Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №5»**

“Исследование блочных шифров”

**Выполнил:** студент 4 курса

1 группы специальности ИСИт

Палазник Арсений Викторович

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

**Разработать авторское приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы. Приложение должно реализовывать следующие операции:**

* **разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;**
* **выполнение требуемых преобразований ключевой информации;**
* **выполнение операций зашифрования/расшифрования;**
* **оценка скорости выполнения операций зашифрования/расшифрования;**
* **пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову**

Реализация приложения:

|  |
| --- |
| // DES encryption and decryption using CryptoJS  const key = CryptoJS.enc.Utf8.parse('Vakulenc'); // Fixed key (8 characters for DES)  const blockSize = 8; // DES block size is 8 bytes  // Function to pad the input text to a multiple of 8 bytes  function padInput(input) {  const paddingSize = blockSize - (input.length % blockSize);  const padding = new Array(paddingSize).fill(String.fromCharCode(paddingSize)).join('');  return input + padding;  }  // Function to split input text into blocks  function splitIntoBlocks(input) {  const blocks = [];  for (let i = 0; i < input.length; i += blockSize) {  blocks.push(input.slice(i, i + blockSize));  }  return blocks;  }  // Function to measure the encryption speed  function measureSpeed(fn) {  const startTime = performance.now();  fn();  const endTime = performance.now();  return (endTime - startTime).toFixed(4) + ' ms';  }  // Function to calculate the avalanche effect  function calculateAvalancheEffect(originalText, encryptedText) {  let changedBits = 0;  for (let i = 0; i < originalText.length; i++) {  const originalChar = originalText.charCodeAt(i);  const encryptedChar = encryptedText.charCodeAt(i);  changedBits += (originalChar ^ encryptedChar).toString(2).split('1').length - 1;  }  return changedBits;  }  // Encrypt function  function encryptText() {  const inputText = document.getElementById('inputText').value;  const paddedText = padInput(inputText);  // Encrypting using DES  const encrypted = CryptoJS.DES.encrypt(CryptoJS.enc.Utf8.parse(paddedText), key, {  mode: CryptoJS.mode.ECB, // DES mode (ECB in this case)  padding: CryptoJS.pad.Pkcs7 // Padding scheme  }).toString();    document.getElementById('result').textContent = encrypted;    return encrypted;  }  // Decrypt function  function decryptText() {  const inputText = document.getElementById('result').textContent;  // Decrypting using DES  const decrypted = CryptoJS.DES.decrypt(inputText, key, {  mode: CryptoJS.mode.ECB,  padding: CryptoJS.pad.Pkcs7  }).toString(CryptoJS.enc.Utf8);  document.getElementById('result').textContent = decrypted;  }  // Bind events to buttons  document.getElementById('encryptBtn').addEventListener('click', function () {  const originalText = document.getElementById('inputText').value;  // Measure encryption speed  const encryptionSpeed = measureSpeed(() => encryptText());  document.getElementById('speed').textContent = `Encryption speed: ${encryptionSpeed}`;  // Calculate avalanche effect  const encryptedText = encryptText();  const avalancheEffect = calculateAvalancheEffect(originalText, encryptedText);  document.getElementById('avalancheEffect').textContent = `Avalanche Effect: ${avalancheEffect} bits changed.`;  });  document.getElementById('decryptBtn').addEventListener('click', function () {  decryptText();  }); |

Результат выполнения:



