Основи інформаційної безпеки

ПЗ №7

“ Асиметричне шифрування як засіб забезпечення конфіденційності інформації”

Підготував:

Студент групи МІТ-21

Міхальський Владислав

Завдання №1

using System;

using System.Security.Cryptography;

using System.Text;

namespace lab7

{

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[] cypherbytes;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_publicKey);

cypherbytes = rsa.Encrypt(dataToEncrypt, true);

}

return cypherbytes;

}

public byte[] DecryptData(byte[] dataToDecrypt)

{

byte[] plainText;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_privateKey);

plainText = rsa.Decrypt(dataToDecrypt, true);

}

return plainText;

}

}

class Program

{

static void Main(string[] args)

{

var rsaParams = new RSAWithRSAParameterKey();

const string original = "Just Password";

rsaParams.AssignNewKey();

Console.WriteLine("Original Text: " + original);

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

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

var decrypted = rsaParams.DecryptData(encrypted);

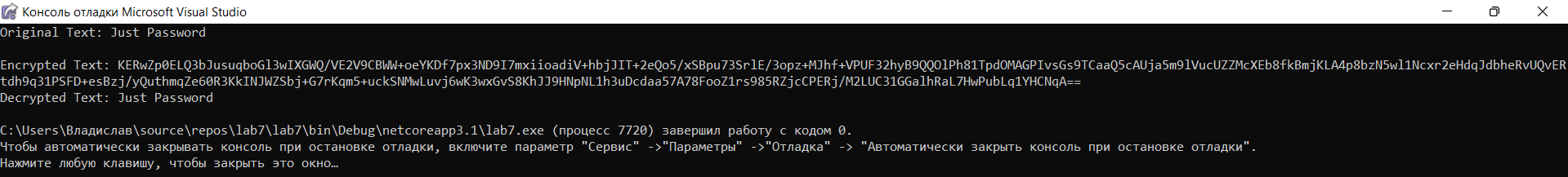
Console.WriteLine("Decrypted Text: " + Encoding.Default.GetString(decrypted));

}

}

}

Результат:



Завдання №2

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace lab7.\_2

{

class RSAWithRSAParameterKey

{

private RSAParameters \_privateKey;

public void AssignNewKeys(string publicKeyPath)

{

using (var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

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

\_privateKey = rsa.ExportParameters(true);

}

}

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

{

byte[] cypherbytes;

using (var rsa = new RSACryptoServiceProvider(2048))

{

rsa.PersistKeyInCsp = false;

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

cypherbytes = rsa.Encrypt(dataToEncrypt, true);

}

return cypherbytes;

}

public byte[] DecryptData(byte[] dataToDecrypt)

{

byte[] plainText;

using (var rsa = new RSACryptoServiceProvider())

{

rsa.PersistKeyInCsp = false;

rsa.ImportParameters(\_privateKey);

plainText = rsa.Decrypt(dataToDecrypt, true);

}

return plainText;

}

}

class Program

{

static void Main(string[] args)

{

var rsaParams = new RSAWithRSAParameterKey();

Console.WriteLine("Enter text to encrypt: ");

string original = Convert.ToString(Console.ReadLine());

Console.WriteLine("Enter desired path to public key that will be generated: ");

string publicKeyPath = Convert.ToString(Console.ReadLine());

rsaParams.AssignNewKeys(publicKeyPath);

Console.WriteLine("Original Text: " + original);

var encrypted = rsaParams.EncryptData(publicKeyPath, Encoding.Unicode.GetBytes(original));

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

var decrypted = rsaParams.DecryptData(encrypted);

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

Console.WriteLine("Want to encrypt text with someone's public key? y/n: ");

string encryptAnother = Convert.ToString(Console.ReadLine());

if (encryptAnother == "y")

{

Console.WriteLine("Enter text to encrypt: ");

string additionalOriginal = Convert.ToString(Console.ReadLine());

Console.WriteLine("Enter path to existed public key: ");

string additionalPublicKeyPath = Convert.ToString(Console.ReadLine());

Console.WriteLine("Original Text: " + additionalOriginal);

var additionalEncrypted = rsaParams.EncryptData(additionalPublicKeyPath, Encoding.Unicode.GetBytes(additionalOriginal));

Console.WriteLine("Encrypted Text: " + Convert.ToBase64String(additionalEncrypted));

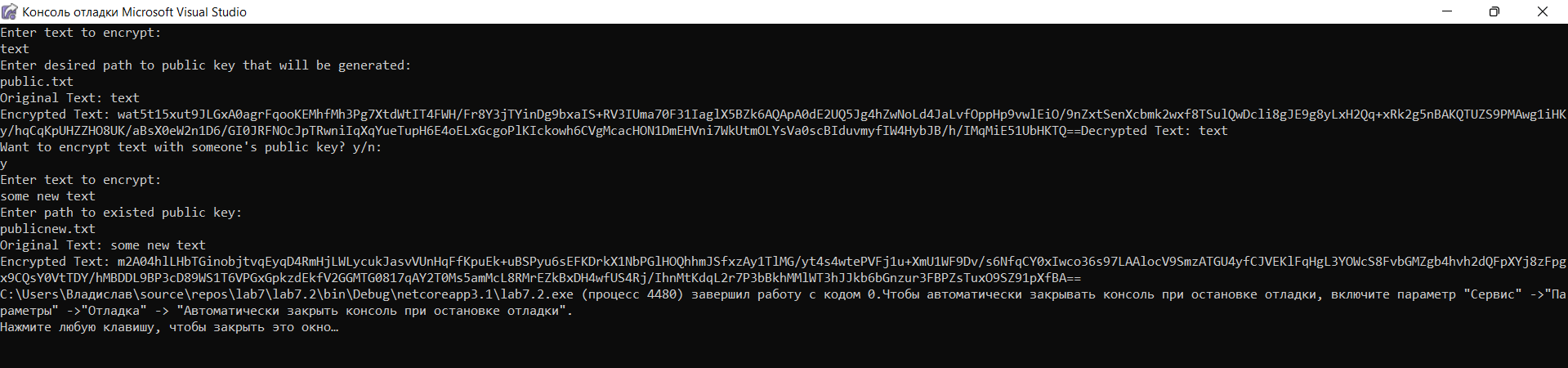
}

}

}

}

Результат:



Посилання на гітхаб репозиторій: <https://github.com/vmikhalskyi/Security-basics-2021.git>

Висновок: у ході лабораторної роботи я навчився, зашифровувати та розшифровувати паролі(або інші дані) за допомогою асиметричного шифрування. Також, зрозумів, як працює асиметричне шифрування, що таке приватні та публічні ключи. Навчився зберігати приватні та публічні ключі різними способами: просто у пам’яті комп’ютера, записувати у створений файл. Навчився зашифровувати повідомлення за допомогою публічних ключів, які зберігаються у файлі комп’ютера.