



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 :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

ECDSA Workshop – Digital Signatures Demo:

The tool demonstrates Elliptic Curve Digital Signature Algorithm (ECDSA) - the same cryptography used by Bitcoin, Ethereum, and most blockchain networks. It shows how digital signatures provide authentication, integrity, and non-repudiation.

Key Components:

1. Elliptic Curve Cryptography (secp256k1)
 - Uses the same curve as Bitcoin/Ethereum (secp256k1)
 - Private key: A random 256-bit number (64 hex characters)
 - Public key: Derived from private key using elliptic curve multiplication
 - One-way function: Easy to generate public key from private key, but impossible to reverse
2. SHA-256 Hashing
 - Converts any message into a fixed 256-bit fingerprint
 - Ensures message integrity - any change in message changes the hash completely
3. Digital Signature Process
 - Signing: Private key + Message hash = Signature
 - Verification: Public key + Message hash + Signature = Valid/Invalid

Step-by-Step Workflow

Phase 1: Key Generation

User clicks "Generate New Key Pair"

↓
`elliptic.ec('secp256k1').genKeyPair()`

↓
Private Key: 64-character hex (e.g., "2f4...c3a")

↓
Public Key: 130-character hex (derived from private key)

Phase 2: Message Signing

Message: "Hello, Blockchain World!"

↓
SHA-256 Hash: "a591a6...b64b4" (64-character hex)

↓
Private Key signs the hash using ECDSA algorithm

↓
Digital Signature: "3044...022100...0220..." (DER format)

Coding Phase: Pseudo Code / Flow Chart / Algorithm

Phase 3: Signature Verification

Original Message + Signature + Public Key



SHA-256 Hash of original message



elliptic.verify(messageHash, signature, publicKey)



Result: VALID or INVALID

* Softwares used

- VSCode
- Node.js
- React
- Elliptic.js, crypto.js
- Curve- secp256k1
- SHA256
- HTML, CSS, Javascript

*** Testing Phase: Compilation of Code (error detection)**

Compiled with problems:

ERROR in ./src/components/Signer.jsx 34:15-36

export 'generateRandomMessage' (imported as 'generateRandomMessage') was not found in '../utils/ecdsaHelpers' (possible exports: formatKey, generateKeyPair, isValidHex, signMessage, verifySignature)

Original Message:

* Implementation Phase: Final Output (no error)

ECDSA Digital Signature Tool

Generate keys, sign messages, and verify signatures using ECDSA with secp256k1

ECDSA Key Generator

Generate New Key Pair

⚠ Important: Never share your private key! It should be kept secret and secure.

Message Signer

Message to Sign:

Enter your message here...

Use Random Message

Sign Message

Signature Verifier

Original Message:

Enter the original message...

Digital Signature:

Enter the digital signature...

Public Key:

Enter the public key...

Verify Signature

Clear Form

Built with React & Elliptic.js | ECDSA (secp256k1) | SHA-256

ECDSA Digital Signature Tool

Generate keys, sign messages, and verify signatures using ECDSA with secp256k1

ECDSA Key Generator

Generate New Key Pair

⚠ Important: Never share your private key! It should be kept secret and secure.

Message Signer

Message to Sign:

Enter your message here...

Use Random Message

Sign Message

Page No.....

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Implementation Phase: Final Output (no error)

Signature Verifier

Original Message:

Enter the original message...

Digital Signature:

Enter the digital signature...

Public Key:

Enter the public key...

Verify Signature

Clear Form

Built with React & Elliptic.js | ECDSA secp256k1 | SHA-256

ECDSA Digital Signature Tool

Generate keys, sign messages, and verify signatures using ECDSA with secp256k1

ECDSA Key Generator

Generate New Key Pair

Private Key (Keep Secret!)

f05c31e1b07180529890de9faa4669c16974054c23532e8ba87d454985b07272

Copy Private Key

Public Key (Can Share)

0461a4f790f84c8a780de1912fcc5b2c519d04b15b81e074649eee5c48826db
24b8e36e790ef42a92e9f23275d601c71d6a0768079aefc0a11569672de81334
69

Copy Public Key

Important: Never share your private key! It should be kept secret and secure.

ECDSA Key Generator

Generate New Key Pair

Copied: f05c31e1b0718052...

Private Key (Keep Secret!)

f05c31e1b07180529890de9faa4669c16974054c23532e8ba87d454985b07272

Copy Private Key

ECDSA Key Generator

Generate New Key Pair

Copied: 0461a4f79b84c8a78d8e1912fcc5b2c519d04b15b81e074649eee5cb48826db

Private Key (Keep Secret!)
f05c31e1b07188520890de9faa4669c16974054c23532e8ba87d454985b07272
Copy Private Key

Public Key (Can Share)
0461a4f79b84c8a78d8e1912fcc5b2c519d04b15b81e074649eee5cb48826db
24b8e36e790efda2a92e9f23275d001c71d6a070b879aefc0a11569672de8133468
Copy Public Key

⚠ Important: Never share your private key! It should be kept secret and secure.

Message Signer

Message to Sign:
Digital Signature Verification

Use Random Message

Sign Message

Signature Created Successfully

Original Message:
Digital Signature Verification

Message Hash (SHA-256): 3f588689423ff312aa27a7b0188ee758aa54c4df8a0e1913261939214dce7aca

Digital Signature:
30440220240bca34533b533e97ad1c9634ede86f9a6ee3b72ef918507218c5b1315df4702284c220f1f39bc597b24c96413151888305776c247004c8920090574677ac32
Copy Signature

localhost:3000 says

Signature copied to clipboard!

OK

Signature Verifier

Original Message:
Hello, Blockchain World!

Digital Signature:
3044022039978e7196f5d2a9ba1bbf167b658a7b6ae23b9adbb9507681ea7f0257208a96022047f2b9b8ea66b91c598bf27ac4e9127e7ce85746774369166c3898e108f5cc8

Public Key:
0461a4f79b84c8a78d8e1912fcc5b2c519d04b15b81e074649eee5cb48826db24b8e36e790efd2a92e9f23275d601c71d6a070b879aefc0a11569672de8133468

Verify Signature Clear Form

✗ Signature is INVALID
The signature does not match the message or public key.
Message Hash: d8fe8a89b79ce52deecce79f4002fb453bac6cb6f1ae14fa08b4de45cab8395

* Implementation Phase: Final Output (no error)

Applied and Action Learning

Signature Verifier

Original Message:

Hello, Blockchain World!

Digital Signature:

3044022039978e7196f5d2a90a1bbf167b658a7b6ae23b9adb9507681ea7f0257208a96022047f2b9b08ea66b91c598bf27ac4ef9127e7ce85746774369166c3898e100f5cc8

Public Key:

0461a4f79bf84c8a78dde1912fcc5b2c519d04b15b81e074649ee5cb48826db24b8e36e790efd2a92e9f23275d601c71d6a076b879aefc0a11569672de8133469

Verify Signature

Clear Form

Signature is VALID

The signature matches the message and was created by the corresponding private key.
Message Hash: d8fe8a89b79ce52deec79f4002fb453bac6cb6f1ae14fa08b4de45cab8395

* Observations

- Authentication
 - Only the holder of the private key can create a valid signature
 - Verifier can confirm the signer's identity using the public key
- Integrity
 - Any change to the message invalidates the signature
 - SHA-256 ensures even a single character change produces completely different hash
- Non-Repudiation
 - Signer cannot deny having signed the message
 - Signature is mathematically tied to their private key

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:

Page No.....

**As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.*