**Week 9**

**1. Implement SHA-1, SHA-2 and SHA-3 using SageMath.**

import hashlib

text = "CyberSecurity"

sha1 = hashlib.sha1(text.encode()).hexdigest()

sha256 = hashlib.sha256(text.encode()).hexdigest()

sha3 = hashlib.sha3\_256(text.encode()).hexdigest()

print("SHA-1:", sha1)

print("SHA-2 (SHA-256):", sha256)

print("SHA-3 (SHA3-256):", sha3)

**2. Implement MD-5 using SageMath.**

import hashlib

text = "CyberSecurity"

md5\_hash = hashlib.md5(text.encode()).hexdigest()

print("MD5:", md5\_hash)

**3. Demonstrate Hash-based Message Authentication Code (HMAC) implementation using Python.**

import hmac

import hashlib

key = b'secret'

message = b'CyberSecurity'

h = hmac.new(key, message, hashlib.sha256)

print("HMAC:", h.hexdigest())

**4. Compare Message Authentication Code (MAC) and Digital Signatures on parameters like how they use keys, security level, and computational cost by simulating both in Python.**

# Simulated MAC using HMAC

import hmac, hashlib

mac\_key = b'secret'

msg = b'Hello'

mac = hmac.new(mac\_key, msg, hashlib.sha256).hexdigest()

# Simulated Digital Signature (concept)

from Crypto.PublicKey import RSA

from Crypto.Signature import pkcs1\_15

from Crypto.Hash import SHA256

key = RSA.generate(1024)

msg\_hash = SHA256.new(msg)

signature = pkcs1\_15.new(key).sign(msg\_hash)

# Verify

try:

pkcs1\_15.new(key.publickey()).verify(msg\_hash, signature)

print("Signature verified.")

except:

print("Signature verification failed.")

print("MAC:", mac)

print("Signature:", signature.hex()[:32], "...") # partial