Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №6»**

“Исследование потоковых шифров”

**Выполнил:** студент 4 курса

1 группы специальности ИСИт

Палазник Арсений Викторович

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

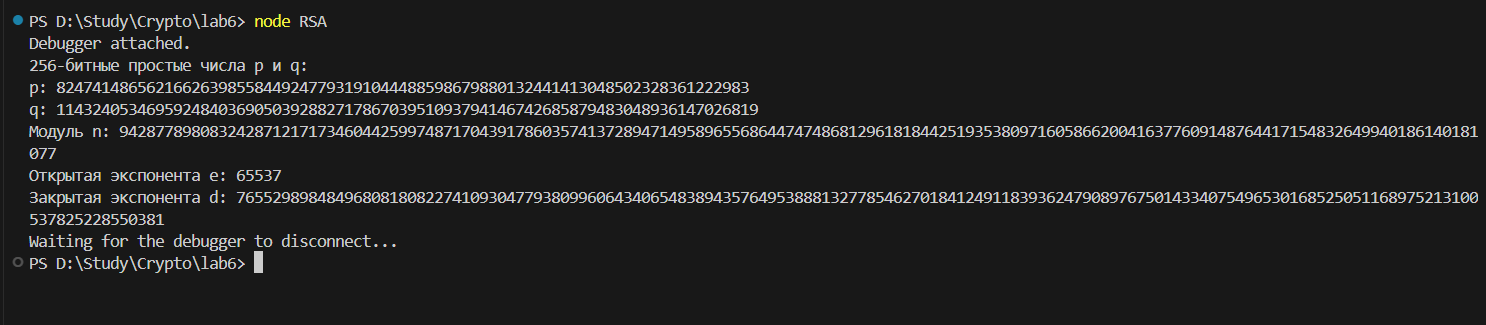
**Разработать авторские многооконные приложения в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими заданные алгоритмы.**

**Приложение 1 должно реализовывать генерацию ПСП в соответствии с вариантом из табл. 6.7.**

Реализация приложения:

|  |
| --- |
| // Приложение 1: RSA Key Generation  const bigintCryptoUtils = require("bigint-crypto-utils");  // Задаём значение для e, часто используемое значение - 65537  const e = BigInt(65537);  async function generateRSAKeys() {  // Генерация 256-битных простых чисел p и q  const p = await bigintCryptoUtils.prime(256);  const q = await bigintCryptoUtils.prime(256);  // Вычисляем n = p \* q  const n = p \* q;  // Вычисляем функцию Эйлера φ(n) = (p - 1) \* (q - 1)  const phi = (p - BigInt(1)) \* (q - BigInt(1));  // Проверка на то, что e и φ(n) - взаимно просты  if (bigintCryptoUtils.gcd(e, phi) !== BigInt(1)) {  throw new Error("e и φ(n) должны быть взаимно простыми");  }  // Вычисляем d - закрытую экспоненту, используя обратный элемент по модулю φ(n)  const d = bigintCryptoUtils.modInv(e, phi);  console.log("256-битные простые числа p и q:");  console.log("p:", p.toString());  console.log("q:", q.toString());  console.log("Модуль n:", n.toString());  console.log("Открытая экспонента e:", e.toString());  console.log("Закрытая экспонента d:", d.toString());  }  generateRSAKeys().catch(console.error); |

Результат выполнения:



**Приложение 2 должно реализовывать алгоритм RC4 в соответствии с вариантом из табл. 6.8, а также дополнительно выполнять оценку скорости выполнения операций генерации ПСП.**

Реализация приложения:

|  |
| --- |
| const CryptoJS = require("crypto-js");  // Новый 6-байтовый ключ  const keyArray = [61, 60, 23, 22, 21, 20];  // Преобразуем массив ключа в строку  const keyStr = keyArray.map((num) => String.fromCharCode(num)).join("");  // Произвольный текст для шифрования  const message = "Пример текста для шифрования";  // Функция для измерения времени выполнения  function measureExecutionTime(fn) {  const start = process.hrtime.bigint();  fn();  const end = process.hrtime.bigint();  return Number(end - start) / 1e6; // Время в миллисекундах  }  // Функция для RC4 шифрования и дешифрования  function encryptDecryptRC4(message, keyStr) {  const encrypted = CryptoJS.RC4.encrypt(message, keyStr);  const decrypted = CryptoJS.RC4.decrypt(encrypted, keyStr).toString(  CryptoJS.enc.Utf8  );  console.log("Зашифрованное сообщение:", encrypted.toString());  console.log("Расшифрованное сообщение:", decrypted);  }  // Оценка скорости генерации ПСП (примерная симуляция)  function randomPrimeGenerationTime() {  measureExecutionTime(() => {  for (let i = 0; i < 100000; i++) {  Math.random(); // Симулируем генерацию случайных чисел  }  });  }  // Выполняем операции  console.log("Шифрование RC4 и дешифрование:");  encryptDecryptRC4(message, keyStr);  console.log("Оценка времени генерации ПСП:");  console.log(  "Время генерации:",  measureExecutionTime(randomPrimeGenerationTime),  "мс"  ); |

Результат выполнения:

