**Лабораторна робота №5**

Генерація великих простих чисел. Піднесення великого простого числа в цілу степінь по модулю

Мета: отримати представлення та реалізувати програмно метод генерації великих простих чисел та перевірки великих чисел на простоту, отримати представлення та програмно реалізувати алгоритм піднесення великого простого числа до цілого степення по модулю.

Завдання: cтворити програму, що генеруeє прості числа з перевіркою Міллера-Рабина. Реалізувати алгоритм піднесення великого простого числа *a* до цілого степення *b* по модулю *t*.

Хід роботи

1. Текст програми

expMode.ts

export default function expMod(base: number, exp: number, mod: number) {

   if (exp === 1) return base % mod;

   const bs = exp.toString(2).split('').map(b => +b);

   const mods = new Array(bs.length);

   mods[0] = base;

   bs.forEach((b, i) => {

      if (i === 0) return;

      const t = mods[i - 1] \*\* 2 % mod;

      mods[i] = (b === 1) ? (t \* base) % mod : t;

   });

   return mods[mods.length - 1];

}

genPrime.ts

import smallPrimes from "./smallPrimes";

import rabinMillersCheck from "./rabinMillersCheck";

export default function genPrime(size = 16) {

   let strN = new Array(size - 2).fill('')

      .map(\_ => (Math.random() > 0.5) ? '1' : '0').join('');

   let n = parseInt(`1${strN}1`, 2);

   while (!check(n)) {

      n += 2;

   }

   return n;

}

function check(n: number): boolean {

   return smallPrimes.some(p => n % p === 0) &&

      rabinMillersCheck(n);

}

rabinMillerCheck.ts

import expMod from "./expMod";

/\*\*

 \* Miller-Rabin Primality Test

 \*

 \* @returns true if n is \*\*probably\*\* prime and

 \*    false if n is definitely not prime

 \*/

export default function rabinMillersCheck(n: number): boolean {

   n = Math.abs(n);

   if (n < 2) return false;

   if (n == 2) return true;

   const a = 2;

   let k = 1;

   while ((n - 1) % 2 \*\* k === 0) {

      k++;

   }

   const m = (n - 1) / 2 \*\* --k;

   let b = expMod(a, m, n);

   if (b === 1 || b === n - 1) {

      return true;

   }

   for (let i = 0; i < k; i++) {

      b = (b \* b) % n;

      if (b === 1) return false;

      if (b === n - 1) return true;

   }

   return false;

}

smallPrimes.ts

export default [

   2, 3, 5, 7, 11, 13, 17, 19, 23, 29,

   . . .

   1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987,

   1993, 1997, 1999,

];

Index.ts

import 'normalize.scss/normalize.scss';

import './index.scss';

import gen from './scripts/genPrime';

import mod from './scripts/expMod';

import rabinMillersCheck from './scripts/rabinMillersCheck';

const global = <any>window;

printHello();

global.genPrime = gen;

global.checkPrime = rabinMillersCheck;

global.expMod = mod;

function printHello() {

   console.log('%cSoftware Security', 'color: red; font-size: 28px');

   console.log('%cPrime numbers generation & Exponential modulo', 'color: red; font-size: 20px');

   console.log(`%cAvailable commands:

      > genPrime(size) - generate random size-bits prime number

      > checkPrime(n) - check if n is prime (Miller-Rabin primality test)

      > expMod(base, exp, n) - find the exponential modulo (base^exp mod n)

   `.replace(/\n\s+/g, '\n'), 'font-size: 16px');

}

1. Результат виконання програми



**Висновок:** на цій лабораторній роботі я отримав представлення про типові схеми ідентифікації та аутентифікації, набув практичних навичок створення програм для ідентифікації та аутентифікації з використанням Playfair Cipher.