Лабораторна робота 3

Асиметричне шифрування. Алгоритм RSA

Мета:

Дослідити і реалізувати механізм асиметричного алгоритму шифрування RSA.

Завдання

Розробити додаток обміну таємними посиланнями між двома клієнтами за допомогою алгоритму шифрування RSA

- Реалізувати алгоритм генерації ключів (public / private keys) для алгоритму RSA. Створити ключі заданої довжини (напр. 1024 біт)
- Реалізувати та продемонструвати роботу алгоритму шифрування та дешифрування повідомлення RSA
- Підтвердити роботу реалізованого алгоритму шляхом порівняння результату кодування з існуючим алгоритмом (наприклад, використовуючи утиліту openssl або вбудовані системи шифрування обраної мови програмування)

Файл 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 = bigInt.lcm(p.prev(), q.prev());
    } while (
      bigInt.gcd(e, totient).notEquals(1) ||
      p
        .minus(q)
        .abs()
        .shiftRight(keysize / 2 - 100)
        .isZero()
    );

    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;
```

RSA.generate(keysize)

Створює ключ шифрування заданого розміру ключа (у бітах), використовуючи RSA.generate(). Ця функція повертає об’єкт із властивостями public key, private key , exp.

RSA.encode(string)

Перетворює рядок буквено-цифрових символів у стандартне десяткове кодування utf-8. Зашифрувати можна лише числа, тому для шифрування будь-яких нечислових даних необхідне кодування.

RSA.encrypt(data, publicKey, publicExponent)

Шифрує числові дані за допомогою відкритих частин створеного або переданого ключа. Це будуть властивості в об’єкті, які повертає функція.eRSA.generate()

RSA.decrypt(text, privateKey, publicKey)

Використовуючи приватну частину ключа шифрування , розшифровує рядок зашифрованого тексту з цифр. Поверне закодований рядок або просто число, залежно від мети передачі даних

RSA.decode(number)

Повертає дані назад до рядкової форми, якщо вони спочатку були закодовані в коді utf-8.