МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

**«**ИССЛЕДОВАНИЕ ПОТОКОВЫХ ШИФРОВ**»**

Студент: Бутурля Р.А.

ФИТ 3 курс 5 группа

Вариант 3

Преподаватель: Савельева М. Г.

Минск 2023

1. Разработать авторские многооконные приложения в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими заданные алгоритмы.
2. Приложение 1 должно реализовывать генерацию ПСП(BBS, 256)

Для выполнения генерации ПСП был разработан метод doBBS(p, q, iter). Параметры *p* и *q* – простые числа, сравнимые с числом 3 по модулю 4, параметр *iter* – количество символов, которые должен вернуть нам генератор.

|  |
| --- |
| function doBBS(p,q, iter){      if(p%4!=3 || q%4!=3)throw new Error('wrong p or q');      let n = p\*q;      console.log(factors(n));      let x = findX(factors(n));      let resArr = [], resArrBin = [];      for(let i = 0; i < iter; i++){          resArr[i] = Math.pow((i == 0 ? x : resArr[i - 1]), 2) % n;          resArrBin[i] = resArr[i] % 2;      }      return [resArr, resArrBin];  } |

Листинг 1 – Метод doBBS(p, q, iter)

Метод doBBS(p, q, iter) вызывает во время выполнения метод factors(n). Метод factors(n) расклалывает поступившее в качестве параметра n число на простые множители и возвращает их.

|  |
| --- |
| function factors(a, arr = [], b = 2) {      if (b > Math.sqrt(a)) {        arr.push(a);        return arr;      } else if (a % b == 0) {        arr.push(b);        factors(a / b, arr, b);      } else {        factors(a, arr, ++b);      }      return arr;  } |

Листинг 2 – Метод factors(n)

Также метод doBBS(p, q, iter) вызывает метод findX(nFactor). Данный метод определяет некоторое значение *x*, которое должно быть взаимно простым с числом *n*. В параметрe *nFactor* хранится разложение *n* на простые числа и на основе этого параметра находится *x*.

|  |
| --- |
| function findX(nFactor){      let isRightX = false, a = true;      let x = 1, xFactor;      while(!isRightX){          x++;          xFactor = factors(x);          console.log(xFactor);          for(let i = 0; i < xFactor.length; i++){              for(let j = 0; j < nFactor.length; j++){                  if(xFactor[i] == nFactor[j]){                      a=false;                      break;                  }              }              if(!a)break;          }          if(a){              return x;          } else {a = true};      }  } |

Листинг 3 – findX(nFactor)

1. Приложение 2 должно реализовывать алгоритм RC4

Для реализации алгоритма RC4 был разработан метод doRC4(word, key). Параметр *word* – шифруемое слово, *key* – ключ, состоящий из некоторого количества чисел.

|  |
| --- |
| function doRC4(word,key){      let start = performance.now();      let sTable = getStartedStable();      let kTable = createKTable(key);      sTable = doPermutationSTable(sTable, kTable);      let resultKey = getResultKey(sTable, key.length);      let result = cipher(word, resultKey);      let end = performance.now();      return {result:result, time:(end-start)};  } |

Листинг 4 – Метод doRC4(word, key)

Метод doRC4(word, key) поочередно вызывает следующие методы:

* getStartedStable() – генерация стартовой таблицы S;
* createKTable(key) – генерирует таблицу K, записывая в нее последовательность ключей необходимое количество раз;
* doPermutationSTable(sTable, kTable) и getResultKey(sTable, length) – на основе таблиц, полученных из методов пункта 1 и 2, методы подготавливают таблицу S к основному этапу и выполняют его;
* cipher(word, key) – метод выполняет конечное преобразования слова с помощью битовой операции.

|  |
| --- |
| function getStartedStable(){      let arr = [];      for(let i = 0; i < n\*n; i++){          arr[i] = i;      }      return arr;  } |

Листинг 5 – Метод getStartedStable()

|  |
| --- |
| function createKTable(key){      let k = 0, keyLength = key.length;      let arr = [];      for(let i = 0; i < n\*n; i++){          if(i >= keyLength\*(k+1))k++          arr[i] = +key[i - k \* keyLength]      }      return arr;  } |

Листинг 6 – Метод createKTable(key)

|  |
| --- |
| function doPermutationSTable(sTable, kTable){      let sPTable = sTable;      let kPTable = kTable;      let x;      let i = 0, j = 0;      while(i < n\*n){          j = (j + sPTable[i] + kPTable[i])%(n\*n);          x = sPTable[i];          sPTable[i] = sPTable[j];          sPTable[j] = x;          i++;      }      return sPTable;  } |

Листинг 7 – Метод doPermutationSTable(sTable, kTable)

|  |
| --- |
| function getResultKey(sTable, length){      let sRTable = sTable;      let k = 0, x, a;      let i = 0; j = 0;      let arr = []      while(k < length){          i = (++i)%(n\*n);          j = (j + sRTable[i])%(n\*n);          x = sRTable[i];          sRTable[i] = sRTable[j];          sRTable[j] = x;          a = (sRTable[i] + sRTable[j])%(n\*n);          arr[k] = sRTable[a];          k++;      }      return arr;  } |

Листинг 8 – Метод getResultKey(sTable, length)

|  |
| --- |
| function cipher(word, pKey){      let numWord = [];      for(let item in word){          numWord[item] = ConvertToNum(word[item]);      }      let key = [];      let k = 0, keyLength = pKey.length;      for(let i = 0; i < word.length; i++){          if(i >= keyLength\*(k+1))k++          key[i] = pKey[i - k \* keyLength]      }      let result = [];      for(let i = 0; i < key.length; i++){          result[i] = numWord[i]^key[i];      }      for(let item in result){          result[item] = String.fromCharCode(result[item]);      }      return result;  } |

Листинг 9 – Метод cipher(word, pKey)

Для написания оконного приложения был выбран JavaScript(Node.js).

|  |
| --- |
| const express = require('express');  const {doBBS} = require('./bbs');  const {doRC4} = require('./rc4');  const fs = require('fs');  const app = express();  app.use(express.json());  app.get('/', (req, res)=>{      res.end(fs.readFileSync('first.html'));  });  app.post('/',(req, res)=>{      console.log(req.body);      let {p,q} = req.body;      console.log(p,q);      let arr = doBBS(+p,+q,10);      res.end(JSON.stringify(arr));  });  app.get('/rc4', (req, res)=>{      res.end(fs.readFileSync('second.html'));  });  app.post('/rc4', (req, res)=>{      let {word,key} = req.body;      key = key.split(' ');      console.log(word, key);      let obj = doRC4(word,key);      res.end(JSON.stringify(obj));  })  app.listen(3000); |

Листинг 10 – Приложение для демонстрации работы потоковых шифров

Результаты выполнения приложения:

* BBS – входные данные: 11 и 19. Результат выполнения представлен на рисунке 1.
* RC4 – входные данные: «roman» и «1 2 3». Результат выполнения представлен на рисунке 1.

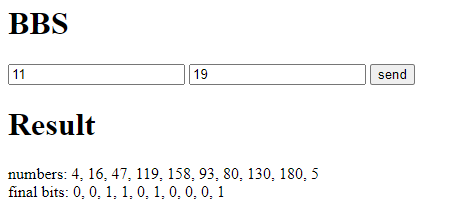


Рисунок 1 – Результат работы BBS

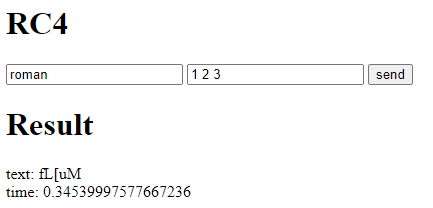


Рисунок 2 – Результат работы RC4

**Вывод**

Применил практические навыки разработки и использования приложений для реализации потоковых шифров.