



School: Campus:

Academic Year: Subject Name: Subject Code:

Semester: Program: Branch: Specialization:

Date:

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiment : ECDSA Workshop – Digital Signatures Demo

Objective/Aim:

To study and demonstrate the working of the Elliptic Curve Digital Signature Algorithm (ECDSA) by generating keys, signing a message, and verifying the signature.

Apparatus/Software Used:

- Computer with internet access

Theory/Concept:

ECDSA (Elliptic Curve Digital Signature Algorithm) is a cryptographic algorithm used for digital signatures.

- It provides authentication, data integrity, and non-repudiation.

Process:

1. **Key Generation** – A private key is chosen randomly, and a public key is derived using elliptic curve multiplication.
2. **Signing** – A hash of the message is generated and signed using the private key to produce a digital signature.
3. **Verification** – The signature is verified using the sender's public key and the original message hash.

Advantages: Strong security with smaller key sizes compared to RSA, widely used in blockchain (e.g., Bitcoin, Ethereum).

Procedure:

1. Install the required cryptography library (pip install ecdsa).
2. Generate an elliptic curve key pair (private and public keys).
3. Take an input message (e.g., "Blockchain Lab Demo").
4. Hash the message using SHA-256.
5. Use the private key to sign the message hash → digital signature.
6. Verify the signature using the public key and message hash.
7. Observe that:
 - If the message or signature is altered, verification fails.
 - If unchanged, verification passes.

```
Key: c60fde415ecb2785aca77177137ede07d662dda0559f83d96ddb6f46000e50
Key: feffbd1d6e8b11d3816e78bdc5ff09ebe6e5ed0e811195c35ceabaad77b65cbab79363817086e2669f21551353f060dd5acf18392924d4afc6510536bf61905e

: Blockchain Lab Demo
Hash: e036e51357918c4ea0e79561181eca3d85633138dad9df6ca1e2c8ae5f64c30b

Signature: c1692251ab710ff5fd7c2d789680d860084076ddba565ce2611b030aace75d04f63628108fe8930f1ce7dfe2a3163529269e222b89490bcbf65143ff292fa38

ation Result: Valid ✓
```

```
# Tamper the message
fake_message = b"Blockchain Lab Tampered"
fake_hash = hashlib.sha256(fake_message).digest()

# Try to verify with fake message
try:
    public_key.verify(signature, fake_hash)
    print("Tampered Message Verification: Valid ✓")
except:
    print("Tampered Message Verification: Invalid ✗")
```

Tampered Message Verification: Invalid ✗

Observation Table:

| Step | Input/Process | Output/Result |
|------------------------|----------------------------------|---|
| Key Generation | Random private key | Public key derived from EC multiplication |
| Message Input | "Blockchain Lab Demo" | Message ready for hashing |
| Hashing | SHA-256(Message) | 64-character hash value |
| Signature Generation | Private Key + Hash | Digital Signature |
| Signature Verification | Public Key + Message + Signature | Valid (if original) / Invalid (if tampered) |

ASSESSMENT

| Rubrics | Full Mark | Marks Obtained | Remarks |
|--|-----------|----------------|---------|
| Concept | 10 | | |
| Planning and Execution/ Practical Simulation/ Programming | 10 | | |
| Result and Interpretation | 10 | | |
| Record of Applied and Action Learning | 10 | | |
| Viva | 10 | | |
| Total | 50 | | |

Signature of the Faculty:

Signature of the Student:
Name :
Regn. No.