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

Халалеенко Андрей Николаевич

**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:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Encryptor</title>      <style>          body {              font-family: Arial, sans-serif;              display: flex;              justify-content: center;              align-items: center;              height: 100vh;              margin: 0;              background-color: #f0f0f0;          }          .container {              max-width: 400px;              width: 100%;              padding: 20px;              border: 1px solid #ccc;              border-radius: 10px;              box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);              background-color: #fff;          }          input, button {              width: 100%;              padding: 10px;              margin: 5px 0;              font-size: 16px;          }          button {              cursor: pointer;          }          .key-display {              margin-top: 10px;              padding: 10px;              background-color: #e9e9e9;              border-radius: 5px;              font-size: 14px;              word-wrap: break-word;          }      </style>  </head>  <body>      <div class="container">          <h2>Encryptor</h2>          <input type="text" id="message" placeholder="Enter message (e.g., Name Surname)">          <input type="text" id="keyLength" placeholder="Enter key length (e.g., 6 for Base64)">          <button onclick="generateKey()">Generate Key</button>          <div class="key-display" id="keyDisplay">Keys will be displayed here...</div>          <button onclick="encryptMessage()">Encrypt</button>          <button onclick="decryptMessage()">Decrypt</button>          <input type="text" id="encryptedMessage" placeholder="Encrypted message" readonly>          <input type="text" id="decryptedMessage" placeholder="Decrypted message" readonly>          <p id="timeElapsed"></p>      </div>      <script>          let privateKey = [];          let publicKey = [];          function generateKey() {              const keyLength = parseInt(document.getElementById("keyLength").value);              if (isNaN(keyLength) || keyLength <= 0) {                  alert("Invalid key length");                  return;              }              privateKey = generateSuperincreasingSequence(keyLength);              publicKey = generateNormalSequence(privateKey);  *// Отображаем ключи в отдельном блоке*              document.getElementById("keyDisplay").innerText =                  `Private Key: ${privateKey.join(", ")}\nPublic Key: ${publicKey.join(", ")}`;          }          function generateSuperincreasingSequence(length) {              const sequence = [1];              for (let i = 1; i < length; i++) {                  sequence.push(sequence.reduce((a, b) => a + b) + Math.floor(Math.random() \* 10) + 1);              }              return sequence;          }          function generateNormalSequence(privateKey) {              return privateKey.map(num => num \* 2);          }          function encryptMessage() {              const message = document.getElementById("message").value;              if (!message) {                  alert("Please enter a message");                  return;              }              const startTime = performance.now();              const base64Message = btoa(message); *// Encode message in Base64*  *// Добавляем задержку для демонстрации времени выполнения*              setTimeout(() => {                  const encrypted = base64Message.split("").map((char, index) => {                      return String.fromCharCode(char.charCodeAt(0) + publicKey[index % publicKey.length]);                  }).join("");                  const endTime = performance.now();                  document.getElementById("encryptedMessage").value = encrypted;                  document.getElementById("timeElapsed").innerText = `Encryption time: ${(endTime - startTime).toFixed(2)} ms`;              }, 50); *// Задержка в 50 мс*          }          function decryptMessage() {              const encrypted = document.getElementById("encryptedMessage").value;              if (!encrypted) {                  alert("Please encrypt a message first");                  return;              }              const startTime = performance.now();  *// Добавляем задержку для демонстрации времени выполнения*              setTimeout(() => {                  const decryptedBase64 = encrypted.split("").map((char, index) => {                      return String.fromCharCode(char.charCodeAt(0) - publicKey[index % publicKey.length]);                  }).join("");                  const decrypted = atob(decryptedBase64); *// Decode from Base64*                  const endTime = performance.now();                  document.getElementById("decryptedMessage").value = decrypted;                  document.getElementById("timeElapsed").innerText = `Decryption time: ${(endTime - startTime).toFixed(2)} ms`;              }, 50); *// Задержка в 50 мс*          }      </script>  </body>  </html> |

Результат выполнения:

