Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ

По лабораторной работе №9:

«Исследование асимметричных шифров»

Студент: Войцехович Е.А.

ФИТ 3 курс 10 группа

Преподаватель: Ржеутская Н.В.

Минск 2023

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров.

**Практическая часть**

Задание

Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться до- ступными библиотеками либо программными кодами.

Для выполнения задания использовалась библиотека merkle-hellman, подключенная в папку с лабораторной работой.

|  |
| --- |
| import {  Decoder,  Encoder,  } from './merkle-hellman/out/index.js';    const decoder = new Decoder();  const encoder = new Encoder(decoder.publicKey);    const message = 'Koss Valeria';    const encodedMessage = encoder.encode(message);  const decodedMessage = decoder.decode(encodedMessage);    console.log(`Source message: ${message}`);  console.log(`Encoded message: ${encodedMessage}`);  console.log(`Decoded message: ${decodedMessage}`); |

Листинг 1 — файл app.js

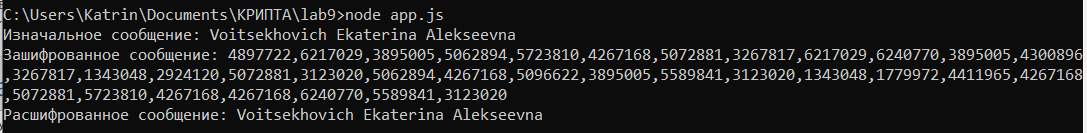


Рисунок 1 — Результат шифрования

**Цель:** в ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации асимметричных шифров.

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ

По лабораторной работе №10:

«Исследование асимметричных шифров RSA и Эль-Гамаля»

Студент: Войцехович Е.А.

ФИТ 3 курс 10 группа

Преподаватель: Ржеутская Н.В.

Минск 2023

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля.

**Практическая часть**

Задание

Разработать авторское оконное приложение для реализации асимметричных шифров RSA и Эль-Гамаля.

Для выполнения задания использовалась библиотека merkle-hellman, подключенная в папку с лабораторной работой.

|  |
| --- |
| function NOD(a, b) {  if (typeof a != 'number' || typeof b != 'number') throw new Error('Bad type of value');  let max = Math.max(a, b);  let min = Math.min(a, b);  let r = -1;  while (r != 0) {  r = max % min;  max = min;  min = r;  }  return max;  }  function GetNumberByModWithNumber(x, n, number) {  if (typeof x != 'number' || typeof n != 'number') throw new Error('Bad value type');  if (NOD(x, n) != 1) throw new Error('Числа должна быть взаимно простыми');  let y = 1,  k = 0;  while (true) {  k = (number - x \* y) / n;  if (Number.isInteger(k))  return y;  else  y++;  }  }  function GetNumberByMod(x, n) {  if (typeof x != 'number' || typeof n != 'number') throw new Error('Bad value type');  if (NOD(x, n) != 1) throw new Error('Числа должна быть взаимно простыми');  let y = 1,  k = 0;  while (true) {  k = (1 - x \* y) / n;  if (Number.isInteger(k))  return y;  else  y++;  }  }  function isPrimeNumber(number) {  for (let i = 2; i < number / 2; ++i)  if (number % i === 0)  return false;  return true;  }  function RSA({  n,  e,  d  }, message) {  let crypto = "";  const positionA = 'A'.charCodeAt(0);  for (let i = 0, {  length  } = message; i < length; ++i) {  const alphabetPosition = (message.charCodeAt(i) - positionA) + 1;  crypto += String.fromCharCode((Math.pow(alphabetPosition, e) % n) - 1 + positionA);  }  return crypto;  }  function DecryptRSA({  n,  e,  d  }, message) {  let crypto = "";  const positionA = 'A'.charCodeAt(0);  for (let i = 0, {  length  } = message; i < length; ++i) {  const alphabetPosition = (message.charCodeAt(i) - positionA) + 1;  crypto += String.fromCharCode((Math.pow(alphabetPosition, d) % n) - 1 + positionA);  }  return crypto;  }  function ElGamalInit() {  const [p, g, x] = [11, 2, 8];  const y = Math.pow(g, x) % p;  console.log ("y =",y);  return {  p,  g,  x,  y  }  }  function ElGamalSymbol({  p,  g,  x,  y  }, message) {  const k = Math.round(Math.random() \* (p - 1));  console.log("k =", k);  const a = Math.pow(g, k) % p;  const b = (Math.pow(y, k) \* message) % p;  return {  a,  b  };  }  function ElGamal(keys, message) {  const encrypt = [];  for (let i = 0, {  length  } = message; i < length; ++i) {  encrypt.push(ElGamalSymbol(keys, message[i]));  }  return encrypt;  }  function ElGamalDecrypt(keys, message) {  const decrypt = [];  for (let i = 0, {  length  } = message; i < length; ++i) {  decrypt.push(ElGamalSymbolDecrypt(keys, message[i]));  }  return decrypt;  }  function ElGamalSymbolDecrypt({  p,  g,  x,  y  }, {  a,  b  }) {  const m = b \* Math.pow(a, p - 1 - x) % p;  return m;  }  function RSAInit(randomPrimeNumber1,  randomPrimeNumber2) {  if (typeof randomPrimeNumber1 !== 'number' ||  typeof randomPrimeNumber2 !== 'number') throw new Error('Bad type of params');  if (!isPrimeNumber(randomPrimeNumber1) || !isPrimeNumber(randomPrimeNumber2))  throw new Error('Numbers are not prime');  const n = randomPrimeNumber1 \* randomPrimeNumber2;  const fn = (randomPrimeNumber1 - 1) \* (randomPrimeNumber2 - 1);  const e = 7;  const d = GetNumberByMod(e, fn);  return {  n,  e,  d  };  }  const keys = RSAInit(3, 11);  const encryptMessage = RSA(keys 'VOITSEKHOVICH, EKATERINA, ALEKSEEVNA'  );  console.log('RSA', encryptMessage);  console.log('RSA',DecryptRSA(keys, encryptMessage));  const keysElGamal = ElGamalInit();  const encryptElGamal = ElGamal(keysElGamal, [1, 7, 2, 0, 1]);  console.log('ElGamal',encryptElGamal);  console.log('ElGamal',ElGamalDecrypt(keysElGamal, encryptElGamal)); |

Листинг 1 — файл app.js

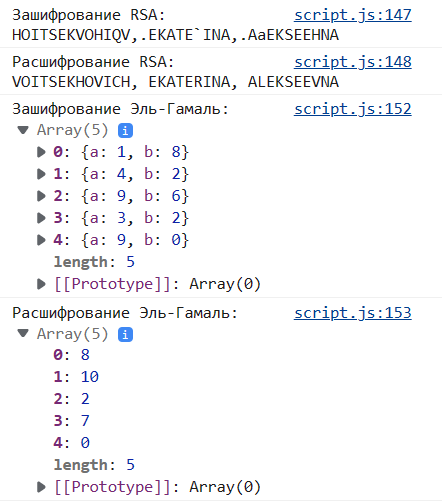


Рисунок 1 — Результат шифрования

**Вывод:** в ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля.

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ

По лабораторной работе №11:

«Исследование криптографических хэш-функций»

Студент: Войцехович Е.А.

ФИТ 3 курс 10 группа

Преподаватель: Ржеутская Н.В.

Минск 2023

**Цель**: изучение и приобретение практических навыков разработки и использования приложений для реализации криптиграфических хэш-функций.

**Практическая часть**

Задание

Разработать оконное приложение, реализующее алгоритм хеширования SHA-256. При этом можно воспользоваться доступными готовыми библиотеками. Язык программирования – на свой выбор.

Для выполнения задания использовалась библиотека crypto-js, подключенная в папку с лабораторной работой.

|  |
| --- |
| var CryptoJS = require("crypto-js")  const message = 'myMessageText'  //поддерживаются другие алгоритмы хеширования, такие как SHA-1, SHA-512, MD5  const enc = CryptoJS.SHA256(message).toString()  console.log("Зашифрование на SHA-256:");  console.log(enc); |

Листинг 1 — файл app.js

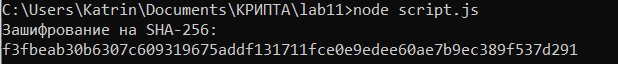


Рисунок 1 — Результат шифрования

**Вывод:** в ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложения для реализации SHA-512.

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ

По лабораторной работе №12

**«Исследование алгоритмов генерации и верификации электронной цифровой подписи»**

Вариант 2

Студент: Войцехович Е.А.

ФИТ 3 курс 10 группа

Преподаватель: Ржеутская Н.В.

Минск 2023

**Цель:** изучение алгоритмов генерации и верификации электронной цифровой подписи и приобретение практических навыков их реализации.

**Практическая часть**

Задание

1. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться результатами выполнения предыдущих лабораторных работ, а также доступными библиотеками либо программными кодами.

|  |
| --- |
| const crypto = require("crypto")  const { publicKey, privateKey } = crypto.generateKeyPairSync("rsa", {  // The standard secure default length for RSA keys is 2048 bits  modulusLength: 2048,  })  console.log(  publicKey.export({  type: "pkcs1",  format: "pem",  }),  privateKey.export({  type: "pkcs1",  format: "pem",  })  )  const data = "Voitsekhovich Ekaterina Alekseevna"  console.log("initial message: ", data)  const encryptedData = crypto.publicEncrypt(  {  key: publicKey,  padding: crypto.constants.RSA\_PKCS1\_OAEP\_PADDING,  oaepHash: "sha256",  },  // We convert the data string to a buffer using `Buffer.from`  Buffer.from(data)  )  console.log("encypted data: ", encryptedData.toString("base64"))  const decryptedData = crypto.privateDecrypt(  {  key: privateKey,  padding: crypto.constants.RSA\_PKCS1\_OAEP\_PADDING,  oaepHash: "sha256",  },  encryptedData  )  console.log("decrypted data: ", decryptedData.toString())  const verifiableData = "this need to be verified" |

Листинг 1 — Файл rsa.js

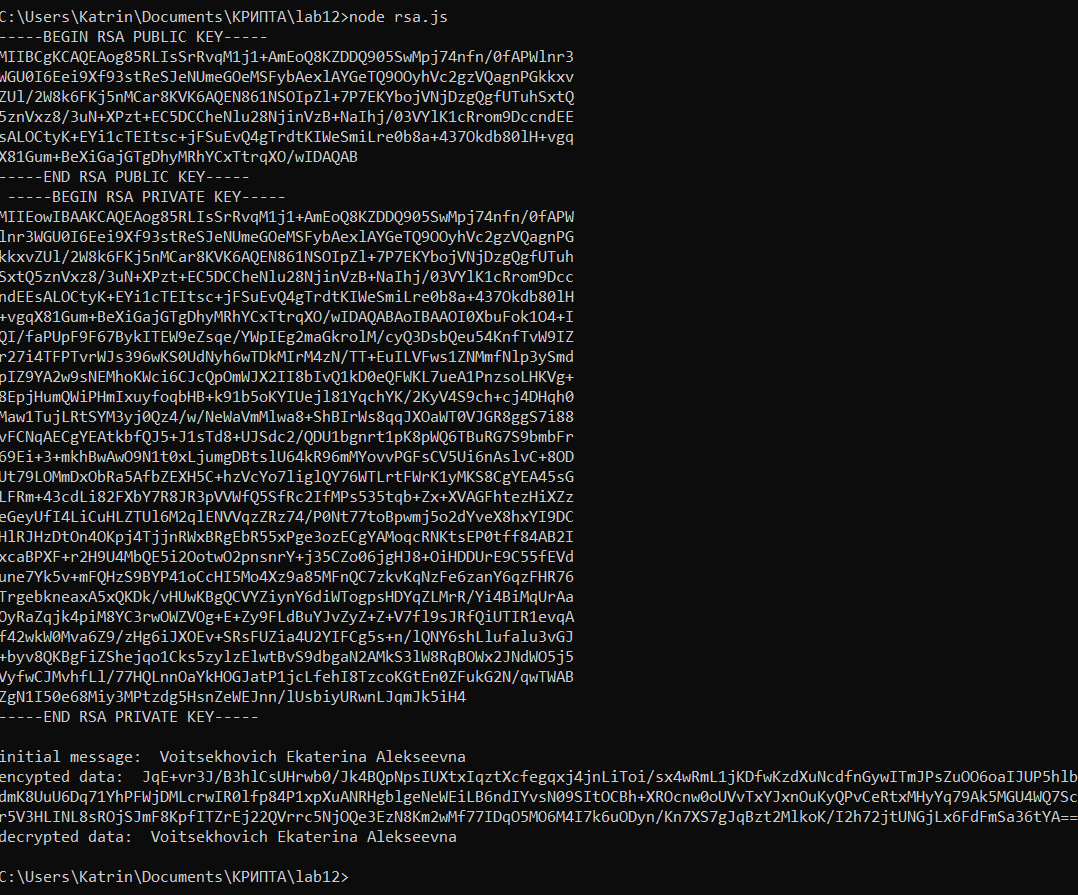


Рисунок 1 — Результат шифрования

|  |
| --- |
| let Alphabet = "!\"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^\_`abcdefghijklmnopqrstuvwxyz{|}~ \nπ®ƒ©∆";  Alphabet = Alphabet.split("");  let Crypto = function (alpha, gen, C) {  let p, B, encrypt, decrypt, f, g, modInv, modPow, toAlpha, to10;  toAlpha = function (x) {  let y, p, l, n;  if (x === 0) {  return "!!!!";  }  y = [];  n = 4;  n = Math.ceil(n);  while (n--) {  p = Math.pow(alpha.length, n);  l = Math.floor(x / p);  y.push(alpha[l]);  x -= l \* p;  }  y = y.join("");  return y;  };  to10 = function (x) {  let y, p, n;  y = 0;  p = 1;  x = x.split("");  n = x.length;  while (n--) {  y += alpha.indexOf(x[n]) \* p;  p \*= alpha.length;  }  return y;  };  modInv = function (gen, mod) {  let v, d, u, t, c, q;  v = 1;  d = gen;  t = 1;  c = mod % gen;  u = Math.floor(mod / gen);  while (d > 1) {  q = Math.floor(d / c);  d = d % c;  v = v + q \* u;  if (d) {  q = Math.floor(c / d);  c = c % d;  u = u + q \* v;  }  }  return d ? v : mod - u;  };  modPow = function (base, exp, mod) {  let c, x;  if (exp === 0) {  return 1;  } else if (exp < 0) {  exp = -exp;  base = modInv(base, mod);  }  c = 1;  while (exp > 0) {  if (exp % 2 === 0) {  base = (base \* base) % mod;  exp /= 2;  } else {  c = (c \* base) % mod;  exp--;  }  }  return c;  };  p = 91744613;  C = parseInt(C, 10);  if (isNaN(C)) {  C = Math.round(Math.sqrt(Math.random() \* Math.random()) \* (p - 2) + 2);  }  B = modPow(gen, C, p);  decrypt = function (a) {  let d, x, y;  x = a[1];  y = modPow(a[0], -C, p);  d = (x \* y) % p;  d = Math.round(d) % p;  return alpha[d - 2];  };  encrypt = function (key, d) {  let k, a;  k = Math.ceil(Math.sqrt(Math.random() \* Math.random()) \* 1E10);  d = alpha.indexOf(d) + 2;  a = [];  a[0] = modPow(key[1], k, key[0]);  a[1] = (d \* modPow(key[2], k, key[0])) % key[0];  return a;  };  f = function (message, key) {  let n, x, y, w;  y = [];  message = message.split("");  n = message.length;  while (n--) {  x = encrypt(key, message[n]);  y.push(toAlpha(x[0]));  y.push(toAlpha(x[1]));  }  y = y.join("");  return y;  };  g = function (message) {  let n, m, d, x;  m = [];  n = message.length / 8;  while (n--) {  x = message[8 \* n + 4];  x += message[8 \* n + 5];  x += message[8 \* n + 6];  x += message[8 \* n + 7];  m.unshift(x);  x = message[8 \* n];  x += message[8 \* n + 1];  x += message[8 \* n + 2];  x += message[8 \* n + 3];  m.unshift(x);  }  x = [];  d = [];  n = m.length / 2;  while (n--) {  x[0] = m[2 \* n];  x[1] = m[2 \* n + 1];  x[0] = to10(x[0]);  x[1] = to10(x[1]);  d.push(decrypt(x));  }  message = d.join("");  return message;  };  return {  pubKey: [p, gen, B],  priKey: C,  decrypt: g,  encrypt: f  };  };  document.querySelector(".elGamal").onclick = () => {  //69 это модификация примитивного корня 91744613  let Alice = Crypto(Alphabet, 69);  let Bob = Crypto(Alphabet, 69);  let message = "Voitsekhovich Ekaterina Alekseevna";  let startTime = new Date().getTime();  encrypted = Alice.encrypt(message, Bob.pubKey);  decrypted = Bob.decrypt(encrypted);  let endTime = new Date().getTime();  let execution = endTime - startTime;  document.querySelector(".elGamal").innerHTML = "Сообщение: <br>" + message + "<br><br>Зашифрованное сообщение:<br>" + encrypted + "<br><br><br>Расшифрованное сообщение:<br>" + decrypted + "<br><br> Время выполнения:<br>" + execution + " мс";  } |

Листинг 2 — Файл elgamal.js

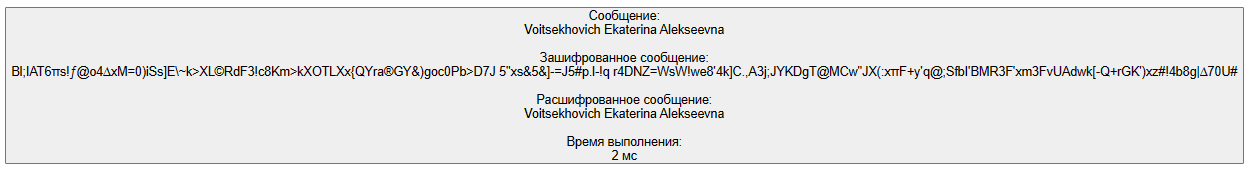


Рисунок 2 — Результат шифрования

**Вывод:** в ходе выполнения лабораторной работы были изучены алгоритмы генерации и верификации электронной цифровой подписи и приобретены практические навыки их реализации.