ASSIGNMENT NO	B4
TITLE	Implementation of RSA
PROBLEM STATEMENT/ DEFINITION	Implementation of RSA
OBJECTIVE	To understand how RSA algorithm works
OUTCOME	Understaning and implementation of asymmetric encryption using RSA.
S/W PACKAGES AND HARDWARE APPARATUS USED	Core 2 DUO/i3/i5/i7 64-bit processor OS-LINUX 64 bit OS Editor-gedit/Eclipse S/W- C++/JAVA//Python
REFERENCES	1. Bernard Menezes, "Network Security and Cryptography", Cengage Learning India, 2014, ISBN No.: 8131513491 2. Nina Godbole, Sunit Belapure, "Cyber Security", Wiley India, 2014, ISBN No.: 978-81-345-2179-1 3. Atul Kahate, "Cryptography and Network Security", Mc Graw Hill Publication, 2nd Edition, 2008, ISBN: 978-0-07-064823-4 4. William Stallings, "Cryptography and network security principles and practices", Pearson, 6th Edition, ISBN: 978-93-325-1877-3 5. Forouzan, "Cryptography and Network Security (SIE)", Mc Graw Hill, ISBN, 007070208X, 9780070702080
STEPS	1.Key generation 2.Encryption 3. Decryption
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment No. 3. Problem Definition 4. Learning Objective

	5. Learning Outcome 6. Concepts Related Theory 7. Algorithm 8. Test Cases 9. Conclusion/Analysis

Prerequisites:

Discrete mathematics, any programming language Java/C++/Python.

Concepts Related Theory:

RSA algorithm involves three steps

1. Key Generation
2. Encryption
3. Decryption

1. Key Generation

Key Generation Algorithm

The key generation algorithm works as follows:

1. Generate two large random primes, p and q, of approximately equal size such that their product $n=pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n=pq$ and $\phi=(p-1)(q-1)$
3. Choose an integer e, $1<e<\phi$, such that $\gcd(e,\phi)=1$
4. Compute the secret exponent d, $1<d<\phi$, such that $ed\equiv 1 \bmod \phi$
5. The public key is (n,e) and the private key (n,d). Keep all the values d, p, q and ϕ secret.

Note:

- n is known as the modulus.
- e is known as the public exponent or encryption exponent or just the exponent.
- d is known as the secret exponent or decryption exponent.

A practical key generation algorithm

A a practical algorithm to generate an RSA key pair is given below. Typical bit lengths are $k=1024,2048,3072,4096,\ldots$, with increasing computational expense for larger values. You will not go far wrong if you choose e as 65537 (=0x10001) in step (1).

Algorithm: Generate an RSA key pair.
INPUT: Required modulus bit length, k.
OUTPUT: An RSA key pair $((N,e),d)$ where N is the modulus, the product of two primes ($N=pq$) not exceeding k bits in length; e is the public exponent, a number less than and coprime to $(p-1)(q-1)$; and d is the private exponent such that $ed\equiv 1 \bmod (p-1)(q-1)$.

1. Select a value of e from $3,5,17,257,65537$
2. **repeat**

p ← genprime(k/2)
until $(p \bmod e)\neq 1$
repeat
q ← genprime(k - k/2)
until $(q \bmod e)\neq 1$
N ← pq
L ← (p-1)(q-1)
d ← modinv(e, L)
return (N,e,d)

The function `genprime(b)` returns a prime of exactly b bits, with the bth bit set to 1. Note that the operation $k/2$ is *integer* division giving the integer quotient with no fraction.

If you've chosen $e=65537$ then the chances are that the first prime returned in steps (3) and (6) will pass the tests in steps (4) and (7), so each repeat-until loop will most likely just take one iteration. The final value of N may have a bit length slightly short of the target k. This actually does not matter too much (providing the message m is always < N), but some schemes require a modulus of exact length. If this is the case, then just repeat the entire algorithm until you get one. It should not take too many goes.

• Encryption:

Sender A does the following:-

1. Obtains the recipient B's public key (n,e)
2. Represents the plaintext message as a positive integer M with $1<M<n$
3. Computes the ciphertext $C=M^{e} \bmod n$
4. Sends the ciphertext C to B.

• Decryption

Recipient B does the following:-

1. Uses his private key (n,d) to compute $m=C^{d} \bmod n$
2. Extracts the plaintext from the message representative m

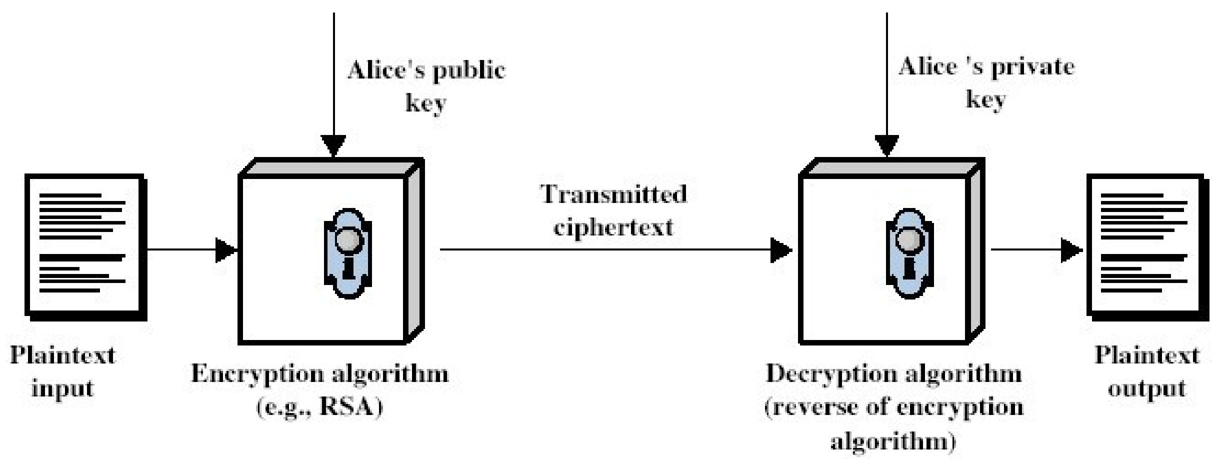

Figure: RSA encryption and decryption

RSA for Digital signing

Sender A does the following:-

1. Creates a message digest of the information to be sent.
2. Represents this digest as an integer m between 1 and $n-1$
3. Uses her *private* key (n,d) to compute the signature $s=m^{d} \bmod n$

4. Sends this signature s to the recipient, B.

Signature verification

Recipient B does the following (*older method*):-

1. Uses sender A's public key (n,e) to compute integer $v=s^e \bmod n$
2. Extracts the message digest H from this integer.
3. Independently computes the message digest H' of the information that has been signed.
4. If both message digests are identical, i.e. $H=H'$, the signature is valid.

***More secure method*:-**

1. Uses sender A's public key (n,e) to compute integer $v=s^e \bmod n$
2. Independently computes the message digest H' of the information that has been signed.
3. Computes the expected representative integer v' by encoding the expected message digest H'
4. If $v=v'$, the signature is valid.

Test Cases:

Generate RSA key pair for different values of p and q. Apply encryption and decryption on the given string.