Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 5**

**з дисципліни «Безпека програм та даних»**

**«Створення програмного засобу для цифрового підпису інформації з використанням Microsoft CryptoAPI»**

**Лектор:**

доц. кафедри ПЗ

Сенів М. М.

**Виконав:**

студ. групи ПІ-41

Павленчик М. М.

**Прийняв:**

доц. кафедри ПЗ

Сенів М. М.

«\_\_\_» \_\_\_\_\_\_ 2018 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**ТЕМА РОБОТИ**: Створення програмного засобу для цифрового підпису інформації з використанням Microsoft CryptoAPI.

**МЕТА РОБОТИ**: Ознайомитись з методами криптографічного забезпечення цифрового підпису, навчитись створювати програмні засоби для цифрового підпису з використанням криптографічних інтерфейсів.

### Теоретичні відомості

У ситуації, коли обидві сторони не довіряють один одному, необхідно щось більше, ніж аутентифікація на основі спільного секрету. Можливим рішенням подібної проблеми є використання цифрового підпису. Цифровий підпис повинен володіти наступними властивостями:

1. Повинна бути можливість перевірити автора, дату й час створення підпису.
2. Повинна бути можливість встановити достовірність вмісту повідомлення на час створення підпису.
3. Повинна бути можливість перевірки підпису третьою стороною для вирішення суперечок.

Таким чином, функція цифрового підпису включає, зокрема, і функцію аутентифікації.

На підставі цих властивостей можна сформулювати наступні вимоги до цифрового підпису:

1. Підпис повинен бути бітовим зразком, який залежить від повідомлення, що підписується.
2. Підпис повинен використовувати деяку унікальну інформацію відправника для запобігання підробки або відмови.
3. Створювати цифровий підпис повинно бути відносно легко.
4. Повинно бути обчислювально неможливо підробити цифровий підпис як створенням нового повідомлення для існуючого цифрового підпису, так і створенням фальшивого цифрового підпису для деякого повідомлення.
5. Цифровий підпис повинен бути досить компактним і не займати багато пам'яті.

Національний інститут стандартів і технології США (NIST) розробив федеральний стандарт цифрового підпису DSS. Для створення цифрового підпису використовується алгоритм DSA (Digital Signature Algorithm). В якості хеш-алгоритму стандарт передбачає використання SHA-1 (Secure Hash Algorithm). DSS спочатку був запропонований в 1991 році й переглянутий в 1993 році у відповідь на публікації, що стосуються безпеки його схеми. У 1996 році в нього були внесені незначні зміни.

DSS використовує алгоритм, що розроблявся для використання тільки в якості цифрового підпису. На відміну від алгоритму RSA, його не можна використовувати для шифрування чи обміну ключами.

**Завдання для лабораторної роботи**

З використання функцій CryptoAPI створити прикладну програму для створення і перевірки цифрового підпису за стандартом DSS. Програмна реалізація повинна виводити значення підпису як для рядка, заданого в полі вводу, так і для файлу. Результат роботи програми повинен відображатись на екрані з можливістю наступного запису в файл. Крім того програма повинна мати можливість перевірити цифровий підпис будь-якого файлу за наявним файлом підпису, записаним у шістнадцятковому форматі. У звіті навести протокол роботи програми та зробити висновки.

### Результати

Протокол роботи програми



Рис. 1. Початковий інтерфейс застосунку

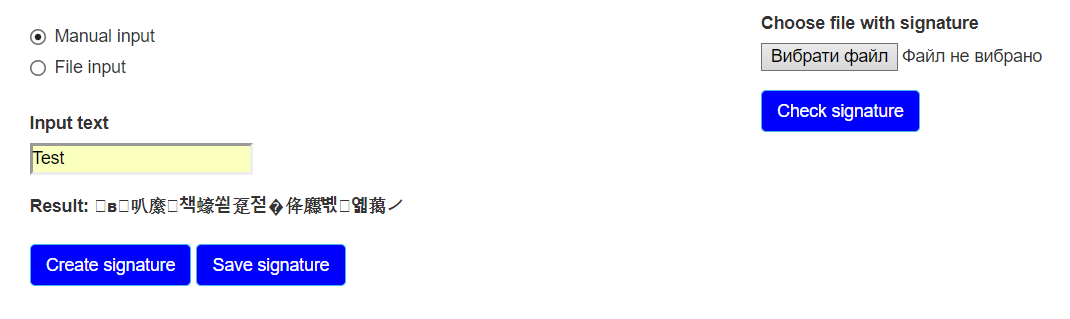


Рис. 2. Результати створення підпису для введеного тексту

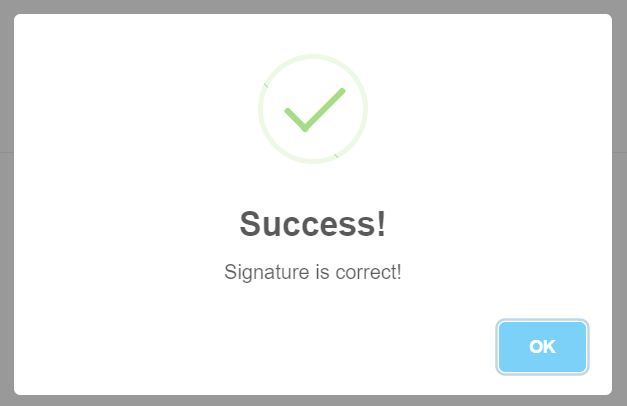


Рис. 3. Результати перевірки підпису

Вихідний код програми

public class DSAService

{

public const string PublicKeyFile = "PublicKey";

public const string PrivateKeyFile = "PrivateKey";

private DSACryptoServiceProvider Provider { get; set; } = new DSACryptoServiceProvider();

private FileService FileService { get; set; } = new FileService();

public KeysModel GenerateKeys()

{

Provider = new DSACryptoServiceProvider();

var publicKey = Provider.ExportCspBlob(false);

var privateKey = Provider.ExportCspBlob(true);

return new KeysModel { PublicKey = publicKey, PrivateKey = privateKey };

}

public byte[] SignData(byte[] input)

{

var privateKey = FileService.LoadFile(PrivateKeyFile);

Provider.ImportCspBlob(privateKey);

return Provider.SignData(input);

}

public bool CheckSign(byte[] input, byte[] sign)

{

var publicKey = FileService.LoadFile(PublicKeyFile);

Provider.ImportCspBlob(publicKey);

return Provider.VerifyData(input, sign);

}

}

public class HomeController : Controller

{

public FileService FileService { get; set; } = new FileService();

public DSAService DSAService { get; set; } = new DSAService();

public ActionResult Index()

{

var rsaKeys = DSAService.GenerateKeys();

FileService.SaveFile(DSAService.PublicKeyFile, rsaKeys.PublicKey);

FileService.SaveFile(DSAService.PrivateKeyFile, rsaKeys.PrivateKey);

var model = new InputVewModel { IsManualInput = true };

return View(model);

}

[HttpPost]

public JsonResult SignData(InputVewModel input)

{

try

{

var inputData = input.IsManualInput

? Encoding.Unicode.GetBytes(input.InputText)

: FileService.LoadFile(input.FileInput);

var signature = DSAService.SignData(inputData);

var successMessage = $"Signature created successfully.";

return Json(new {

Success = true,

SuccessMessage = successMessage,

Result = Encoding.Unicode.GetString(signature)

}, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult SaveSign(InputVewModel input)

{

try

{

if (input.IsManualInput)

{

var inputData = Encoding.Unicode.GetBytes(input.InputText);

var signature = DSAService.SignData(inputData);

FileService.SaveFileWithSuffix("ManualInput.txt", "\_signature", signature);

}

else

{

var inputData = FileService.LoadFile(input.FileInput);

var signature = DSAService.SignData(inputData);

FileService.SaveFileWithSuffix(input.FileInput, "\_signature", signature);

}

var successMessage = $"Signature saved successfully.";

return Json(new

{

Success = true,

SuccessMessage = successMessage,

}, JsonRequestBehavior.AllowGet);

}

catch (Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult CheckSign(InputVewModel input)

{

try

{

var inputData = input.IsManualInput

? Encoding.Unicode.GetBytes(input.InputText)

: FileService.LoadFile(input.FileInput);

var inputDataSignature = FileService.LoadFile(input.FileInputSignature);

if (DSAService.CheckSign(inputData, inputDataSignature))

{

var successMessage = $"Signature is correct!";

return Json(new { Success = true, SuccessMessage = successMessage }, JsonRequestBehavior.AllowGet);

}

else

{

var errorMessage = $"Signature is incorrect!";

return Json(new { Success = false, ErrorMessage = errorMessage }, JsonRequestBehavior.AllowGet);

}

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

}

**Висновки**

На даній лабораторній роботі я ознайомився з методами криптографічного забезпечення цифрового підпису та навчився створювати програмні засоби для цифрового підпису з використанням криптографічних інтерфейсів.