Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Лабораторная работа 3

Проверил: Чербу Ольга

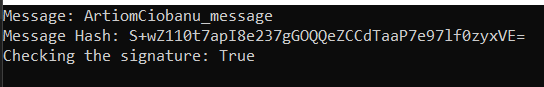
Выполнил: Чобану Артём

Кишинев 2022

**Алгоритм One-Time Электронной подписи Lamport**

Шифруемое слово: **ArtiomCiobanu\_message**

Скриншоты работы программы:



Код программы:

using System.Numerics;  
using System.Security.Cryptography;  
using System.Text;  
  
uint k = 256;  
var privateKey = GetMessages(k);  
var privateKeyBytes = ByteArrayFromTupleArray(privateKey);  
  
var publicKey = GetMD5HashForEachMessage(privateKey);  
var publicKeyBytes = ByteArrayFromTupleArray(privateKey);  
  
string message = "ArtiomCiobanu\_message";  
var messageHash = SHA256.Create().ComputeHash(Encoding.UTF8.GetBytes(message));  
  
var signed = Sign(messageHash, privateKey);  
  
var isValid = Verify(messageHash, signed, publicKey);  
  
Console.WriteLine($"Message: {message}");  
Console.WriteLine($"Message Hash: {Convert.ToBase64String(messageHash)}");  
  
Console.WriteLine($"Checking the signature: {isValid}");  
  
  
static List<byte[]> Sign(  
 byte[] messageHash,  
 Tuple<byte[], byte[]>[] privateKey)  
{  
 var messageHashBigInteger = new BigInteger(messageHash);  
  
 var result = new List<byte[]>();  
 for (int i = 0; i < 256; i++)  
 {  
 var currentBit = (BigInteger)1 << (255 - i);  
  
 var value = messageHashBigInteger & currentBit;  
  
 var valueToAdd = value == currentBit  
 ? privateKey[i].Item1  
 : privateKey[i].Item2;  
 result.Add(valueToAdd);  
 }  
  
 return result;  
}  
  
static bool Verify(  
 byte[] messageHash,  
 IEnumerable<byte[]> signed,  
 Tuple<byte[], byte[]>[] publicKey)  
{  
 var messageHashBigInteger = new BigInteger(messageHash);  
  
 var sequence = new List<byte[]>();  
 for (int i = 0; i < 256; i++)  
 {  
 var currentBit = (BigInteger)1 << (255 - i);  
  
 var value = messageHashBigInteger & currentBit;  
  
 var valueToAdd = value == currentBit  
 ? publicKey[i].Item1  
 : publicKey[i].Item2;  
  
 sequence.Add(valueToAdd);  
 }  
  
 var signedHash = signed.Select(h => SHA256.Create().ComputeHash(h)).ToArray();  
  
 for (int i = 0; i < sequence.Count(); i++)  
 {  
 if (!sequence[i].SequenceEqual(signedHash[i]))  
 {  
 return false;  
 }  
 }  
  
 return true;  
}

Tuple<byte[], byte[]>[] GetMessages(uint k)  
{  
 Tuple<BigInteger, BigInteger>[] p = new Tuple<BigInteger, BigInteger>[k];  
  
 for (uint i = 0; i < p.Length; i++)  
 {  
 var number = (BigInteger)1 << 254;  
  
 p[i] = new Tuple<BigInteger, BigInteger>(number + i, number + p.Length + i);  
 }  
  
 var result = p.Select(t => new Tuple<byte[], byte[]>(  
 t.Item1.ToByteArray(),  
 t.Item2.ToByteArray()));  
  
 return result.ToArray();  
}  
  
Tuple<byte[], byte[]>[] GetMD5HashForEachMessage(Tuple<byte[], byte[]>[] input)  
{  
 var result = new Tuple<byte[], byte[]>[input.Length];  
  
 var md5 = SHA256.Create();  
 for (int i = 0; i < input.Length; i++)  
 {  
 var hash1 = md5.ComputeHash(input[i].Item1);  
 var hash2 = md5.ComputeHash(input[i].Item2);  
 result[i] = new Tuple<byte[], byte[]>(hash1, hash2);  
 }  
  
 return result;  
}  
  
static byte[] ByteArrayFromTupleArray(Tuple<byte[], byte[]>[] tuples)  
{  
 var result = tuples  
 .Select(l => new[] { l.Item1, l.Item2 }.ToList())  
 .SelectMany(l => l)  
 .SelectMany(l => l)  
 .ToArray();  
  
 return result;  
}

static byte[] IntArrayToByteArray(uint[] integerArray)  
{  
 byte[] result = new byte[integerArray.Length \* 4];  
  
 for (int i = 0; i < result.Length; i++)  
 {  
 result[i] = (byte)((integerArray[i / 4] >> i % 4 \* 8) & 0xff);  
 }  
  
 return result;  
}  
  
static uint[] ByteArrayToIntArray(byte[] bytes)  
{  
 uint[] result = new uint[bytes.Length / 4];  
  
 for (int i = 0, counter = 0; i < result.Length; i++)  
 {  
 var current = (bytes[counter++] & 0xff) |  
 ((bytes[counter++] & 0xff) << 8) |  
 ((bytes[counter++] & 0xff) << 16) |  
 ((bytes[counter++] & 0xff) << 24);  
  
 result[i] = (uint)current;  
 }  
  
 return result;  
}