Lab Assignment No.03 ECDSA

Q1)) is a numeric value created using aprivate key to Prove the authenticity and integrity of amessage

* itis important Decause it ensures

① Authentication

② Integrity

3) Non-repudiation

Q2)) * hashing: is used to generate a fixed-size summary of the message, making the siging Process ~~mor~~more efficient

Q3) * Encryption : Protect data confidentiality. Public key to encrypt, Private key to decrypt

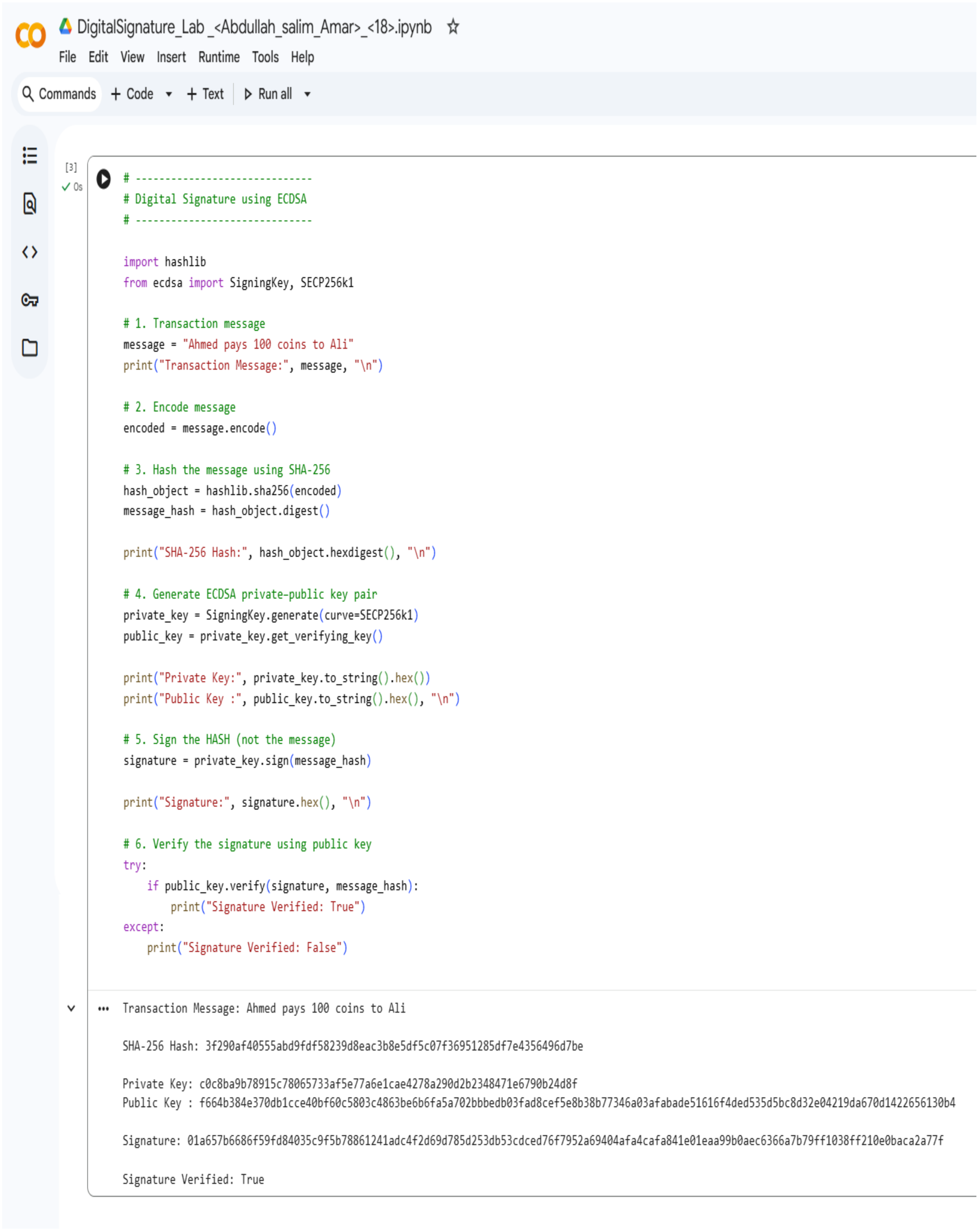
* Digital Signature: Prove data authenticity and non-repudiation.
- Private key to sign, public key to verify

Q4)) In ECDSA, the signature is created using the sender's Private key. only the sender can Produce avalid signature, so they cannot deny the message after signing

Q5) will cause verification to fail, as the signature depends on the hash of the original message

Q6) SHA-256 Provides Strong collision resistance and fixed-Length

Q7) used for official documents sush as government or banking and Password Storage



DigitalSignature_Lab _<Abdullah_salim_Amar>_<18>.ipynb

```python
# -----------------------------
# Digital Signature using ECDSA
# -----------------------------

import hashlib
from ecdsa import SigningKey, SECP256k1

# 1. Transaction message
message = "Ahmed pays 100 coins to Ali"
print("Transaction Message:", message, "\n")

# 2. Encode message
encoded = message.encode()

# 3. Hash the message using SHA-256
hash_object = hashlib.sha256(encoded)
message_hash = hash_object.digest()

print("SHA-256 Hash:", hash_object.hexdigest(), "\n")

# 4. Generate ECDSA private-public key pair
private_key = SigningKey.generate(curve=SECP256k1)
public_key = private_key.get_verifying_key()

print("Private Key:", private_key.to_string().hex())
print("Public Key :", public_key.to_string().hex(), "\n")

# 5. Sign the HASH (not the message)
signature = private_key.sign(message_hash)

print("Signature:", signature.hex(), "\n")

# 6. Verify the signature using public key
try:
    if public_key.verify(signature, message_hash):
        print("Signature Verified: True")
except:
    print("Signature Verified: False")
```

```
Transaction Message: Ahmed pays 100 coins to Ali

SHA-256 Hash: 3f290af40555abd9fdf58239d8eac3b8e5df5c07f36951285df7e4356496d7be

Private Key: c0c8ba9b78915c78065733af5e77a6e1cae4278a290d2b2348471e6790b24d8f
Public Key : f664b384e370db1cce40bf60c5803c4863be6b6fa5a702bbbedb03fad8cef5e8b38b77346a03afabade51616f4ded535d5bc8d32e04219da670d1422656130b4

Signature: 01a657b6686f59fd84035c9f5b78861241adc4f2d69d785d253db53cdced76f7952a69404afa4cafa841e01eaa99b0aec6366a7b79ff1038ff210e0baca2a77f

Signature Verified: True
```