EXPERIMENT 4

RSA Algorithm

3.1 Aim

To write a program for implementing the RSA algorithm.

3.2 Algorithm

1. START
2. Create a Node.js project to work .
3. Create the index.js file as the starting point of application and rsa.js file to declare class for encryption operations.
4. Install npm package *big-integer* which allows arithmetic operations on integers .
5. Assign the message to be encrypted into a variable named *message .*
6. Using the library function generate 2 random prime numbers and assign them to primeNumber1 and *primeNumber2* respectively.
7. Calculate the totient function; $\phi(n)=(p-1)(q-1)$.
8. Select an integer *e*, such that *e* is co-prime to $\phi(n)$ and $1 < e < \phi(n)$.
9. Calculate n as p multiply q and d such that $e.d=1 \bmod \phi(n)$.
10. The pair of numbers makes up the public key.
11. Log the public key (The pair of numbers *(n,e)*) , private key (The pair of numbers (n,d)) in the console.
12. Encode the given message of alphanumeric characters to standard utf-8 decimal encoding as *encoded_message .*
13. Encoded text is encrypted as *encrypted_message* = (*encoded_message ^ e) mod n .*
14. *encrypted_message* is decrypted by *decrypted_message* = (*encrypted_message ^ d)mod n.*
15. *Decrypted_message* is then reverted back to alphanumeric string form .

3.3 Program

File 1: index.js

```
const RSA = require('./rsa');

// Message
const message = 'This is a sample text';

// Generate RSA keys
const keys = RSA.generate(250);

console.log('Keys');
```

```
console.log('n:', keys.n.toString());
console.log('d:', keys.d.toString());
console.log('e:', keys.e.toString());

const encoded_message = RSA.encode(message);
const encrypted_message = RSA.encrypt(encoded_message, keys.n, keys.e);
const decrypted_message = RSA.decrypt(encrypted_message, keys.d, keys.n);
const decoded_message = RSA.decode(decrypted_message);

console.log('Message:', message);
console.log('Encoded:', encoded_message.toString());
console.log('Encrypted:', encrypted_message.toString());
console.log('Decrypted:', decrypted_message.toString());
console.log('Decoded:', decoded_message.toString());
```

File 2 : rsa.js

```
const bigInt = require('big-integer');

class RSA {
 static randomPrime(bits) {
  const min = bigInt.one.shiftLeft(bits - 1);
  const max = bigInt.one.shiftLeft(bits).prev();

  while (true) {
   let p = bigInt.randBetween(min, max);
   if (p.isProbablePrime(256)) {
    return p;
   }
  }
 }

 static generate(keysize) {
  const e = bigInt(65537);
  let p;
  let q;
  let totient;

  do {
   p = this.randomPrime(keysize / 2);
   q = this.randomPrime(keysize / 2);
   totient = p.prev().multiply(q.prev());
  } while (bigInt.gcd(e, totient).notEquals(1) );

  return {
```

```
    e,
    n: p.multiply(q),
    d: e.modInv(totient),
   };
  }

  static encrypt(encodedMsg, n, e) {
   return bigInt(encodedMsg).modPow(e, n);
  }

  static decrypt(encryptedMsg, d, n) {
   return bigInt(encryptedMsg).modPow(d, n);
  }

  static encode(str) {
   const codes = str
    .split('')
    .map(i => i.charCodeAt())
    .join('');

   return bigInt(codes);
  }

  static decode(code) {
   const stringified = code.toString();
   let string = '';

   for (let i = 0; i < stringified.length; i += 2) {
    let num = Number(stringified.substr(i, 2));

    if (num <= 30) {
     string += String.fromCharCode(Number(stringified.substr(i, 3)));
     i++;
    } else {
     string += String.fromCharCode(num);
    }
   }

   return string;
  }
 }

 module.exports = RSA;
```

3.4 Output

```
PS C:\Users\cinoy\OneDrive\Documents\rsa> node index.js
Keys
n: 741396257400510738304306093410784764076775970960238005835178446972075830071
d: 491341740508054730866089789815075677186533046598063128010279963280602209169
e: 65537
Message: This is a sample text
Encoded: 84104105115321051153297321159710911210810132116101120116
Encrypted: 613099025079440615278808563326536004299304485507152351786946794075825217508
Decrypted: 84104105115321051153297321159710911210810132116101120116
Decoded: This is a sample text
PS C:\Users\cinoy\OneDrive\Documents\rsa>
```

3.5 Result

The RSA algorithm was implemented successfully.