Util.cs

namespace Encryption

{

static class Util

{

public static bool IsPrime(int n)

{

int i, m = 0;

m = n / 2;

for (i = 2; i <= m; i++)

{

if (n % i == 0)

return false;

}

return true;

}

public static int RandomPrime(int min,int max)

{

Random r = new Random();

int random = min + r.Next(max-min);

while (true)

{

if (IsPrime(random))

return random;

else

random = min + r.Next(max-min);

}

}

public static int GCD(int a, int b)

{

while (a != 0 && b != 0)

{

if (a > b)

a %= b;

else

b %= a;

}

return a == 0 ? b : a;

}

public static int NegMod(int inp, int mod)

{

int div = inp / mod;

return inp - (div - 1) \* mod;

}

public static int InverseMod(int inp, int mod)

{

int m = 1;

for (int i = 1; i < mod; i++)

{

if (inp \* i % mod == 1)

{

m = i;

break;

}

}

return m;

}

}

}

Elgamel.cs

namespace Encryption

{

class Elgamel

{

int p, x, y, alpha, k, m, r, s;

public int M { get { return m; } set { m = value; } }

public Elgamel()

{

}

public void SetAttr(int p, int a, int? x, int? k)

{

this.p = p;

if (x.HasValue)

{

this.x = x??0;

}

else

{

for (int i = 1; i < p; i++)

{

if (Util.GCD(i, p - 1) == 1)

{

this.x = i;

break;

}

}

}

this.alpha = a;

if (k.HasValue)

{

this.k = k ?? 0;

}

else

{

for (int i = 1; i < p; i++)

{

if (Util.GCD(i, p - 1) == 1)

{

this.k = i;

break;

}

}

}

}

public void KeyGen()

{

Console.WriteLine("Private Key: " + x);

this.y = ((int) Math.Pow(alpha, x)) % p;

Console.WriteLine("Public Key: {0}",y);

}

public void Sign()

{

this.r = ((int)Math.Pow(alpha, k)) % p;

this.s = Util.InverseMod(k,p-1) \* ((m - r \* x) < 0 ? Util.NegMod(m - r \* x,p-1):((m-r\*x)%(p-1))) % (p-1);

Console.WriteLine("Signed Message: [{0},({1},{2})]", m, r, s);

}

public bool Verify()

{

return Math.Pow(alpha, m) % p == Math.Pow(y, r) \* Math.Pow(r, s) % p;

}

}

}

DiffieHellman.cs

namespace Encryption

{

class DiffieHellman

{

int p, g, xa, ya, xb, yb, k;

public void SetAttr(int p, int g, int xa, int xb) {

this.p = p;

this.g = g;

this.xa = xa;

this.xb = xb;

}

public void KeyGen()

{

ya = (int) Math.Pow(g,xa) % p;

yb = (int)Math.Pow(g, xb) % p;

k = (int)Math.Pow(yb, xa) % p;

Console.WriteLine("Private Key of A: {0}", xa);

Console.WriteLine("Public Key of A: {0}", ya);

Console.WriteLine("Private Key of B: {0}", xb);

Console.WriteLine("Public Key of B: {0}", yb);

Console.WriteLine("k = {0}", k);

}

}

}

Program.cs

namespace Encryption

{

class Program

{

static void Main(string[] args)

{

int p, a, x, k, m;

Console.WriteLine("Enter Values of p, a, x and k:");

p = Int32.Parse(Console.ReadLine());

a = Int32.Parse(Console.ReadLine());

x = Int32.Parse(Console.ReadLine());

k = Int32.Parse(Console.ReadLine());

Console.WriteLine("Enter Message:");

m = Int32.Parse(Console.ReadLine());

var el = new Elgamel();

el.SetAttr(p, a, x, k);

el.M = m;

el.KeyGen();

el.Sign();

Console.WriteLine("Verification: " + (el.Verify() ? "Success" : "Failed"));

}

}

}

