

 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).

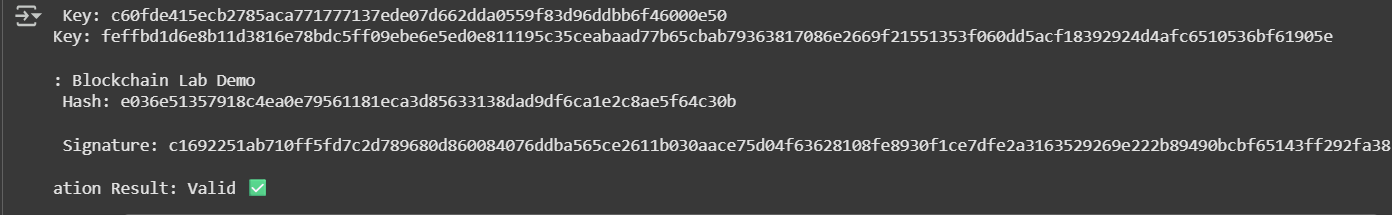


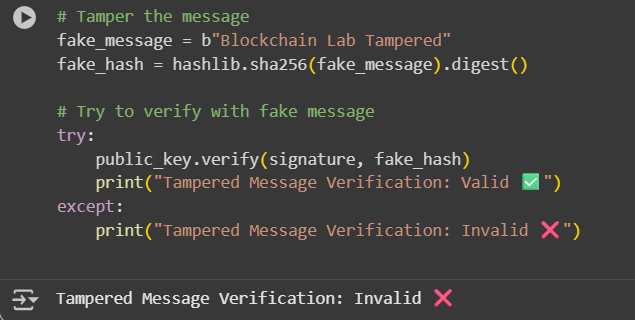
**Procedure:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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.





**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) |



