Centurion University
Shaping Lives... Empowering Communities...

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 Experiement : 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
- Google colab

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.

```
Private Key: 6764c67e6a084fdf30beda2228b5f1a99c0084138f58e796d346f59a59532073
Public Key: a6f11173b8ee4046a134fc28b6ec6b9281d061d16f5f07567b24cf7951bf58ee5f461d6a4b269135d98e62b894295

Message: Blockchain Lab Demo
Message Hash: e036e51357918c4ea0e79561181eca3d85633138dad9df6ca1e2c8ae5f64c30b

Digital Signature: 35dbb500894b80084f373793e9fc3c813ff28f744e904410be8b3cddf7219f5668c5ea0245f916b0bde155

Verification 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 Student:

Name :
Regn. No.

Signature of the Faculty: