	
Course Information	
Programme	B.Tech. (Computer Science and Engineering)
Class, Semester	Final Year B. Tech., Sem VII
Course Code	6CS451
Course Name	Cryptography and Network Security Lab
PRN	22510016

Experiment No. 10

Title Implement the SIGNATURE SCHEME – Digital Signature Standard

---

Objectives:

To implement the ***Digital Signature Standard (DSS)*** using the ***Digital Signature Algorithm (DSA)*** to create and verify digital signatures, thereby ensuring the authenticity and integrity of electronic messages or documents.

Problem Statement:

In digital communication, ensuring the authenticity of a sender and the integrity of the message is critical. A ***digital signature*** serves as a cryptographic technique to validate both.

Design and implement a system using the ***Digital Signature Standard (DSS)*** that performs the following tasks:

1. ***Key Generation***:

o Generate a private key (used for signing) and a public key (used for verifying).
o Use parameters such as large prime numbers and hash functions (e.g., SHA-1 or SHA-256) as required by DSA.
2. ***Message Signing***:
o Accept a message (text input).
o Use the private key to generate a ***digital signature*** for the message using the DSA algorithm.
3. ***Signature Verification***:
o Use the corresponding public key to verify the ***digital signature*** against the original message.
o Confirm whether the signature is valid or has been tampered with.
4. ***Demonstrate Security***:
o Show that any alteration in the message or signature results in a failed verification.
o Explain how this ensures ***message integrity***, ***authentication***, and ***non-repudiation***.

The system should simulate a real-world use case where one party signs a message and another party verifies it, highlighting how digital signatures help in secure digital communications, such as email signing, legal document authentication, or secure software distribution.

Equipment/Tools:

1. Hardware system: Computer system
2. Programming language: Python, libraries

Theory:

DSA (Digital Signature Algorithm) is a public-key signature algorithm standardized in the Digital Signature Standard (DSS).
A signer has a private key (used to sign) and a public key (used by verifiers).
Signing uses a hash (SHA-1, SHA-256, etc.) of the message and DSA math to produce an (r, s) signature.
Verification checks the signature against the message hash and public key. Any change to message or signature causes verification to fail.
DSA provides integrity**,** authentication (only private key holder could produce signature), and non-repudiation (signer cannot later plausibly deny the signature, assuming private key secrecy).

Procedure:

1. Generate DSA parameters and key pair (private/public).
2. Hash the message (signing API will handle hashing when configured).
3. Sign the message with the private key to produce signature bytes.
4. Verify signature with the public key and original message.
5. Demonstrate tampering: alter message or signature → verification fails.

Steps:

```python
from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.exceptions import InvalidSignature

def generate_keys(key_size=2048):
    private_key = dsa.generate_private_key(key_size=key_size)
    public_key = private_key.public_key()
    return private_key, public_key

def sign_message(private_key, message: bytes):
    return private_key.sign(message, hashes.SHA256())

def verify_signature(public_key, message: bytes, signature: bytes):
    try:
        public_key.verify(signature, message, hashes.SHA256())
        return True
    except InvalidSignature:
        return False

def demo():
    priv, pub = generate_keys()
    msg = input("Enter message to sign: ").encode()
    signature = sign_message(priv, msg)
    print("\n--- Test Cases ---")
    print("1. Valid Input Test:")
    print("Signature Verified:", verify_signature(pub, msg, signature))
    print()
    tampered_msg = input("Enter tampered message: ").encode()
    print("2. Invalid Input Test (Tampered Message):")
    print("Signature Verified:", verify_signature(pub, tampered_msg, signature))
    tampered_sig = bytearray(signature)
    tampered_sig[3] ^= 0xAA
    print()
    print("3. Invalid Input Test (Tampered Signature):")
    print("Signature Verified:", verify_signature(pub, msg, bytes(tampered_sig)))

if __name__ == "__main__":
    demo()
```

```
Enter message to sign: Hello World

--- Test Cases ---
1. Valid Input Test:
Signature Verified: True

Enter tampered message: Hello World ??
2. Invalid Input Test (Tampered Message):
Signature Verified: False

3. Invalid Input Test (Tampered Signature):
Signature Verified: False
```

Observations and Conclusion:

Test Case	*Input Type*	*Expected Result*	*Actual Result*
1	Original message	True (valid)	True
2	Tampered message	False (invalid)	False
3	Tampered signature	False (invalid)	False

The DSA-based Digital Signature System successfully ensures:

Integrity – message change is detected.
Authentication – only signer with private key can sign.
Non-repudiation – signer cannot deny the signature later.

Any alteration in message or signature results in failed verification, proving the security of the Digital Signature Standard (DSS).
