Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе 12»**

“ИССЛЕДОВАНИЕ АЛГОРИТМОВ ГЕНЕРАЦИИ И ВЕРИФИКАЦИИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ”

**Выполнила:** студентка 3 курса

4 группы специальности ПОИТ

Сапегина Екатерина Игоревна,

Вариант 11 (1)

Минск 2023

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

***Задачи:***

1. Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций генерации и верификации электронной цифровой подписи (ЭЦП).

2. Получить навыки практической реализации методов генерации и верификации ЭЦП на основе хеширования подписываемых сообщений и алгоритмов RSA, Эль-Гамаля и Шнорра, а также DSA.

3. Разработать приложение для реализации заданных алгоритмов генерации и верификации ЭЦП.

4. Оценить скорость генерации и верификации ЭЦП.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

***Практическая часть:***

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. Приложение должно реализовывать следующие операции:

• генерацию и верификацию ЭЦП на основе алгоритмов RSA, Эль-Гамаля и Шнорра;

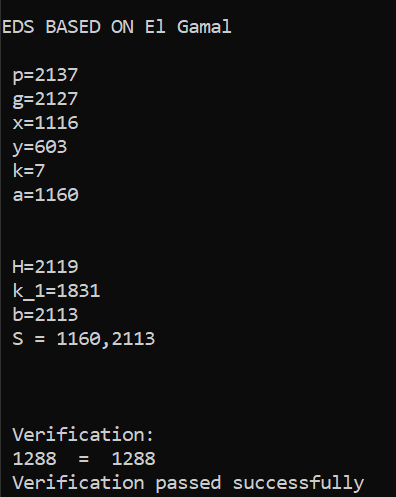
• оценку времени выполнения указанных процедур при реальных (требуемых) ключевых параметрах.

Для вычисления хешей можно также воспользоваться доступными online-средствами, например katvin (https://katvin.com/tools/ hash-generator.html).

1. Для выполнения необходимых операций передачи (по сети)/верификации информации обменяться открытой ключевой информацией с получателем подписанного сообщения для каждого исследуемого алгоритма (по согласованию с преподавателем).

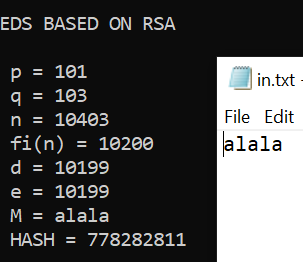
***EL GAMAL***

|  |
| --- |
| ***using System;***  ***using System.Collections.Generic;***  ***using System.Linq;***  ***using System.Text;***  ***using System.Threading.Tasks;***  ***using System.Security.Cryptography;***  ***using System.Numerics;***  ***using System.IO;***  ***using System.Diagnostics;***  ***namespace Ell\_Gamal\_sign***  ***{***  ***class EllGamal***  ***{***  ***public static int obr(int a, int n)***  ***{***  ***int res = 0;***  ***for (int i = 0; i < 10000; i++)***  ***{***  ***if (((a \* i) % n) == 1) return (i);***  ***}***  ***return (res);***  ***}***  ***static void Main(string[] args)***  ***{***  ***Console.WriteLine($"\nEDS BASED ON El Gamal\n");***  ***int p = 2137;***  ***int g = 2127;***  ***int x = 1116;***  ***int y = (int)BigInteger.ModPow(g, x, p);***  ***int k = 7;***  ***int a = (int)BigInteger.ModPow(g, k, p);***  ***Console.WriteLine($" p={p}\n g={g}\n x={x}\n y={y}\n k={k}\n a={a}\n\n");***  ***int H = 2119;***  ***int m = p - 1;***  ***int k\_1 = obr(k, p - 1);***  ***var b = new BigInteger((k\_1 \* (H - (x \* a) % m) % m) % m);***  ***Console.WriteLine($" H={H}\n k\_1={k\_1}\n b={b}\n S = {a},{b} \n\n");***  ***Console.WriteLine("\n Verification:");***  ***var ya = BigInteger.ModPow(y, a, p);***  ***var ab = BigInteger.ModPow(a, b, p);***  ***var pr1 = BigInteger.ModPow(ya \* ab, 1, p);***  ***var pr2 = BigInteger.ModPow(g, H, p);***  ***if (pr1 == pr2)***  ***{***  ***Console.WriteLine($" {pr1} = {pr2}\n Verification passed successfully");***  ***}***  ***Console.ReadKey();***  ***}***  ***}***  ***}*** |

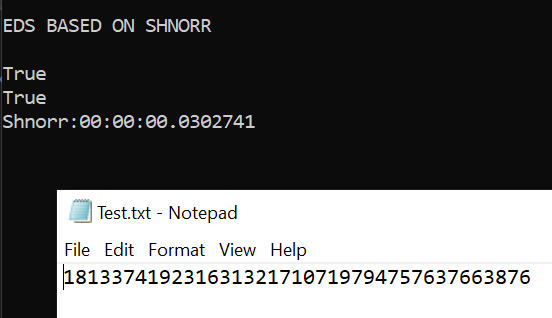
******

***RSA***

|  |
| --- |
| ***using System;***  ***using System.Collections.Generic;***  ***using System.Diagnostics;***  ***using System.IO;***  ***using System.Numerics;***  ***namespace RSA\_sign***  ***{***  ***class RSA***  ***{***  ***public static readonly char[] characters = new char[] { '#', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-' };***  ***public bool IsTheNumberSimple(long n)***  ***{***  ***if (n < 2) return false;***  ***if (n == 2) return true;***  ***for (long i = 2; i < n; i++)***  ***if (n % i == 0) return false;***  ***return true;***  ***}***  ***public int Calculate\_e(int d, int m)***  ***{***  ***int e = 10;***  ***while (true)***  ***{***  ***if ((e \* d) % m == 1) break;***  ***else e++;***  ***}***  ***return (int)e;***  ***}***  ***public int Calculate\_d(int m)***  ***{***  ***int d = m - 1;***  ***for (int i = 2; i <= m; i++)***  ***if ((m % i == 0) && (d % i == 0))***  ***{***  ***d--;***  ***i = 1;***  ***}***  ***return d;***  ***}***  ***public List<string> RSA\_Encode(string hash, int e, int n)***  ***{***    ***List<string> result = new List<string>();***  ***BigInteger bi;***  ***for (int i = 0; i < hash.Length; i++)***  ***{***  ***int index = Array.IndexOf(characters, hash[i]);***  ***bi = new BigInteger(index);***  ***bi = BigInteger.Pow(bi, (int)e);***  ***BigInteger n\_ = new BigInteger((int)n);***  ***bi = bi % n\_;***  ***result.Add(bi.ToString());***  ***}***  ***return result;***  ***}***  ***public string RSA\_Decode(List<string> input, int d, int n)***  ***{***  ***try***  ***{***  ***string result = "";***  ***BigInteger bi;***  ***foreach (string item in input)***  ***{***  ***bi = new BigInteger(Convert.ToDouble(item));***  ***bi = BigInteger.Pow(bi, (int)d);***  ***BigInteger n\_ = new BigInteger((int)n);***  ***bi = bi % n\_;***  ***int index = Convert.ToInt32(bi.ToString());***  ***result += characters[index].ToString();***  ***}***  ***return result;***  ***}***  ***catch (Exception ex) { return ""; }***  ***}***  ***}***  ***class Program***  ***{***  ***public static readonly char[] characters = new char[] { '#', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-' };***  ***static void Main(string[] args)***  ***{***  ***Console.WriteLine($"\nEDS BASED ON RSA\n");***  ***var rsa = new RSA();***  ***string M = File.ReadAllText("in.txt");***  ***Process.Start("in.txt");***  ***int p = 101;***  ***int q = 103;***  ***string hash = M.GetHashCode().ToString();***  ***int n = p \* q;***  ***int m = (p - 1) \* (q - 1);***  ***int d = rsa.Calculate\_d(m);***  ***int e\_ = rsa.Calculate\_e(d, m);***  ***Console.WriteLine($" p = {p}\n q = {q}\n n = {n}\n fi(n) = {m}\n d = {d}\n e = {e\_}\n M = {M}\n HASH = {hash}\n");***  ***List<string> sign = rsa.RSA\_Encode(hash, e\_, n);***  ***while (true)***  ***{***  ***Console.ReadKey();***  ***{***  ***List<string> input = new List<string>();***  ***string hash2 = File.ReadAllText("in.txt").GetHashCode().ToString();***  ***string result = rsa.RSA\_Decode(sign, d, n);***  ***Console.WriteLine($"Hash of the electronic signature = {result}");***  ***Console.WriteLine($"Hash of file = {hash2}");***  ***if (result.Equals(hash2)) Console.WriteLine("The file is genuine. The signature is correct. \n");***  ***else Console.WriteLine("WARNING! The file is NOT authentic!\n");***  ***}***  ***}***  ***}***  ***}***  ***}*** |

******

***SHNORR***

******

|  |
| --- |
| ***using System;***  ***using System.Collections.Generic;***  ***using System.Linq;***  ***using System.Text;***  ***using System.Threading.Tasks;***  ***using System.IO;***  ***using System.Numerics;***  ***using System.Security.Cryptography;***  ***namespace Shorn\_sign***  ***{***  ***public static class ElGamal***  ***{***  ***public static BigInteger CalculateMd5Hash(string input)***  ***{***  ***var md5 = MD5.Create();***  ***var inputBytes = Encoding.ASCII.GetBytes(input);***  ***var hash = md5.ComputeHash(inputBytes);***  ***return new BigInteger(hash.Concat(new byte[] { 0 }).ToArray());***  ***}***  ***}***  ***public static class Shnorr***  ***{***  ***public static void Do()***  ***{***  ***BigInteger p = 2267;***  ***BigInteger q = 103;***  ***string text = File.ReadAllText(".\\text.txt");***  ***BigInteger g = 354; //взаимопрост с p***  ***BigInteger obg = 967;***  ***int x = 30;***  ***BigInteger y = BigInteger.ModPow(obg, x, p);***  ***BigInteger a = BigInteger.Pow(g, 13) % p;***  ***BigInteger hash = ElGamal.CalculateMd5Hash(text + a.ToString());***  ***File.WriteAllText(".\\Test.txt", hash.ToString());***  ***BigInteger b = (13 + x \* hash) % q;***  ***BigInteger dov = BigInteger.ModPow(g, b, p);***  ***BigInteger X = (dov \* BigInteger.ModPow(y, hash, p)) % p;***  ***BigInteger hash2 = ElGamal.CalculateMd5Hash((text + X.ToString()));***  ***var f = hash == hash2;***  ***Console.WriteLine(f);***  ***string text2 = File.ReadAllText(".\\FakeTest.txt");***  ***BigInteger hash3 = ElGamal.CalculateMd5Hash((text2 + X.ToString()));***  ***var f2 = hash == hash3;***  ***Console.WriteLine(f2);***  ***}***  ***}***  ***class Program***  ***{***  ***static void Main()***  ***{***  ***Console.WriteLine($"\nEDS BASED ON SHNORR\n");***  ***Console.InputEncoding = Encoding.ASCII;***  ***var t = DateTime.Now;***  ***Shnorr.Do();***  ***Console.WriteLine("Shnorr:" + (DateTime.Now - t));***  ***Console.ReadLine();***  ***}***  ***}***  ***}*** |