**WEEK 8**

### 1. Find Inverse of a number under modulo(p) (where p is a prime number) using SageMath

a = 17

p = 43 # p should be a prime number

inv = inverse\_mod(a, p)

print(f"Inverse of {a} mod {p} is {inv}")

### 2. Check whether a set is a Group, Ring, or Field using SageMath

# Example: Set of integers modulo 7

R = Integers(7)

print("Is Group:", R.is\_commutative())

print("Is Ring:", isinstance(R, Ring))

print("Is Field:", isinstance(R, Field))

### 3. Implement Elliptic Curve Cryptography using SageMath

E = EllipticCurve(GF(97), [2, 3]) # y^2 = x^3 + 2x + 3 over GF(97)

P = E.random\_point()

Q = 2 \* P

print("Base Point P:", P)

print("2P =", Q)

### 4. Simulate Elliptic Curve Diffie-Hellman (ECDH) Key Exchange in SageMath

E = EllipticCurve(GF(97), [2, 3])

G = E.random\_point()

# Alice

a = 5

A = a \* G

# Bob

b = 7

B = b \* G

# Shared secret

S1 = a \* B

S2 = b \* A

print("Shared Secret (Alice):", S1)

print("Shared Secret (Bob):", S2)

### 5. RSA Algorithm using SageMath

p = random\_prime(100, lbound=50)

q = random\_prime(100, lbound=50)

n = p \* q

phi = (p - 1) \* (q - 1)

e = 3

while gcd(e, phi) != 1:

e += 2

d = inverse\_mod(e, phi)

message = 42

cipher = power\_mod(message, e, n)

decrypted = power\_mod(cipher, d, n)

print("Public Key:", (e, n))

print("Private Key:", d)

print("Encrypted:", cipher)

print("Decrypted:", decrypted)