Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №8**

“Исследование асимметричных шифров RSA и Эль-Гамаля”

**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:**

**• зашифрование и расшифрование текстовых документов**

**на основе алгоритмов RSA и Эль-Гамаля;**

**• определение времени выполнения операций.**

Реализация приложения:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>RSA и ElGamal Шифрование</title>  </head>  <body>  <h2>Шифрование сообщения (RSA и ElGamal)</h2>  <label>Сообщение:</label>  <input type="text" id="messageInput" placeholder="Введите сообщение"><br><br>  <label>Публичный ключ (RSA):</label>  <textarea id="publicKeyDisplay" readonly></textarea><br><br>  <label>Публичный ключ (ElGamal):</label>  <textarea id="gamalPublicKeyDisplay" readonly></textarea><br><br>  <button onclick="generateKeys()">Сгенерировать ключи</button>  <button onclick="encryptMessage()">Зашифровать</button>  <button onclick="decryptMessage()">Расшифровать</button><br><br>  <label>Зашифрованное сообщение (RSA):</label>  <textarea id="encryptedMessageDisplay" readonly></textarea><br><br>  <label>Расшифрованное сообщение (RSA):</label>  <textarea id="decryptedMessageDisplay" readonly></textarea><br><br>  <label>Зашифрованное сообщение (ElGamal):</label>  <textarea id="gamalEncryptedMessageDisplay" readonly></textarea><br><br>  <label>Расшифрованное сообщение (ElGamal):</label>  <textarea id="gamalDecryptedMessageDisplay" readonly></textarea><br><br>  <script>  let rsaKeys;  let elGamalKeys;  async function generateKeys() {  // Генерация RSA ключей  rsaKeys = await window.crypto.subtle.generateKey(  {  name: "RSA-OAEP",  modulusLength: 2048,  publicExponent: new Uint8Array([1, 0, 1]),  hash: "SHA-256"  },  true,  ["encrypt", "decrypt"]  );  // Экспорт RSA публичного ключа  const exportedPublicKey = await window.crypto.subtle.exportKey("spki", rsaKeys.publicKey);  document.getElementById("publicKeyDisplay").value = btoa(String.fromCharCode(...new Uint8Array(exportedPublicKey)));  // Генерация ключей для ElGamal  elGamalKeys = generateElGamalKeys();  document.getElementById("gamalPublicKeyDisplay").value = `Public Key (p, g, y): (${elGamalKeys.p}, ${elGamalKeys.g}, ${elGamalKeys.y})`;  }  function generateElGamalKeys() {  const p = generateLargePrime();  let g;  do {  g = randomBigInt(2n, p - 1n);  } while (modExp(g, (p - 1n) / 2n, p) === 1n);  const x = randomBigInt(1n, p - 2n);  const y = modExp(g, x, p);  return { p, g, y, x };  }  function generateLargePrime() {  return 104729n; // Простое число для демонстрации  }  function randomBigInt(min, max) {  return BigInt(Math.floor(Math.random() \* Number(max - min)) + Number(min));  }  function modExp(base, exp, mod) {  let result = 1n;  base = base % mod;  while (exp > 0n) {  if (exp % 2n === 1n) result = (result \* base) % mod;  exp = exp >> 1n;  base = (base \* base) % mod;  }  return result;  }  async function encryptMessage() {  const message = document.getElementById("messageInput").value;  const encoder = new TextEncoder();  const data = encoder.encode(message);  // Шифрование с помощью RSA публичного ключа  const encrypted = await window.crypto.subtle.encrypt(  { name: "RSA-OAEP" },  rsaKeys.publicKey,  data  );  document.getElementById("encryptedMessageDisplay").value = btoa(String.fromCharCode(...new Uint8Array(encrypted)));  // Шифрование с помощью ElGamal  const elGamalEncrypted = encryptElGamal(message, elGamalKeys);  document.getElementById("gamalEncryptedMessageDisplay").value = JSON.stringify(elGamalEncrypted);  }  function encryptElGamal(message, keys) {  const { p, g, y } = keys;  const encryptedChars = [];  for (let char of message) {  const m = BigInt(char.charCodeAt(0));  const k = randomBigInt(1n, p - 2n);  const a = modExp(g, k, p);  const b = (m \* modExp(y, k, p)) % p;  encryptedChars.push({ a: a.toString(), b: b.toString() });  }  return encryptedChars;  }  async function decryptMessage() {  const encryptedMessage = document.getElementById("encryptedMessageDisplay").value;  const encryptedData = Uint8Array.from(atob(encryptedMessage), c => c.charCodeAt(0)).buffer;  try {  // Дешифрование с помощью RSA приватного ключа  const decrypted = await window.crypto.subtle.decrypt(  { name: "RSA-OAEP" },  rsaKeys.privateKey,  encryptedData  );  const decoder = new TextDecoder();  const decryptedMessage = decoder.decode(decrypted);  document.getElementById("decryptedMessageDisplay").value = decryptedMessage;  } catch (err) {  alert("Ошибка при дешифровании RSA: " + err.message);  }  // Дешифрование с помощью ElGamal  const elGamalEncrypted = JSON.parse(document.getElementById("gamalEncryptedMessageDisplay").value);  const decryptedElGamal = decryptElGamal(elGamalEncrypted, elGamalKeys);  document.getElementById("gamalDecryptedMessageDisplay").value = decryptedElGamal;  }  function decryptElGamal(encrypted, keys) {  const { p, x } = keys;  let decryptedMessage = "";  for (let { a, b } of encrypted) {  const aBigInt = BigInt(a);  const bBigInt = BigInt(b);  const ax = modExp(aBigInt, x, p);  const axInv = modExp(ax, p - 2n, p);  const m = (bBigInt \* axInv) % p;  decryptedMessage += String.fromCharCode(Number(m));  }  return decryptedMessage;  }  </script>  </body>  </html> |

Результат выполнения:

