

ECDSA Workshop – Digital Signatures Demo  
  
**Objective/Aim:**



To understand the working of the **Elliptic Curve Digital Signature Algorithm (ECDSA)** by generating

key pairs, signing a message, and verifying the digital signature.

**Apparatus/Software Used:**

* Laptop / PC
* Python (with ecdsa or cryptography library) OR any ECDSA demo tool
* Word/Docs for documentation

**Theory/Concept:**

* Digital Signatures are cryptographic techniques that ensure:
* Authenticity: Message is really from the sender.
* Integrity: Message is not altered.
* Non-repudiation: Sender cannot deny signing.
* ECDSA (Elliptic Curve Digital Signature Algorithm):
* Based on elliptic curve cryptography (ECC), which is stronger and faster than RSA at smaller key sizes.
* Uses a private key to generate the signature.
* Uses a public key to verify the signature.
* Working Principle:

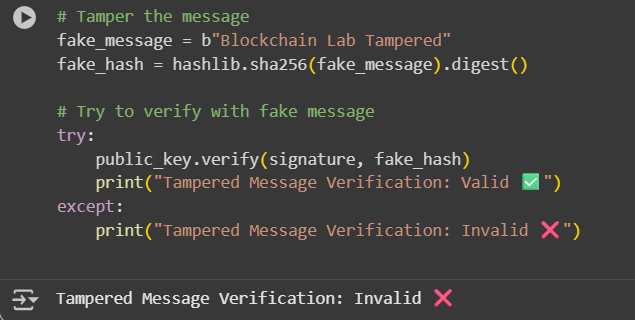
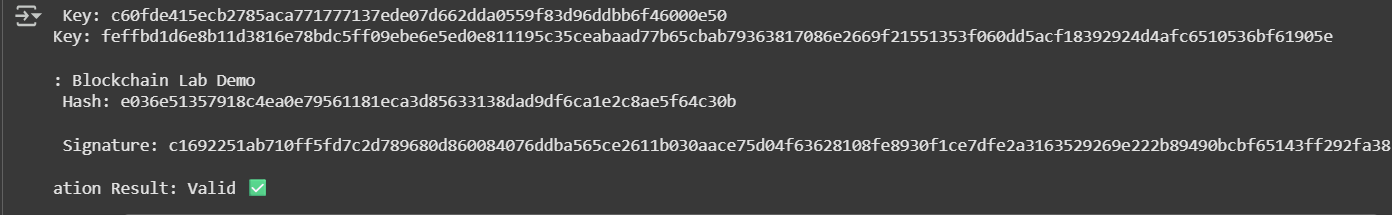
1. Key Generation → Private key & Public key.
2. Signing → Hash the message, then sign using the private key.
3. Verification → Use the public key to check if the signature is valid.



**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:**

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

