Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

радиоэлектроники

Кафедра ЭВМ

Отчет

по лабораторной работе № 2

Выполнил: Проверил:

студент группы 050503 доцент, к.т.н

Торчилов П.Р. Кобяк И. П.

Минск 2014

1. **Исходные данные.**

А и B условились организовать между собой секретную переписку, скрытую от наблюдателя С. С этой целью они выбирают достаточно большое простое число P, такое что значение P-1 хорошо разлагается на большое число множителей. Каждый из абонентов, независимых друг от друга выбирают числа – ключи а и b. После чего рассчитывают сопутствующие параметры α и β.

Для расчета используют:

а α=1(mod р-1), где 0<α< p-1;

b β=1(mod р-1), где 0<β< p-1.

Пусть А решает передать В сообщение µ. А зашифровывает µ:

=(mod p), где 0<,µ<p.

B принимает сообщение и шифрует его:

=(mod p), где 0<<p.

На следующем этапе А формирует .

=(mod p), где 0<<p.

В принимает сообщение и формирует .

=(mod p), где 0<<p.

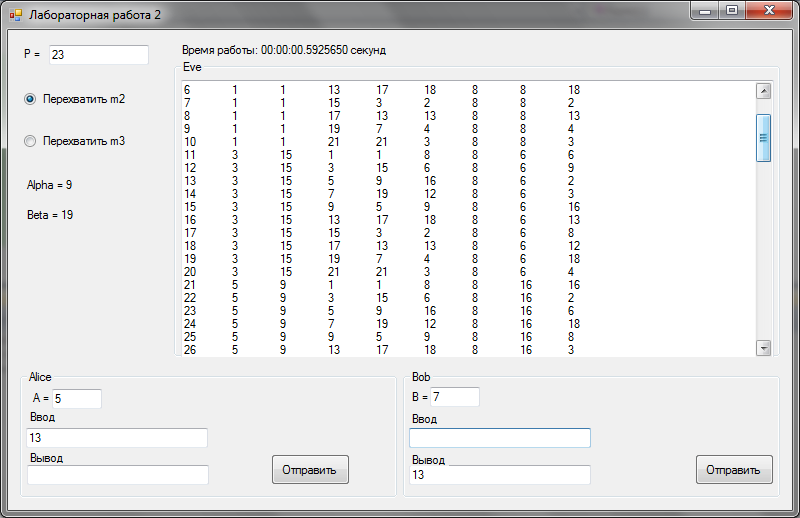
Получаем =µ.

1. **Задание.**

Выбрать P < 100. Затем выбрать a, b и рассчитать для них сопутствующие неслучайные параметры α, β.

Абонент C наблюдает μ2 или μ3. Зная P необходимо определить все ключи для шифрования сообщения.

1. **Результаты работы программы.**



1. **Листинг программы**

public void FindAllKeys()

{

for (var alpha = 1; alpha < P - 1; alpha++)

{

for (var a = 1; a < P - 1; a++)

{

if ((alpha \* a) % (P - 1) == 1)

{

for (var beta = 1; beta < P - 1; beta++)

{

for (var b = 1; b < P - 1; b++)

{

if ((beta \* b) % (P - 1) == 1 &&

(alpha \* a \* beta \* b) % (P - 1) == 1)

{

AlphaList.Add(alpha);

BetaList.Add(beta);

AList.Add(a);

BList.Add(b);

}

}

}

}

}

}

}

public class Transceiver

{

public KeyTool Send(String message)

{

int value;

FindAlpha();

FindBeta();

if (Int32.TryParse(message, out value))

{

keys.M1 = Pow(value, keys.A, KeyTool.P);

keys.M2 = Pow(keys.M1, keys.B, KeyTool.P);

keys.M3 = Pow(keys.M2, keys.Alpha, KeyTool.P);

keys.M4 = Pow(keys.M3, keys.Beta, KeyTool.P);

return keys;

}

return null;

}

public static int Pow(long value, int param, long p)

{

var pow = BigInteger.Pow(value, param);

var result = BigInteger.Divide(pow, new BigInteger(p));

result = BigInteger.Multiply(result, new BigInteger(p));

result = BigInteger.Subtract(pow, result);

return Int32.Parse(result.ToString());

}

private void FindAlpha()

{

for (var i = 1; i < KeyTool.P; i++)

{

if ((keys.A \* i) % (KeyTool.P - 1) == 1)

{

keys.Alpha = i;

break;

}

}

}

private void FindBeta()

{

for (var i = 1; i < KeyTool.P; i++)

{

if ((keys.B \* i) % (KeyTool.P - 1) == 1)

{

keys.Beta = i;

break;

}

}

}

}

private void Button2Click(object sender, EventArgs e)

{

if (this.ValidateP(textBox3.Text) && this.ValidateInput(textBox6.Text))

{

int a, b;

if (Int32.TryParse(textBox1.Text, out a) &&

Int32.TryParse(textBox2.Text, out b))

{

var keys = new KeyTool { A = a, B = b };

var transceiver = new Transceiver(keys);

var result = transceiver.Send(textBox6.Text);

if (result != null && this.ValidateParams(result))

{

label8.Text = @"Alpha = " + result.Alpha;

label9.Text = @"Beta = " + result.Beta;

textBox5.Text = result.M4.ToString(CultureInfo.InvariantCulture);

keys.FindAllKeys();

this.ShowAllKeys(keys);

}

}

}

}

private void ShowAllKeys(KeyTool keys)

{

textBox8.Clear();

for (var i = 0; i < keys.AList.Count; i++)

{

int m1, m2, m3, m4;

if (radioButton1.Checked)

{

m3 = Transceiver.Pow(keys.M2, keys.AlphaList[i], KeyTool.P);

m4 = Transceiver.Pow(m3, keys.BetaList[i], KeyTool.P);

m1 = Transceiver.Pow(m4, keys.AList[i], KeyTool.P);

m2 = Transceiver.Pow(m1, keys.BList[i], KeyTool.P);

}

else

{

m4 = Transceiver.Pow(keys.M3, keys.BetaList[i], KeyTool.P);

m1 = Transceiver.Pow(m4, keys.AList[i], KeyTool.P);

m2 = Transceiver.Pow(m1, keys.BList[i], KeyTool.P);

m3 = Transceiver.Pow(m2, keys.AlphaList[i], KeyTool.P);

}

textBox8.Text += String.Format(

"{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}\t{7}\t{8}\t{9}",

i + 1,

keys.AlphaList[i],

keys.AList[i],

keys.BetaList[i],

keys.BList[i],

m1,

m2,

m3,

m4,

Environment.NewLine);

}

}