Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №5

з дисципліни

«Криптографія»

на тему: «Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем»

Виконали:

студенти 3 курсу ФТІ

групи ФБ-72

Топорова Варвара та Лобанова Уляна

Перевірив: _____

Мета роботи :

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Порядок виконання роботи

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і p1, q1 довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p1q1$; p і q – прості числа для побудови ключів абонента A, p1 iq1 – абонента B.

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p,q) та відкритий ключ (n,e) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e, n) , (e1, n1) та секретні d і d1

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів A и B, перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання.

Результати роботи:

Повідмлення:
289899100659808929238201051408189539524253803255273927234416215083659113160290075285818158638531451233369913897462496523966307361559095626804994074516362982560507552664976283471623786

Відкриті ключі:

- 97813500829844317413620092442920484821935751483349846152018415627096756981530668736025332847384659725478096638840833587167894823133676570841958696892401043682499335927873473074479639
- 65537

Сигнатури:

- 52155698726851034039166976749143395942714803672396200952853328780964014272812572043557685516010086525606748727746462947117529792949021647808831442500035649009951181222970321979870573142830041174312844762107024677434906795741183493414174416605
- 68096753887688657398956665548545528617139684845161670281032328033609435785191730439708022227461590743362981562054580248312436939742052651406304082598947665536419049985207334346086548764458231797356761107514639414493655431028380772989312757016

Receive key

Clear

Key: c4c170dc51a937d4aae07be195a0680a223629d73f3f330f8fb2f03929e926de5cc0a251c770aac42cad882fc45880a

Signature: 690cba7ddd23d905338f2b746354b09e039f1ca159a4bbc4ec14f0444f700e613f12102394872bee69ddc1506b2656

Modulus: 1792804742ea8d87e8a3b6c963b4d4770444f1a5fbcded25d01de7c661f49027124cd78777c110b24d4989e76750f3

Public exponent: 10001

Receive

Key: 06

Verification: true

Код програми

```
const BigNumber = require('bignumber.js');

var rl = require('readline-sync');

const getRandom = (min, max) =>

  BigNumber.random().multipliedBy(max.minus(min)).integerValue(BigNumber.ROUND_FLOOR).plus(min);


const euclideanAlgorithm = (

  a, b, x1 = new BigNumber(1), y1 = new BigNumber(0), x2 = new BigNumber(0), y2 = new BigNumber(1)
```

```
) => {
    if (b == 0) {
        return y1;
    }
    return euclideanAlgorithm(
        b,
        a.modulo(b),
        x2,
        y2,
        x1.minus(a.dividedBy(b).integerValue(BigNumber.ROUND_FLOOR).multipliedBy(x2)),
        y1.minus(a.dividedBy(b).integerValue(BigNumber.ROUND_FLOOR).multipliedBy(y2))
    );
}

const checkPrime = (p) => {
    const primeNumbers = [ new BigNumber(2), new BigNumber(3), new BigNumber(5),
        new BigNumber(7), new BigNumber(11), new BigNumber(13), new BigNumber(17) ];
    for (let i of primeNumbers) {
        if (p.modulo(i) == 0) return false;
    }
    return true;
}
const getGcd = (a, b) => {
    if (b == 0) return a;
    return getGcd(b, a.modulo(b));
}
const hornersMethod = (x, a, m) => {
    const arr = [];
    let temp = a;
    while (temp != 1) {
        if (temp.modulo(2) == 0) arr.unshift(new BigNumber(0));
        else arr.unshift(1);
        temp = temp.dividedBy(2).integerValue(BigNumber.ROUND_DOWN);
    }
    arr.unshift(new BigNumber(1));
```

```
    const result = arr.reduce((prev, curr, i) => {
        if (i !== arr.length - 1) return prev.multipliedBy(x.pow(curr)).modulo(m).pow(2);
        else return prev.multipliedBy(x.pow(curr)).modulo(m);
    }, new BigNumber(1));
    return result;
}
const millerRabin = (p, k, currK = 1) => {
    if (k === currK) return true;
    let d = p.minus(1);
    let s = 0;
    while (d.modulo(2) == 0) {
        d = d.dividedBy(2);
        s++;
    }
    const x = getRandom(new BigNumber(1), p);
    if (getGcd(p, x) == 1) {
        let result = false;
        if (hornersMethod(x, d, p) != 1 && hornersMethod(x, d, p).minus(p) != -1) {
            let xr;
            for (let r = 1; r < s; r++) {
                if (r === 1) xr = hornersMethod(x, d.multipliedBy(2), p);
                else xr = xr.pow(2).modulo(p);
                if (xr.minus(p) == -1) {
                    result = true;
                    break;
                }
                else if (xr == 1) {
                    result = false;
                    break;
                }
            }
        } else result = true;
        if (result === true) return millerRabin(p, k, currK + 1);
        else return false;
    } else return false;
```

```
}
const getPrime = (n0, n1, k) => {
    const x = getRandom(n0, n1);
    if (x.modulo(2) == 0) m0 = x.plus(1);
    else m0 = x;
    let i = 0, temp = m0;
    while (i != n1.minus(m0).dividedBy(2).integerValue(BigNumber.ROUND_DOWN).plus(1)) {
        if (checkPrime(temp) && millerRabin(temp, k)) return temp;
        i += 1;
        temp = m0.plus(i * 2);
    }
    return false;
}
const getPrimeWithUserRange = () => {
    const n0 = new BigNumber(rl.question('Enter first number: '));
    const n1 = new BigNumber(rl.question('Enter second number: '));
    const k = Number(rl.question('Enter k: '));
    return [n0, n1, k];
}
const generatePrimePairs = (n0, n1, k) => {
    return [
        {
            p: getPrime(...getPrimeWithUserRange()),
            q: getPrime(...getPrimeWithUserRange())
        },
        {
            p: getPrime(...getPrimeWithUserRange()),
            q: getPrime(...getPrimeWithUserRange())
        }
    ]
}
const generateKeyPair = ({ p, q }) => {
    const n = p.multipliedBy(q);
    const euler = p.minus(1).multipliedBy(q.minus(1));
    const e = new BigNumber(2).pow(16).plus(1);
```

```
    const d = euclideanAlgorithm(euler, e);

    return {
        closed: { d, p, q },
        opened: { n, e }
    }
}
const encrypt = (m, e, n) => hornersMethod(m, e, n);

const decrypt = (c, d, n) => hornersMethod(c, d, n);

const sign = (m, d, n) => {
    const s = hornersMethod(m, d, n);
    return { m, s };
}

const verify = (s, e, n) => hornersMethod(s, e, n);

const sendKey = (k, e1, d, n, n1) => {
    const { s } = sign(k, d, n);
    const s1 = encrypt(s, e1, n1);
    const k1 = encrypt(k, e1, n1);
return { s1, k1 };
}

const receiveKey = (d1, k1, s1, n1, e, n) => {
    const k = decrypt(k1, d1, n1);
    const s = decrypt(s1, d1, n1);
    const verification = verify(s, e, n);
    console.log("K: ", verification.toFixed(), "M: ", k.toFixed());
    return { verification, k };
}
const [first, second] = generatePrimePairs();
const firstKeys = generateKeyPair(first);
const secondKeys = generateKeyPair(second);
```

```
const message = getRandom(new BigNumber(0), new BigNumber(firstKeys.opened.n));
const mess = sendKey(message, secondKeys.opened.e, firstKeys.closed.d, firstKeys.opened.n, secondKeys.opened.n);
receiveKey(secondKeys.closed.d, mess.k1, mess.s1, secondKeys.opened.n, firstKeys.opened.e, firstKeys.opened.n);
```

Висновок: у ході комп’ютерного практикуму було набуто навичок роботи з числами великої розрядності, написання тестів перевірки чисел на простоту та методів генерації ключів для асиметричної криптосистеми RSA. Набуто навичок побудови цифрового підпису на основі криптосистеми RSA.