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.

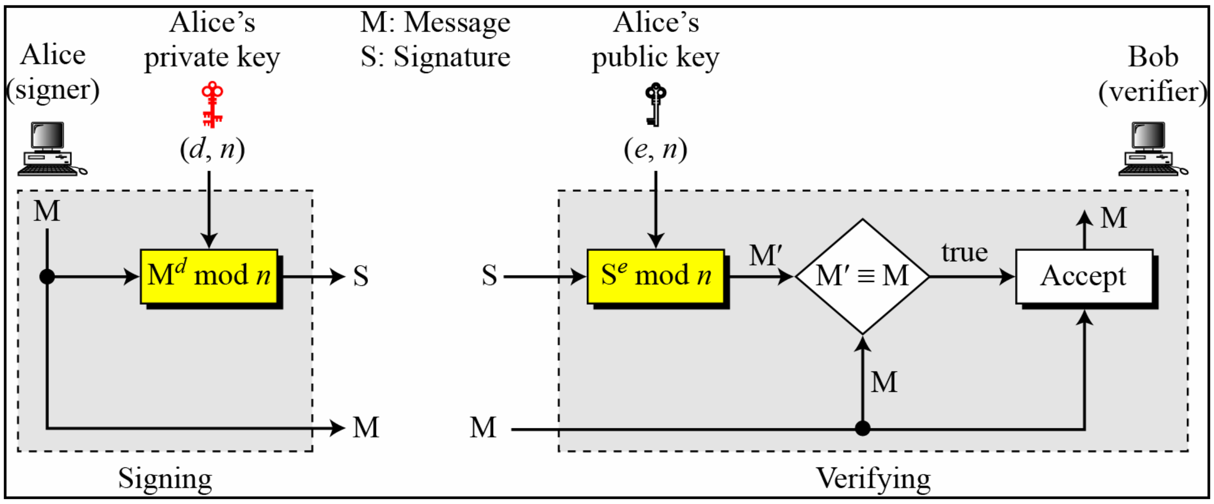
Generation of RSA Key Pair
- Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem.
- Alice chooses two primes p and q and calculates n= p x q.
- Alice calculates $\phi(n) = (p - 1)(q - 1)$.
- She then chooses e, the public exponent, and calculates d, the private exponent such that e x d = 1 mod $\phi(n)$.
- 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.

a. 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)
b. Program should calculate the private key element 'd' using Extended Euclidean Algorithm.
c. Provide a set of public (e, n) and private key (d, n) as the output to the user.
d. For digital sign generation, ask the user to enter the Message. Using a private key, calculate the digital sign 'S' and output the same.
e. 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."

2. Test the output of program for following exercises:

   a. 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.
   b. 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.

Name: Vishal Rajesh Mahajan SL EXP 5

Class: TE IT A Roll No: 62

Code:

```
def extended_gcd(a, b):
   """Extended Euclidean Algorithm to find the greatest common divisor of a
and b, and the coefficients of Bézout's identity."""
   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

def mod_inverse(e, phi):
   """Calculate the modular inverse of e modulo phi using the Extended
Euclidean Algorithm."""
   gcd, x, _ = extended_gcd(e, phi)
   if gcd != 1:
      raise ValueError("Modular inverse does not exist.")
   return x % phi

def rsa_key_generation(p, q, e):
   """Generate RSA public and private keys given primes p, q, and public
exponent e."""
   if not (is_prime(p) and is_prime(q)):
      raise ValueError("p and q must be prime numbers.")
   if not (1 < e < (p - 1) * (q - 1)):
      raise ValueError("e must be in the range 1 < e < (p-1)*(q-1)")

   n = p * q
   phi = (p - 1) * (q - 1)
   d = mod_inverse(e, phi)

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

Name: Vishal Rajesh Mahajan SL EXP 5
Class: TE IT A Roll No: 62

```
def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

def encrypt_decrypt(message, key, n):
    """Encrypt or decrypt a message using RSA."""
    e, d = key
    return pow(message, e, n) if e else pow(message, d, n)

def rsa_sign(message, private_key):
    """Generate a digital signature for a message using the private key."""
    d, n = private_key
    return encrypt_decrypt(message, (None, d), n)

def rsa_verify(message, signature, public_key):
    """Verify a digital signature for a message using the public key."""
    e, n = public_key
    message_prime = encrypt_decrypt(signature, (e, None), n)
    return message_prime == message
```

Name: Vishal Rajesh Mahajan SL EXP 5

Class: TE IT A Roll No: 62

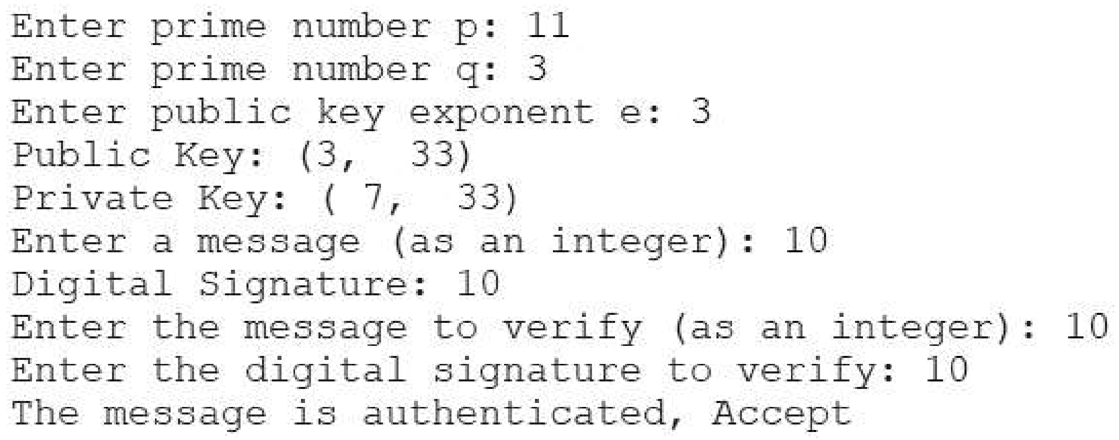
```
def main():
    # Key Generation
    p = int(input("Enter prime number p: "))
    q = int(input("Enter prime number q: "))
    e = int(input("Enter public key exponent e: "))

    public_key, private_key = rsa_key_generation(p, q, e)
    print(f"Public Key: ({public_key[0]},  {public_key[1]})")
    print(f"Private Key: ( {private_key[0]},  {private_key[1]})")

     # Digital Signature Generation
    message = int(input("Enter a message (as an integer): "))
    signature = rsa_sign(message, private_key)
    print(f"Digital Signature: {signature}")

    # Signature Verification
    message_to_verify = int(input("Enter the message to verify (as an integer): "))
    signature_to_verify = int(input("Enter the digital signature to verify: "))

    if rsa_verify(message_to_verify, signature_to_verify, public_key):
        print("The message is authenticated, Accept")
    else:
        print("Message is altered, Discard!")

if __name__ == "__main__":
    main()
```

Name: Vishal Rajesh Mahajan SL EXP 5

Class: TE IT A Roll No: 62

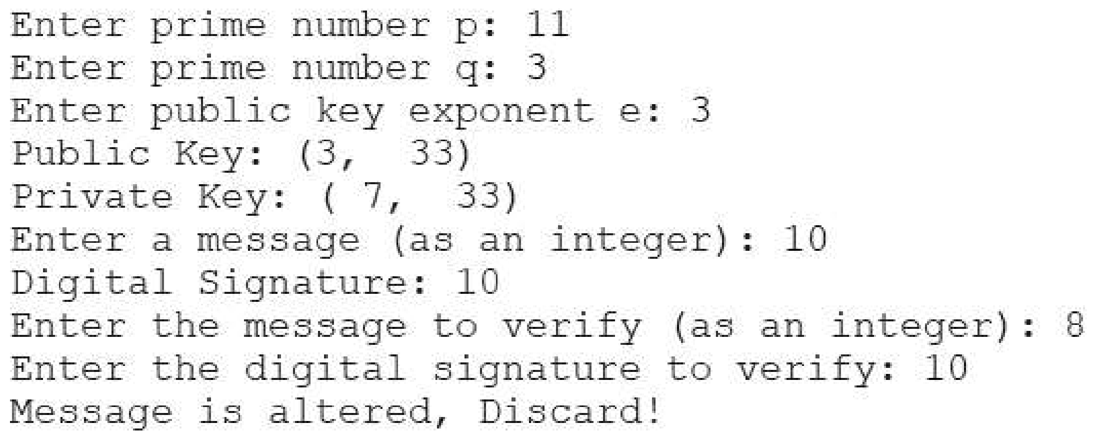
Q. For p=11, q=3 and e=3, find public and private keys using the RSA algorithm. For message, M=10, Find the digital signature value. Also show how any receiver of this signature will verify it.

Case 1: where M = M' then the message is Authenticate ,Accept



```
Enter prime number p: 11
Enter prime number q: 3
Enter public key exponent e: 3
Public Key: (3,  33)
Private Key: ( 7,  33)
Enter a message (as an integer): 10
Digital Signature: 10
Enter the message to verify (as an integer): 10
Enter the digital signature to verify: 10
The message is authenticated, Accept
```

Case 2:where M ≠ M' then the message is altered ,Reject



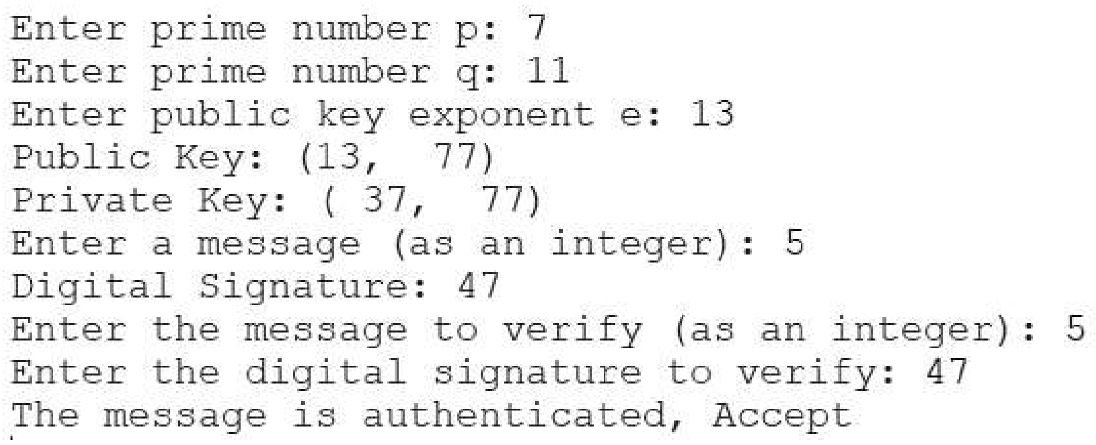
```
Enter prime number p: 11
Enter prime number q: 3
Enter public key exponent e: 3
Public Key: (3,  33)
Private Key: ( 7,  33)
Enter a message (as an integer): 10
Digital Signature: 10
Enter the message to verify (as an integer): 8
Enter the digital signature to verify: 10
Message is altered, Discard!
```

Q.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.

Name: Vishal Rajesh Mahajan SL EXP 5

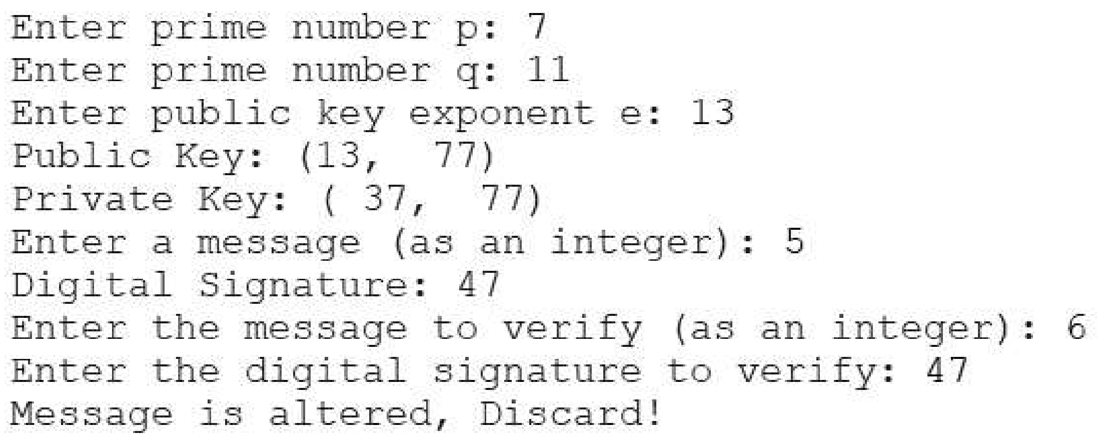
Class: TE IT A Roll No: 62

Case 1: where M = M’ then the message is Authenticate ,Accept



```
Enter prime number p: 7
Enter prime number q: 11
Enter public key exponent e: 13
Public Key: (13,  77)
Private Key: ( 37,  77)
Enter a message (as an integer): 5
Digital Signature: 47
Enter the message to verify (as an integer): 5
Enter the digital signature to verify: 47
The message is authenticated, Accept
```

Case 2: where M ≠ M’ then the message is altered ,Reject



```
Enter prime number p: 7
Enter prime number q: 11
Enter public key exponent e: 13
Public Key: (13,  77)
Private Key: ( 37,  77)
Enter a message (as an integer): 5
Digital Signature: 47
Enter the message to verify (as an integer): 6
Enter the digital signature to verify: 47
Message is altered, Discard!
```