Don Bosco Institute of Technology, Mumbai 400070
Department of Information Technology

Experiment No. : 6

Date: 13/09/2024

Title : Implement the RSA cryptosystem

Problem Definition : Implement asymmetric algorithm RSA and encrypt the given plain-text and decrypt it back.

Pre-requisite : Asymmetric cryptography

Theory :

RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty offactoring the product of two large prime numbers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was not declassified until 1997 .
A user of RSA creates and then publishes a public key based on the two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message.

Procedure/ Algorithm :

Key generation:

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.
2. Compute n = pq.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
3. Compute φ(n) = φ(p)φ(q) = (p − 1)(q − 1) = n - (p + q -1), where φ is Euler's totient function.
4. Choose an integer e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1; i.e., e and φ(n) are coprime.
   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

5. Determine d as d ≡ e−1 (mod φ(n)); i.e., d is the multiplicative inverse of e (modulo φ(n)).
   This is more clearly stated as: solve for d given d · e ≡ 1 (mod φ(n))
   This is often computed using the extended Euclidean algorithm. Using the pseudo code in the Modular integers section, inputs a and n correspond to e and φ(n), respectively.
   d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

- An alternative, used by PKCS#1, is to choose d matching de ≡ 1 (mod λ) with λ = lcm(p − 1, q − 1), where lcm is the least common multiple. Using λ instead of φ(n) allows more choices for d. λ can also be defined using the Carmichael function, λ(n).

Encryption:

Alice transmits her public key (n, e) to Bob and keeps the private key d secret. Bob then wishes to send message M to Alice.

He first turns M into an integer m, such that 0 ≤ m < n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to

$$c \equiv m^e \pmod{n}$$

This can be done efficiently, even for 500-bit numbers, using Modular exponentiation. Bob then transmits c to Alice.

Decryption:

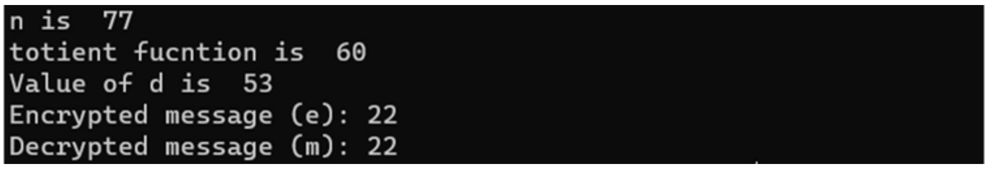
Alice can recover m from c by using her private key exponent d via computing

$$m \equiv c^d \pmod{n}$$

Given m, she can recover the original message M by reversing the padding scheme.

Program code with Result/Output :

```
p=int(input("Enter 1st prime number: "))
q=int(input("Enter 2nd prime number: "))
e=int(input("Enter value of e: "))
m=int(input("Enter value of m: "))
n=p*q
print("n is ", n)
totient_function= (p-1)*(q-1)
print("totient fucntion is ", totient_function)
for k in range(1,n):
 d=(1+(k*totient_function))/e
 if (d%1==0):
 breakDon Bosco InsƟ tute of Technology, Mumbai 400070
Department of InformaƟ on Technology
dx=int(d)
print("Value of d is ",dx)
c = pow(m, e, n)
print("Encrypted message (e):", c)
decrypted = pow(c, dx, n)
print("Decrypted message (m):", decrypted)
```

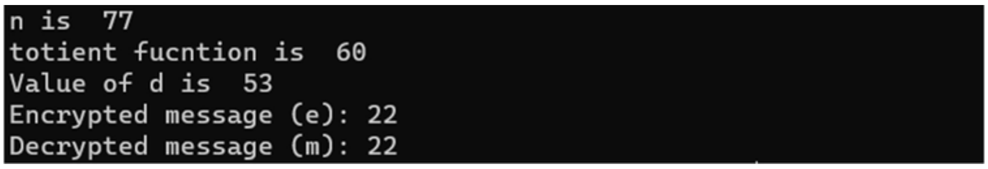


```
n is  77
totient fucntion is  60
Value of d is  53
Encrypted message (e): 22
Decrypted message (m): 22
```

References:
1. http://en.wikipedia.org/wiki/RSA_%28cryptosystem%29
2. http://courses.cs.vt.edu/~cs5204/fall00/protecƟ on/rsa.html

Lab practice (optional) :

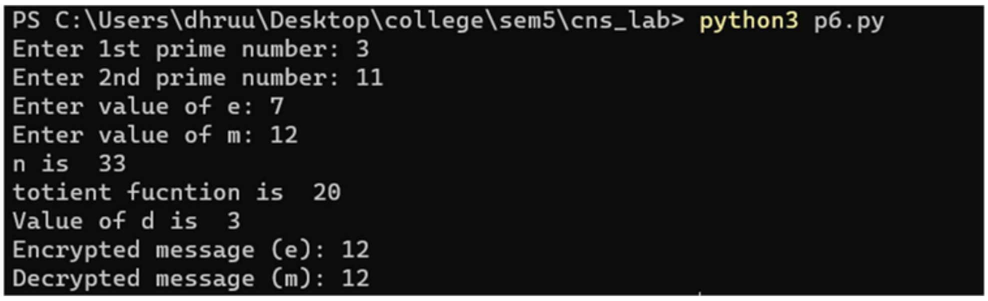
L1 : Given p=7,q=11 and e=17. Find d and encrypt the message m=22. Also decrypt the cipher text.



```
n is  77
totient fucntion is  60
Value of d is  53
Encrypted message (e): 22
Decrypted message (m): 22
```

Questions (Short, Long, MCQs) (optional) :

L1 : Given p=3,q=11 and e=7. Find d and encrypt the message m=12. Also decrypt the cipher text.



```
PS C:\Users\dhruu\Desktop\college\sem5\cns_lab> python3 p6.py
Enter 1st prime number: 3
Enter 2nd prime number: 11
Enter value of e: 7
Enter value of m: 12
n is  33
totient fucntion is  20
Value of d is  3
Encrypted message (e): 12
Decrypted message (m): 12
```

L2 : Write short notes on RSA cryptosystem.

RSA (Rivest-Shamir-Adleman) is a widely used public-key cryptosystem that provides secure encryption and digital signatures. Here are some key points about the RSA cryptosystem:

1. Public-Key Cryptosystem: RSA is a public-key cryptosystem, which means it uses a pair of keys: a public key for encryption and a private key for decryption. These keys are mathematically related but computationally difficult to reverse engineer.
2. Key Generation: To use RSA, two large prime numbers, p and q, are selected. The product of these primes, n (n = p * q), becomes the modulus for both the public and private keys. The public key also includes an exponent (typically e), while the private key includes another exponent (typically d).
3. Encryption: To encrypt a message, the sender uses the recipient's public key. The plaintext message is converted into a numerical value, and then modular exponentiation is applied using the public exponent and modulus. The result is the ciphertext.
4. Decryption: Only the recipient, who possesses the corresponding private key, can decrypt the ciphertext. The recipient applies modular exponentiation using the private exponent and modulus to retrieve the original plaintext.
5. Security: RSA's security relies on the difficulty of factoring the modulus n into its two prime factors (p and q). The larger the key size (i.e., the longer the primes used), the more secure the RSA encryption. RSA is considered secure because factoring large numbers into their primes is a computationally intensive task, especially when using long keys.
6. Digital Signatures: RSA can also be used for digital signatures. In this case, the sender uses their private key to encrypt a hash value of the message. The recipient can verify the signature using the sender's public key and compare it with the hash of the received message.
7. Applications: RSA is widely used in various applications, including securing internet communications (HTTPS), email encryption (PGP), secure file transfer (SFTP), digital certificates, and secure access to networks and systems.
8. Key Length: The security of RSA depends on the length of the keys. Key lengths of 2048 bits or 3072 bits are commonly used today, and longer keys are recommended for higher levels of security.
9. Performance: RSA encryption and decryption can be computationally intensive, especially with longer keys. As a result, RSA is oŌ en used in combination with symmetric-key encryption algorithms, where a symmetric key is used for bulk data encryption, and RSA is used to securely exchange the symmetric key.
10. Challenges: RSA is vulnerable to attacks if key management is not handled properly. For example, if an attacker gains access to the private key, they can decrypt all encrypted data. Additionally, advancements in quantum computing could potentially threaten the security of RSA by making factorization significantly faster.

In summary, RSA is a widely adopted and secure public-key cryptosystem that plays a crucial role in securing data transmission and digital signatures in various applications. Its security strength depends on the size of the key used, and it remains a fundamental component of modern cryptography.