**ST. FRANCIS INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SECURITY LAB**

**Experiment – 5: Implementation of Digital Signature Scheme**

**Aim:** Write a program to implement RSA Digital Signature Scheme.

**Objective:** After performing the experiment, the students will be able to –

* To understand the RSA Digital Signature Scheme

**Lab objective mapped:**

L502.3: Students should be able to Analyze and evaluate performance of digital signature scheme and demonstrate key management.

**Prerequisite:** Basic knowledge of Digital Signature.

**Requirements:** PYTHON

**Pre-Experiment Theory:**

**Digital Signature Requirements:**

* The signature must be a bit pattern that depends on the message being signed.
* The signature must use some information unique to the sender to prevent both forgery and denial.
* It must be relatively easy to produce the digital signature.
* It must be relatively easy to recognize and verify the digital signature.
* It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

RSA can be used for signing and verifying a message. In this case it is called the RSA digital signature scheme.

The digital signature scheme changes the roles of the private and public keys.

1. First, the private and public key of the sender, not the receiver are used.
2. Second, the sender uses his/her own private key to sign the document; the receiver uses the sender’s public key to verify it.

**Generation of RSA Key Pair**

* Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem.
* Alice chooses two primes p and q and calculates n= p x q.
* Alice calculates ϕ(n) = (p – 1)(q – 1).
* She then chooses e, the public exponent, and calculates d, the private exponent such that e x d = 1 mod ϕ(n) .
* Alice keeps d, she publicly announces n and e.

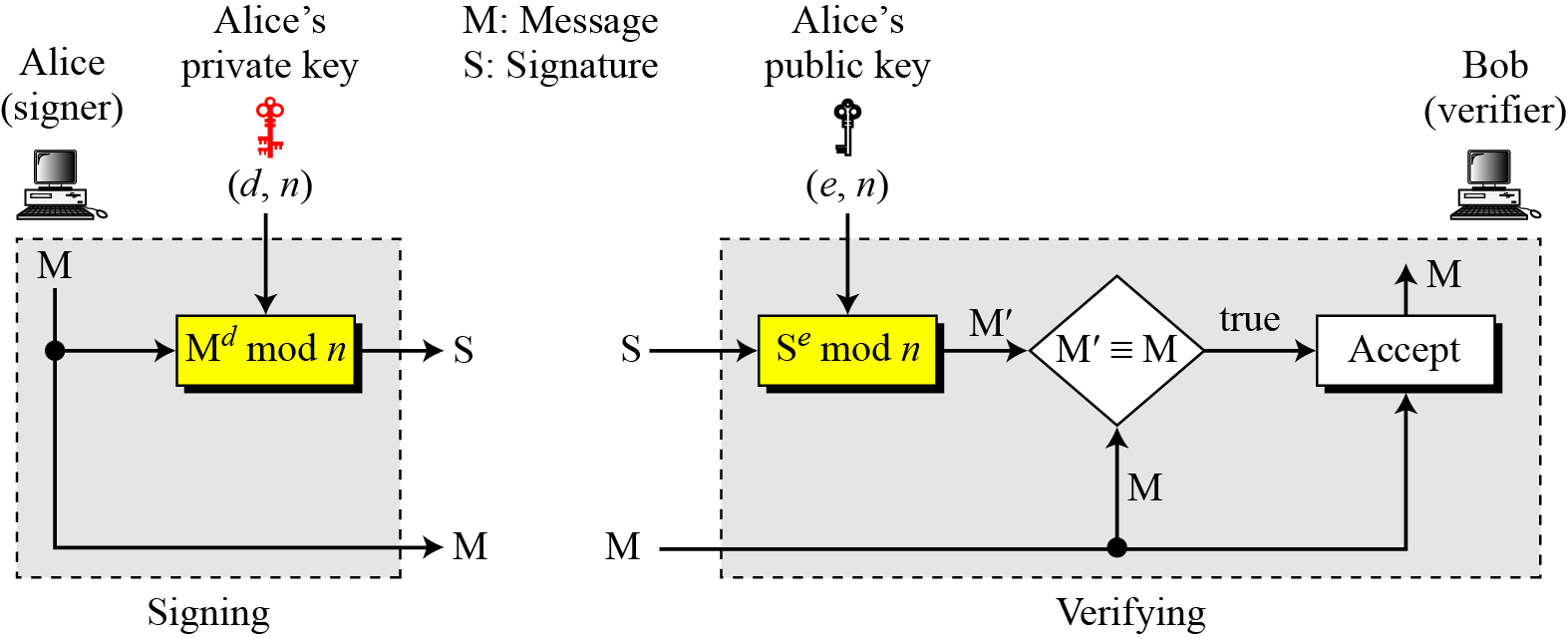
**Signing**

Alice creates a signature out of the message using her private exponent, S = mod n and sends the message and the signature to Bob.

**Verifying**

Bob receives M and S. Bob applies Alice’s public exponent to the signature to create a copy of the message = mod n.

Bob compares the value of with the value of M. If the two values are congruent, Bob accepts the message.



**Procedure:**

Write a program in Python for key generation, signing and verification using the RSA algorithm.

* 1. For Key generation, ask the user to enter the value of prime numbers p & q and a public key ‘e’. (Note that values of p, q & e cannot be random, they should satisfy criteria as per RSA algorithm)
  2. Program should calculate the private key element ‘d’ using Extended Euclidean Algorithm.
  3. Provide a set of public (e, n) and private key (d, n) as the output to the user.
  4. For digital sign generation, ask the user to enter the Message. Using a private key, calculate the digital sign ‘S’ and output the same.
  5. For sign verification, ask the user to enter the sign ‘S’ and the Message ‘M’. Using the public key, calculate the M’ value. Compare this M’ value with Message ‘M’. If they are the same, display that “The message is authenticate” else display “Message is altered, Discard.”

1. Test the output of program for following exercises:
2. For p=11, q=3 and e=3, find public and private keys using the RSA algorithm. For message, M=10, Find the digital signature value. Also show how any receiver of this signature will verify it.
3. For p=7, q=11, e=13. Find public and private keys using the RSA algorithm. For message, M=5, Find the digital signature value. Also show how any receiver of this signature will verify it.

**Output:**

1. Attach the complete code performing key generation, signature generation and verification.
2. Attach the program output for key generation (display public key & private key), digital sign ‘S’ and verification code M’ for the inputs given in the exercise above.

**Post Experimental Exercise:**

1. Solve both the exercises mentioned in the procedure on journal sheets. [Theoretical result and attached code’s output should match].
2. What are the different attacks possible on RSA digital signature scheme?

**Conclusion:**

In RSA Digital Signature Scheme the signing and verifying sites use the same function but with different parameters. The verifier compares the message and the output of the function for congruence. If the result is true, the message is accepted.

Code:

from sympy import mod\_inverse, isprime

def generate\_keys():

# Get prime numbers p and q

while True:

p = int(input("Enter a prime number p: "))

q = int(input("Enter a prime number q: "))

if not (isprime(p) and isprime(q)):

print("Both p and q must be prime numbers. Please try again.")

elif p == q:

print("p and q cannot be the same. Please try again.")

else:

break

# Calculate n and phi(n)

n = p \* q

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

# Get public key e

while True:

e = int(input(f"Enter a public key 'e' such that 1 < e < {phi} and gcd(e, {phi}) = 1: "))

if 1 < e < phi and gcd(e, phi) == 1:

break

else:

print(f"Invalid 'e'. Make sure 1 < e < {phi} and gcd(e, {phi}) = 1. Please try again.")

# Calculate private key d using Extended Euclidean Algorithm

d = mod\_inverse(e, phi)

# Return public and private keys

return (e, n), (d, n)

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def sign\_message(private\_key, message):

d, n = private\_key

# Ensure the message is treated as an integer

message = int(message)

S = pow(message, d, n)

return S

def verify\_signature(public\_key, signature, original\_message):

e, n = public\_key

# Calculate M' using public key (M' = S^e mod n)

m\_prime = pow(signature, e, n)

return m\_prime == original\_message

if \_\_name\_\_ == "\_\_main\_\_":

print("RSA Key Generation:")

public\_key, private\_key = generate\_keys()

print(f"Public Key: {public\_key}")

print(f"Private Key: {private\_key}")

print("\nDigital Signature Generation:")

message = int(input("Enter the message to sign (as an integer): "))

signature = sign\_message(private\_key, message)

print(f"Digital Signature: {signature}")

print("\nSignature Verification:")

verify\_msg = int(input("Enter the original message to verify (as an integer): "))

is\_authentic = verify\_signature(public\_key, signature, verify\_msg)

if is\_authentic:

print("The message is authentic.")

else:

print("Message is altered, discard.")

**Output:**

Q.1 Q.2