Звіт до pz-8 Ремез

Завдання 1

using System;

using System.Security.Cryptography;

using System.IO;

using System.Text;

namespace Task8.\_1

{

public class RSAWithRSAParameterKey

{

private RSAParameters \_publicKey;

private RSAParameters \_privateKey;

public void AssignNewKey()

{

using (var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

\_publicKey = rsa.ExportParameters(false);

\_privateKey = rsa.ExportParameters(true);

}

}

public byte[] EncryptData(byte[] dataToEncrypt)

{

byte[] cipherbytes;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_publicKey);

cipherbytes = rsa.Encrypt(dataToEncrypt, true);

}

return cipherbytes;

}

public byte[] DecryptData(byte[] dataToEncrypt)

{

byte[] plain;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_privateKey);

plain = rsa.Decrypt(dataToEncrypt, true);

}

return plain;

}

}

class Program

{

static void Main(string[] args)

{

var rsaParams = new RSAWithRSAParameterKey();

const string original = "It is encrypted message using RSA and keys";

rsaParams.AssignNewKey();

var encrypted = rsaParams.EncryptData(Encoding.UTF8.GetBytes(original));

var decrypted = rsaParams.DecryptData(encrypted);

Console.WriteLine("Main Text = " + original);

Console.WriteLine();

Console.WriteLine("Encrypted Text = " + Convert.ToBase64String(encrypted));

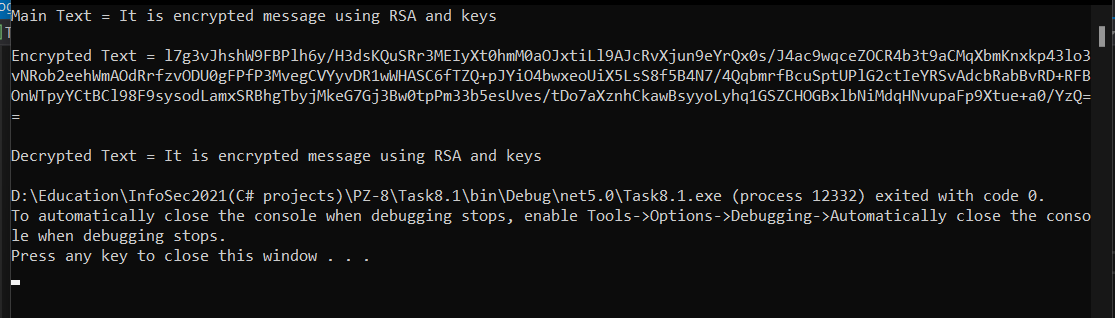
Console.WriteLine();

Console.WriteLine("Decrypted Text = " + Encoding.Default.GetString(decrypted));

}

}

}



Завдання 2

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace Task8.\_2

{

class Program

{

private readonly static string CspContainerName = "RsaContainer";

public static void AssignNewKey(string publicKeyPath)

{

CspParameters cspParameters = new CspParameters(1)

{

KeyContainerName = CspContainerName,

ProviderName = "Microsoft Strong Cryptographic Provider"

};

var rsa = new RSACryptoServiceProvider(cspParameters)

{

PersistKeyInCsp = true

};

File.WriteAllText(publicKeyPath, rsa.ToXmlString(false));

}

public static byte[] DecryptData(byte[] dataToDecrypt)

{

byte[] plainBytes;

var cspParams = new CspParameters

{

KeyContainerName = CspContainerName,

};

using (var rsa = new RSACryptoServiceProvider(cspParams))

{

rsa.PersistKeyInCsp = true;

plainBytes = rsa.Decrypt(dataToDecrypt, false);

}

return plainBytes;

}

public static byte[] EncryptData(string publicKeyPath, byte[] dataToEncrypt)

{

byte[] cipherbytes;

using (var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

rsa.FromXmlString(File.ReadAllText(publicKeyPath));

cipherbytes = rsa.Encrypt(dataToEncrypt, false);

}

return cipherbytes;

}

static void Main(string[] args)

{

string message = "this message is secret";

AssignNewKey("public.xml");

var encrypted = EncryptData("public.xml", Encoding.Unicode.GetBytes(message));

var decrypted = DecryptData(encrypted);

Console.WriteLine("Message = " + message);

Console.WriteLine();

Console.WriteLine("Encrypted message = " + Convert.ToBase64String(encrypted));

Console.WriteLine();

Console.WriteLine("Decrypted message = " + Encoding.Unicode.GetString(decrypted));

}

}

}

