Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №7»**

“Исследование ассиметричных шифров”

**Выполнил:** студент 4 курса

1 группы специальности ИСИт

Палазник Арсений Викторович

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

**Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами.**

**В основе вычислений – кодировочные таблицы Base64 и ASCII. Приложение должно реализовывать следующие операции:**

* **генерация сверхвозрастающей последовательности (тайного ключа); старший член последовательности – 100-битное число; в простейшем случае принимается z = 6 (для кодировки Base64) и z = 8 (для кодировки ASCII);**
* **вычисление нормальной последовательности (открытого ключа);**
* **зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;**
* **расшифрование сообщения;**
* **оценка времени выполнения операций зашифрования и расшифрования.**

Реализация приложения:

|  |
| --- |
| const crypto = require("crypto");  const fs = require("fs");  const path = require("path");  //NODE\_OPTIONS=--openssl-legacy-provider node index.js  // Функция для проверки и подготовки ключа  function getValidKey(key) {  const newKey = Buffer.alloc(8);  if (key.length < 8) {  Buffer.from(key).copy(newKey, 0);  for (let i = key.length; i < 8; i++) {  newKey[i] = key[i % key.length];  }  } else if (key.length > 8) {  newKey.set(Buffer.from(key.slice(0, 8)));  } else {  newKey.set(Buffer.from(key));  }  return newKey;  }  // Шифрование DES  function encryptDES(plainText, key) {  const validKey = getValidKey(key);  const cipher = crypto.createCipheriv("des-ecb", validKey, null);  cipher.setAutoPadding(true);  const encrypted = Buffer.concat([cipher.update(plainText), cipher.final()]);  return encrypted;  }  // Расшифрование DES  function decryptDES(cipherText, key) {  const validKey = getValidKey(key);  const decipher = crypto.createDecipheriv("des-ecb", validKey, null);  decipher.setAutoPadding(true);  const decrypted = Buffer.concat([  decipher.update(cipherText),  decipher.final(),  ]);  return decrypted;  }  // Подсчет изменения битов для эффекта лавины  function getAvalancheEffect(initialOpenText, encryptedText) {  let changedBits = 0;  for (let i = 0; i < initialOpenText.length; i++) {  const originalByte = initialOpenText[i];  const encryptedByte = encryptedText[i];  let xor = originalByte ^ encryptedByte;  while (xor !== 0) {  if ((xor & 1) === 1) {  changedBits++;  }  xor >>= 1;  }  }  return changedBits;  }  // Главная функция  (async () => {  try {  const key = Buffer.from("palaznik", "utf8"); // Ключ  const plainTextPath = path.resolve(\_\_dirname, "text.txt");  const encryptedPath = path.resolve(\_\_dirname, "Encrypt.txt");  const decryptedPath = path.resolve(\_\_dirname, "Decrypt.txt");  const plainText = fs.readFileSync(plainTextPath);  console.time("Encrypt DES");  const encryptedText = encryptDES(plainText, key);  console.timeEnd("Encrypt DES");  fs.writeFileSync(encryptedPath, encryptedText);  console.time("Decrypt DES");  const decryptedText = decryptDES(encryptedText, key);  console.timeEnd("Decrypt DES");  fs.writeFileSync(decryptedPath, decryptedText);  const totalBits = plainText.length \* 8;  const changedBits = getAvalancheEffect(plainText, encryptedText);  console.log();  console.log(`Total bits count:\t${totalBits} bits`);  console.log(`Avalanche Effect:\t${changedBits} bits (changed)`);  console.log(  `Percentage ratio:\t${((changedBits / totalBits) \* 100).toFixed(2)}%`  );  } catch (error) {  console.error("Error:", error.message);  }  })(); |

Результат выполнения:

