Министерство образования и науки Российской Федерации  
Новосибирский Государственный Технический Университет

Кафедра Прикладной математики

Основы криптографии

Лабораторная работа №2

|  |  |
| --- | --- |
| Факультет: | ПМИ |
| Группа: | ПМ-04 |
| Студенты: | Белокриницкая С. С.  Сальников Д. Д.  Сумин Я. Е. |
| Преподаватель: | Ступаков И.М. |
|  |  |

Новосибирск 2022

1. **Цель**

Знакомство с ассиметричными криптографическими алгоритмами. Все сделанные пункты работы должны быть описаны в отчете.

**Вариант 1** (15 баллов). Алгоритм Диффи-Хеллмана

1. **Основная идея алгоритма**

Предположим, существует два абонента: Алиса и Боб. Обоим абонентам известны два числа **g** и **p**, таких что *p* является простым числом*, g* является первообразным корнем по модулю *p*, которые не являются секретными и могут быть известны также другим заинтересованным лицам. Для того, чтобы создать неизвестный более никому секретный ключ, оба абонента генерируют большие случайные числа: Алиса — число ***a***, Боб — число ***b***. Затем Алиса вычисляет значение (1):

|  |  |
| --- | --- |
|  | (1) |

и пересылает его Бобу, а Боб вычисляет (2):

|  |  |
| --- | --- |
|  | (2) |

и передаёт Алисе. Предполагается, что злоумышленник может получить оба этих значения, но не модифицировать их (то есть, у него нет возможности вмешаться в процесс передачи).

На втором этапе Алиса на основе имеющегося у неё ***a*** и полученного по сети ***B*** вычисляет значение (3):

|  |  |
| --- | --- |
|  | (3) |

Боб на основе имеющегося у него ***b*** и полученного по сети ***A*** вычисляет значение:

|  |  |
| --- | --- |
|  | (4) |

Как нетрудно видеть, у Алисы и Боба получилось одно и то же число:

|  |  |
| --- | --- |
|  | (5) |

Его они и могут использовать в качестве секретного ключа, поскольку здесь злоумышленник встретится с практически неразрешимой (за разумное время) проблемой вычисления (3) или (4) по перехваченным **A**{\displaystyle g^{a}{\bmod {p}}} и **B**{\displaystyle g^{b}{\bmod {p}}}, если числа ***p***, ***a***, ***b*** выбраны достаточно большими.

1. **Текст программы**

*ClientServer.cs*

using Dia2Lib;

using System.IO;

using System.Net;

using System.Net.Http;

using System.Net.Sockets;

using System.Text;

using Сrypt2.DiffieHellman.GenerationKey;

namespace ClientServer

{

public class Program

{

static void Main()

{

Console.WriteLine("Enter the number p, which will be simple");

\_ = uint.TryParse(Console.ReadLine(), out uint p);

Console.WriteLine("Enter the number g, which will be the primitive root of p");

\_ = uint.TryParse(Console.ReadLine(), out uint g);

if (p == 0) { Console.WriteLine("p is not a natural number!"); return; }

if (g == 0) { Console.WriteLine("q is not a natural number!"); return; }

KeyGeneration keyGeneration = new(p, g);

if (!keyGeneration.IsPrime()) { Console.WriteLine("p is not a prime nubmer!"); return; }

if (!keyGeneration.IsPrimitive()) { Console.WriteLine("g is not a primitive number modulo p"); return; }

var PublicKeyA = keyGeneration.PublicKey();

Console.WriteLine("\np = " + keyGeneration.p);

Console.WriteLine("g = " + keyGeneration.g);

Console.WriteLine("\nSend A = " + PublicKeyA);

try

{

TcpListener tcpListener = new(IPAddress.Any, 7000);

tcpListener.Start();

TcpClient tcpClient = tcpListener.AcceptTcpClient();

NetworkStream stream = tcpClient.GetStream();

var y1 = Encoding.ASCII.GetBytes(p.ToString() + " ");

var y2 = Encoding.ASCII.GetBytes(g.ToString() + " ");

var y3 = Encoding.ASCII.GetBytes(PublicKeyA.ToString() + " ");

stream.Write(y1, 0, y1.Length);

stream.Write(y2, 0, y2.Length);

stream.Write(y3, 0, y3.Length);

stream.Flush();

byte[] bytesRead = new byte[4];

int length = stream.Read(bytesRead, 0, bytesRead.Length);

var PublicKeyB = Convert.ToUInt64(Encoding.ASCII.GetString(bytesRead, 0, length));

Console.WriteLine("Got B = " + PublicKeyB);

var secret = keyGeneration.BinaryPower(PublicKeyB, keyGeneration.a);

Console.WriteLine("Secret = " + secret);

Console.WriteLine("Enter to finish!");

tcpClient.Close();

tcpListener.Stop();

Console.ReadKey();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

*Client.cs*

using System.ComponentModel;

using System.Net.Sockets;

using System.Text;

using Сrypt2.DiffieHellman.GenerationKey;

namespace Client

{

public class Client

{

static void Main()

{

try

{

TcpClient tcpClient = new("192.168.0.12", 7000);

NetworkStream stream = tcpClient.GetStream();

byte[] bytesRead = new byte[256];

int length = stream.Read(bytesRead, 0, bytesRead.Length);

var answer = Encoding.ASCII.GetString(bytesRead, 0, length);

var l = answer.Split(" ");

var value1 = UInt64.Parse(l[0]);

Console.WriteLine(l[1]);

var value2 = UInt64.Parse(l[1]);

KeyGeneration key = new(value1, value2);

var PublicKeyB = key.PublicKey();

// Отправляем B

string request = PublicKeyB.ToString();

byte[] bytesWrite = Encoding.ASCII.GetBytes(request); ;

stream.Write(bytesWrite, 0, bytesWrite.Length);

stream.Flush();

var secret = key.BinaryPower(Convert.ToUInt64(l[2]), key.a);

Console.WriteLine(secret);

Console.WriteLine("Enter to finish!");

tcpClient.Close();

Console.ReadKey();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

*KeyGeneration.cs*

namespace Сrypt2.DiffieHellman.GenerationKey;

using static Math;

public class KeyGeneration

{

public readonly ulong p;

public readonly ulong g;

public ulong a { get; private set; }

public KeyGeneration(ulong p, ulong g)

{

this.p = p;

this.g = g;

}

public KeyGeneration() : base() { }

#region Проверка на простоту числа p

public bool IsPrime()

{

if (p < 2) return false;

for (uint i = 2; i <= Truncate(Sqrt(p)); ++i)

if (p % i == 0) return false;

return true;

}

#endregion

#region Проверка g на первообразный корень

public bool IsPrimitive()

{

var tmp = p - 1;

if (BinaryPower(g, tmp) != 1) return false;

for (ulong i = 2; i < Truncate(Sqrt(tmp)); ++i)

{

if (tmp % i == 0)

{

if (BinaryPower(g, i) == 1 || BinaryPower(g, tmp / i) == 1)

return false;

}

}

return true;

}

#endregion

#region Быстрое возведение в степень по модулю

public ulong BinaryPower(ulong b, ulong e)

{

ulong v = 1;

while (e != 0)

{

if ((e & 1) != 0)

{

v \*= b;

v %= this.p;

}

b \*= b;

b %= this.p;

e >>= 1;

}

return v;

}

#endregion

#region Вычисление A

public ulong PublicKey()

{

this.a = RandomValue();

return BinaryPower(g, a);

}

#endregion

#region Случайное число в диапазоне от 1..(p - 1)

private ulong RandomValue()

{

Random rnd = new();

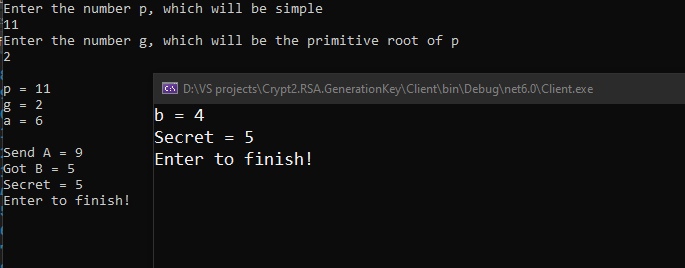
return Convert.ToUInt64(rnd.Next(1, Convert.ToInt32(p - 1)));

}

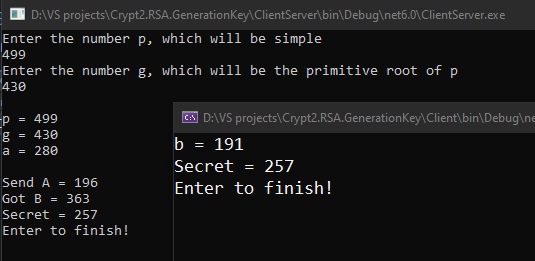
#endregion

}

1. **Тесты**



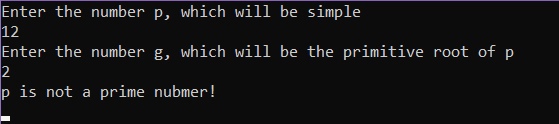
Числа совпали.



Числа совпали.

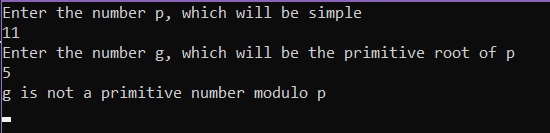
3)

р – не простое число:



4)

g – не первообразный корень



5)

Числа совпали.

