Основи інформаційної безпеки

ПЗ №6

“ Забезпечення конфіденційності інформації з використанням симетричного шифрування”

Підготував:

Студент групи МІТ-21

Міхальський Владислав

Завдання №1

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace lab6

{

public class aesChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

public class desChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, des.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, des.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

public class tripleDesChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var tripleDes = new TripleDESCryptoServiceProvider())

{

tripleDes.Mode = CipherMode.CBC;

tripleDes.Padding = PaddingMode.PKCS7;

tripleDes.Key = key;

tripleDes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, tripleDes.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var tripleDes = new DESCryptoServiceProvider())

{

tripleDes.Mode = CipherMode.CBC;

tripleDes.Padding = PaddingMode.PKCS7;

tripleDes.Key = key;

tripleDes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, tripleDes.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

class Program

{

static void Main(string[] args)

{

//TRIPLEDES

var tripleDes = new desChipher();

const string originalTripleDes = "Text";

var keyTripleDes = tripleDes.GenerateRandomNumber(8);

var ivTripleDes = tripleDes.GenerateRandomNumber(8);

var encryptedTripleDes = tripleDes.Encrypt(Encoding.UTF8.GetBytes(originalTripleDes), keyTripleDes, ivTripleDes);

var decryptedTripleDes = tripleDes.Decrypt(encryptedTripleDes, keyTripleDes, ivTripleDes);

var decryptedMessageTripleDes = Encoding.UTF8.GetString(decryptedTripleDes);

Console.WriteLine("Triple DES Encryption");

Console.WriteLine("Original Text = " + originalTripleDes);

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

Console.WriteLine("Decrypted Text = " + decryptedMessageTripleDes);

//DES

var des = new desChipher();

const string originalDes = "Text";

var keyDes = des.GenerateRandomNumber(8);

var ivDes = des.GenerateRandomNumber(8);

var encryptedDes = des.Encrypt(Encoding.UTF8.GetBytes(originalDes), keyDes, ivDes);

var decryptedDes = des.Decrypt(encryptedDes, keyDes, ivDes);

var decryptedMessageDes = Encoding.UTF8.GetString(decryptedDes);

Console.WriteLine("DES Encryption");

Console.WriteLine("Original Text = " + originalDes);

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

Console.WriteLine("Decrypted Text = " + decryptedMessageDes);

//AES

var aes = new aesChipher();

const string original = "Text";

var key = aes.GenerateRandomNumber(32);

var iv = aes.GenerateRandomNumber(16);

var encrypted = aes.Encrypt(Encoding.UTF8.GetBytes(original), key, iv);

var decrypted = aes.Decrypt(encrypted, key, iv);

var decryptedMessage = Encoding.UTF8.GetString(decrypted);

Console.WriteLine("AES Encryption");

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

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

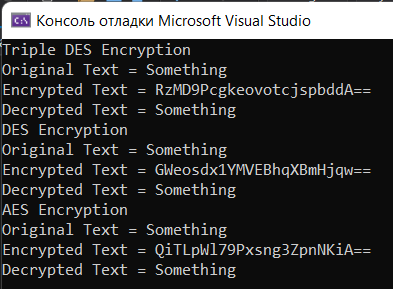
Console.WriteLine("Decrypted Text = " + decryptedMessage);

}

}

}

Результат:



Завдання №2

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace lab6

{

public class aesChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

public class desChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, des.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, des.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

public class tripleDesChipher

{

public byte[] GenerateRandomNumber(int length)

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public byte[] Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var tripleDes = new TripleDESCryptoServiceProvider())

{

tripleDes.Mode = CipherMode.CBC;

tripleDes.Padding = PaddingMode.PKCS7;

tripleDes.Key = key;

tripleDes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, tripleDes.CreateEncryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public byte[] Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var tripleDes = new DESCryptoServiceProvider())

{

tripleDes.Mode = CipherMode.CBC;

tripleDes.Padding = PaddingMode.PKCS7;

tripleDes.Key = key;

tripleDes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var CryptoStream = new CryptoStream(memoryStream, tripleDes.CreateDecryptor(), CryptoStreamMode.Write);

CryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

CryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

}

public class PBKDF2

{

public static byte[] GenerateSalt()

{

using (var randomNumberGenerator = new RNGCryptoServiceProvider())

{

var randomNumber = new byte[32];

randomNumberGenerator.GetBytes(randomNumber);

return randomNumber;

}

}

public static byte[] HashPasswordSHA256(byte[] toBeHashed, byte[] salt, int numberOfRounds, int length)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds, HashAlgorithmName.SHA256))

{

return rfc2898.GetBytes(length);

}

}

}

class Program

{

static void Main(string[] args)

{

Random randomKey = new Random(1234);

Random randomIV = new Random(5678);

int Key = randomKey.Next(100000, 1000000);

Console.WriteLine("Pseudo-random key: " + Key);

int IV = randomIV.Next(100000, 1000000);

Console.WriteLine("Pseudo-random IV: " + IV);

byte[] salt = PBKDF2.GenerateSalt();

int numberOfRounds = 180000;

//TRIPLEDES

var tripleDes = new desChipher();

const string originalTripleDes = "Text";

var keyTripleDes = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(Key), salt, numberOfRounds, 8);

var ivTripleDes = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(IV), salt, numberOfRounds, 8);

var encryptedTripleDes = tripleDes.Encrypt(Encoding.UTF8.GetBytes(originalTripleDes), keyTripleDes, ivTripleDes);

var decryptedTripleDes = tripleDes.Decrypt(encryptedTripleDes, keyTripleDes, ivTripleDes);

var decryptedMessageTripleDes = Encoding.UTF8.GetString(decryptedTripleDes);

Console.WriteLine("Triple DES Encryption");

Console.WriteLine("Original Text = " + originalTripleDes);

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

Console.WriteLine("Decrypted Text = " + decryptedMessageTripleDes);

//DES

var des = new desChipher();

const string originalDes = "Text";

var keyDes = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(Key), salt, numberOfRounds, 8);

var ivDes = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(IV), salt, numberOfRounds, 8);

var encryptedDes = des.Encrypt(Encoding.UTF8.GetBytes(originalDes), keyDes, ivDes);

var decryptedDes = des.Decrypt(encryptedDes, keyDes, ivDes);

var decryptedMessageDes = Encoding.UTF8.GetString(decryptedDes);

Console.WriteLine("DES Encryption");

Console.WriteLine("Original Text = " + originalDes);

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

Console.WriteLine("Decrypted Text = " + decryptedMessageDes);

//AES

var aes = new aesChipher();

const string original = "Text";

var key = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(Key), salt, numberOfRounds, 32);

var iv = PBKDF2.HashPasswordSHA256(BitConverter.GetBytes(IV), salt, numberOfRounds, 16);

var encrypted = aes.Encrypt(Encoding.UTF8.GetBytes(original), key, iv);

var decrypted = aes.Decrypt(encrypted, key, iv);

var decryptedMessage = Encoding.UTF8.GetString(decrypted);

Console.WriteLine("AES Encryption");

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

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

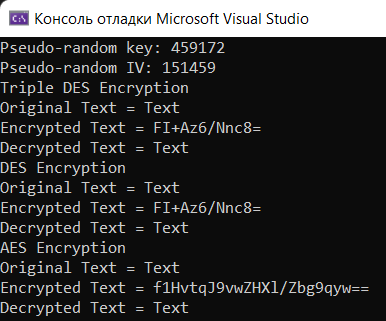
Console.WriteLine("Decrypted Text = " + decryptedMessage);

}

}

}

Результат:



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

Висновок: у ході лабораторної роботи я навчився, шифрувати паролі за допомогою різних алгоритмів симетричного шифрування, таких як: AES, DES, TripleDES. Також, зробив хешування з сіллю 180000 тисяч разів ключа та вектора ініціалізації, а потім шифрування за допомогою них пароля.