Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе 10»**

“Исследование асимметричных шифров

RSA и Эль-Гамаля”

**Выполнила:** студентка 3 курса

4 группы специальности ПОИТ

Сапегина Екатерина Игоревна,

Вариант 11 (1)

Минск 2023

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

***Задачи:***

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров RSA и Эль-Гамаля.

2. Разработать приложение для реализации асимметричного

зашифрования/расшифрования на основе алгоритмов RSA и Эль-Гамаля.

3. Выполнить анализ криптостойкости асимметричных шифров RSA и Эль-Гамаля.

4. Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить

в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

***Практическая часть:***

1. С помощью простого консольного приложения составить

табличную или графическую форму зависимости времени

вычисления параметра у, функционально заданного выражением

вида:

у ≡ ax mod n,

от параметров: а (десятичные числа от 5 до 35; можно взять 1 или

2 числа), х (числа, желательно простые, из диапазона от 103 до 10100;

для примера взять 5–10 чисел, равномерно распределенных в указанном диапазоне), n (для примера взять числа, в двоичном виде состоящие из 1024 и 2048 битов).

2. 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.

Исходный текст для зашифрования – собственные фамилия, имя,

отчество. Для численного представления блоков текста можно в том

числе пользоваться указанными выше кодировочными таблицами.

Ключевую информацию для обоих алгоритмов можно сгенерировать самостоятельно либо воспользоваться, например, одной

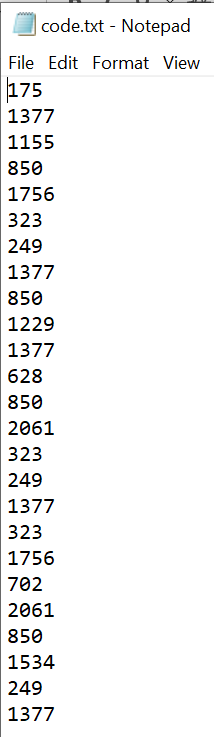
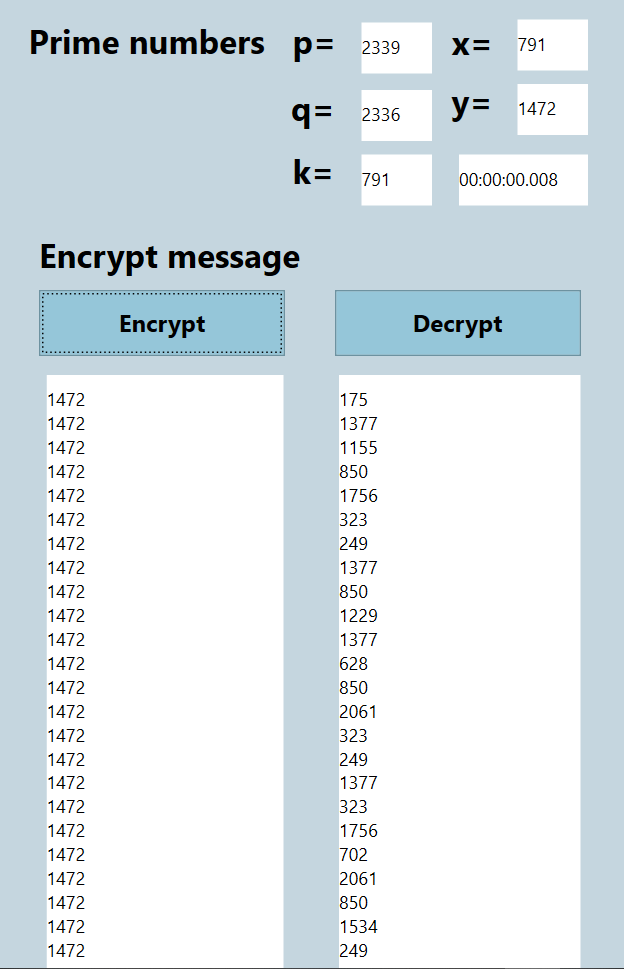
из утилит криптографической библиотеки OpenSSL, с помощью

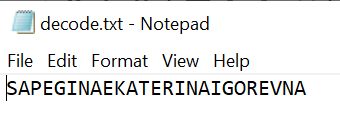
которой, в частности, можно сгенерировать ключевую информацию для алгоритма RSA.

***Реализация:***

Эль-Гамаль

|  |
| --- |
| private void codeButton\_Click(object sender, RoutedEventArgs e)  {  try  {  aNumber.Text = "";  resultNumber.Text = "";  p\_main = Search\_p();  Random random = new Random();  x = random.Next(1, p\_main - 1); //Генерирую закрытый ключ  BigInteger y = BigInteger.Pow(g\_main, x) % p\_main; //Нахожу открытый ключ  pNumber.Text = p\_main.ToString();  gNumber.Text = g\_main.ToString();  xNumber.Text = x.ToString();  yNumber.Text = y.ToString();  string s = "";  StreamReader sr = new StreamReader("text.txt");  while (!sr.EndOfStream)  {  s += sr.ReadLine();  }  sr.Close();  textLength = s.Length;  s = s.ToUpper();  resultCode.RemoveAll(u => true);  var startTime = System.Diagnostics.Stopwatch.StartNew();  resultCode = Code(s, p\_main, y);  startTime.Stop();  var resultTime = startTime.Elapsed;  string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:000}",  resultTime.Hours,  resultTime.Minutes,  resultTime.Seconds,  resultTime.Milliseconds);  timeWork.Text = elapsedTime;  kNumber.Text = k.ToString();  StreamWriter sw = new StreamWriter("code.txt");  foreach(var item in a)  {  aNumber.Text += item.ToString() + '\n';  }  foreach (var item in resultCode)  {  sw.WriteLine(item);  resultNumber.Text += item.ToString() + '\n';  }  sw.Close();    }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private void decodeButton\_Click(object sender, RoutedEventArgs e)  {  try  {  List<string> input = new List<string>();  StreamReader sr = new StreamReader("code.txt");  while (!sr.EndOfStream)  {  input.Add(sr.ReadLine());  }  sr.Close();  var startTime = System.Diagnostics.Stopwatch.StartNew();  string resultDecode = Decode(textLength, resultCode, x, p\_main);  startTime.Stop();  var resultTime = startTime.Elapsed;  string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:000}",  resultTime.Hours,  resultTime.Minutes,  resultTime.Seconds,  resultTime.Milliseconds);  timeWork.Text = elapsedTime;  StreamWriter sw = new StreamWriter("decode.txt");  sw.WriteLine(resultDecode);  sw.Close();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  public static List<BigInteger> Code(string text, int p, BigInteger y)  {  List<BigInteger> array = new List<BigInteger>();  a.RemoveAll(x => true);  Random random = new Random();  k = random.Next(1, p - 1);    for (int i = 0; i != text.Length; i++)  {  a.Add(BigInteger.Pow(g\_main, k) % p);    array.Add((BigInteger.Pow(y, k) \* (int)text[i]) % p);  }  return array;  }  public static bool Search\_g(int p, int g)  {  bool boolean = false;  List<BigInteger> array\_mod\_number = new List<BigInteger>();  BigInteger integer = ((BigInteger.Pow(g, 1)) % p);  array\_mod\_number.Add(integer);  for (int i = 2; i != p; i++)  {  integer = BigInteger.Pow(g, i) % p;  for (int j = 0; j != i - 1; j++)  {  if (array\_mod\_number[j] == integer)  {  g--;  array\_mod\_number.Clear();  i = 1;  integer = BigInteger.Pow(g, 1) % p;  array\_mod\_number.Add(integer);  break;  }  if ((j == i - 2) && (array\_mod\_number[j] != integer))  {  array\_mod\_number.Add(integer);  }  }  }  g\_main = g;  boolean = true;  return boolean;  }  public static string Decode(int length\_text, List<BigInteger> array\_number, int x, int p)  {  string save\_text = "";  BigInteger integer;  for (int i = 0; i != length\_text; i++)  {  integer = (array\_number[i] \* (BigInteger.Pow(a[i], p - 1 - x))) % p;  save\_text += (char)integer;  }  return save\_text;  }  public static int Search\_p()  {  Random random = new Random();  int p = 0;  Boolean boolean = false;  do  {  p = random.Next(2000, 2500);  for (int i = 2; i != p; i++)  {  if (i == p - 1)  {  boolean = Search\_g(p, p - 1);  break;  }  if (p % i == 0) break;  }  }  while (boolean == false);  return p;  }  } |





RSA

|  |
| --- |
| public partial class MainWindow : Window  {  char[] characters = new char[] { '#', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И',  'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С',  'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', 'Ъ',  'Э', 'Ю', 'Я', ' ', '1', '2', '3', '4', '5', '6', '7',  '8', '9', '0' };  public MainWindow()  {  InitializeComponent();  }  private void codeButton\_Click(object sender, RoutedEventArgs e)  {  try  {  if (pNumber.Text.Length > 0 && qNumber.Text.Length > 0)  {  int p = Convert.ToInt32(pNumber.Text);  int q = Convert.ToInt32(qNumber.Text);  if (IsTheNumberSimple(p) && IsTheNumberSimple(q))  {  string s = "";  StreamReader sr = new StreamReader("text.txt");  while (!sr.EndOfStream)  {  s += sr.ReadLine();  }  sr.Close();  s = s.ToUpper();  long n = p \* q;  long m = (p - 1) \* (q - 1);  long e\_ = Calculate\_e(m);  long d = Calculate\_d(e\_, m);  var startTime = System.Diagnostics.Stopwatch.StartNew();  List<string> result = RSA\_Endoce(s, e\_, n);  startTime.Stop();  var resultTime = startTime.Elapsed;  string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:000}",  resultTime.Hours,  resultTime.Minutes,  resultTime.Seconds,  resultTime.Milliseconds);  timeWork.Text = elapsedTime;  StreamWriter sw = new StreamWriter("code.txt");  foreach (string item in result)  sw.WriteLine(item);  sw.Close();  dNumber.Text = d.ToString();  nNumber.Text = n.ToString();  eNumber.Text = e\_.ToString();  nNumber\_Copy.Text = n.ToString();  }  else  MessageBox.Show("p и q должны быть простыми");  }  else  MessageBox.Show("Введите p и q");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private void decodeButton\_Click(object sender, RoutedEventArgs e)  {  try  {  if ((dNumber.Text.Length > 0) && (nNumber\_Copy.Text.Length > 0))  {  long d = Convert.ToInt64(dNumber.Text);  long n = Convert.ToInt64(nNumber\_Copy.Text);  List<string> input = new List<string>();  StreamReader sr = new StreamReader("code.txt");  while (!sr.EndOfStream)  {  input.Add(sr.ReadLine());  }  sr.Close();  var startTime = System.Diagnostics.Stopwatch.StartNew();  string result = RSA\_Dedoce(input, d, n);  startTime.Stop();  var resultTime = startTime.Elapsed;  string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:000}",  resultTime.Hours,  resultTime.Minutes,  resultTime.Seconds,  resultTime.Milliseconds);  timeWork.Text = elapsedTime;    StreamWriter sw = new StreamWriter("decode.txt");  sw.WriteLine(result);  sw.Close();  }  else  MessageBox.Show("Введите секретный ключ!");  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  private List<string> RSA\_Endoce(string s, long e, long n)  {  List<string> result = new List<string>();  BigInteger bi;  for (int i = 0; i < s.Length; i++)  {  int index = Array.IndexOf(characters, s[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;  }  private 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;  }  //вычисление параметра e. e должно быть взаимно простым с m  private long Calculate\_e(long m)  {  long e = m - 1;// е это взаимно простое число с фи от n m- это фи от n  for (long i = 2; i <= m; i++)  if ((m % i == 0) && (e % i == 0)) //если имеют общие делители  {  e--;  i = 1;  }  return e;  }  //вычисление параметра d  private long Calculate\_d(long e, long m)  {  long d = 1000;  while (true)  {  if ((e \* d) % m == 1)  break;  else  d++;  }  return d;  }    private string RSA\_Dedoce(List<string> input, long d, long n)  {  string result = "";  System.Numerics.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;  }  } |

