**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

**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**

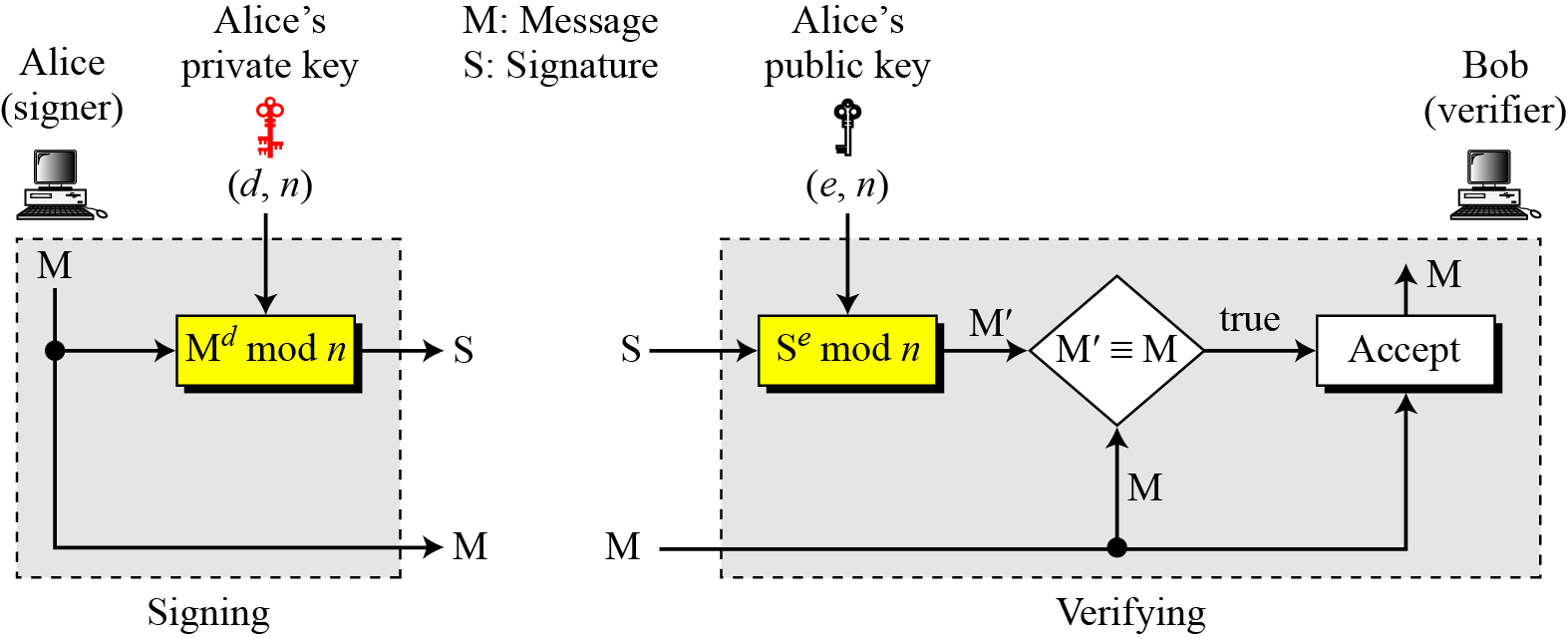
* Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem.
* Alice chooses two primes p and q and calculates n= p x q.
* Alice calculates ϕ(n) = (p – 1)(q – 1).
* She then chooses e, the public exponent, and calculates d, the private exponent such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

* 1. For Key generation, ask 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 user to enter the Message. Using private key calculate the digital sign ‘S’ and output the same.
  5. For sign verification, ask user to enter the sign ‘S’ and the Message ‘M’. Using public key calculate the M’ value. Compare this M’ value with Message ‘M’. If they are 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 key using 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 find public and private key using 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.

**Output:**

1. **Attach the complete code performing key generation, signature generation and verification.**

**CODE:**

# Function to compute gcd and the coefficients for Extended Euclidean Algorithm

def extended\_gcd(a, b):

if a == 0:

return b, 0, 1

gcd, x1, y1 = extended\_gcd(b % a, a)

x = y1 - (b // a) \* x1

y = x1

return gcd, x, y

# Function to compute modular inverse of e under modulo phi\_n

def mod\_inverse(e, phi\_n):

gcd, x, y = extended\_gcd(e, phi\_n)

if gcd != 1:

raise ValueError("Modular inverse does not exist")

else:

return x % phi\_n

# Function to compute n (part of both public and private keys)

def compute\_n(p, q):

return p \* q

# Function to compute φ(n)

def compute\_phi\_n(p, q):

return (p - 1) \* (q - 1)

# Function to generate digital signature 'S'

def generate\_signature(M, d, n):

S = pow(M, d, n)

return S

# Function to verify digital signature

def verify\_signature(S, e, n):

M\_prime = pow(S, e, n)

return M\_prime

# Main program

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

# Key generation

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

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

e = int(input("Enter public key e: "))

n = compute\_n(p, q)

phi\_n = compute\_phi\_n(p, q)

d = mod\_inverse(e, phi\_n)

print(f"Public Key (e, n): ({e}, {n})")

print(f"Private Key (d, n): ({d}, {n})")

# Digital signature generation

M = int(input("\nEnter the message to sign: "))

S = generate\_signature(M, d, n)

print(f"Digital Signature is: {S}")

# Signature verification

M\_received = int(input("\nEnter the received message to verify: "))

M\_prime = verify\_signature(S, e, n)

if M\_prime == M\_received:

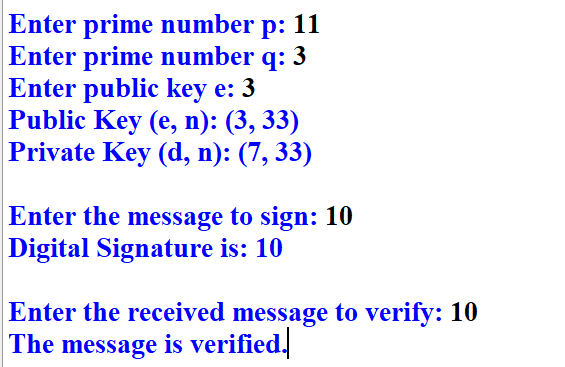
print("The message is verified.")

else:

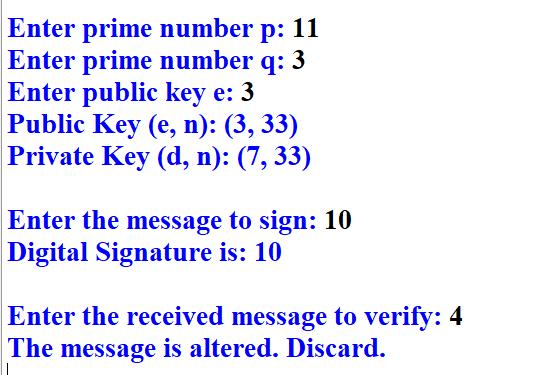
print("The message is altered. Discard.")

1. **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.**
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.

For the correct value of message:

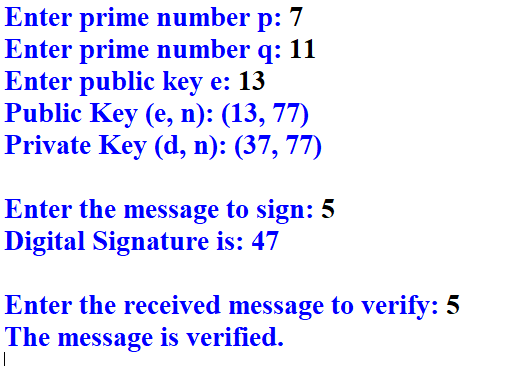


For the incorrect value of message:



1. 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.

For the correct value of message:



For the incorrect value of message:

