МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи №5
З дисципліни: *"Безпека програм та даних"*
На тему: *"Створення програмного засобу для цифрового підпису інформації з використанням microsoft cryptoapi"*

Лектор:
доцент каф. ПЗ
Сенів М. М.

Виконав:
ст. гр. ПЗ-43
Лесневич Є. Є.

Прийняв:
ст. викладач каф. ПЗ
Угриновський Б. В.

«_____»_____2024 р.

∑=_____ .....

Львів – 2024

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

Аутентифікація захищає двох учасників, які обмінюються повідомленнями, від впливу деякої третьої сторони. Однак проста аутентифікація не захищає учасників один від одного, тоді як і між ними теж можуть виникати певні форми суперечностей.

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.
    a. 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.

Завдання до виконання 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.

Код аглоритму

```
using Lab01GUI.Services.Interfaces;
using System.Net;
using System.Security.Cryptography;
using System.Text;

namespace Lab01GUI.Services.Implementation;

public class DSSService : IDSSService
{
	private int KeySize = 2048;

	public void SetKeySize(int keySize)
	 => KeySize = keySize;

	public int GetKeySize()
		=> KeySize;

	public DSSKeyResult GetKeyResult()
	{
		using var dsa = new DSACryptoServiceProvider(KeySize);

		var privateKey = dsa.ToXmlString(true);
		var publicKey = dsa.ToXmlString(false);

		return new(publicKey, privateKey);
	}

	public byte[] Sign(string data, string privateKey)
	{
		return Sign(Encoding.UTF8.GetBytes(data), privateKey);
	}

	public async Task<byte[]> Sign(IFormFile file, string privateKey)
	{
		return Sign(await GetFileBytesAsync(file), privateKey);
	}

	public byte[] Sign(byte[] data, string privateKey)
	{
		using var dsa = new DSACryptoServiceProvider();
		privateKey = WebUtility.HtmlDecode(privateKey);
		dsa.FromXmlString(privateKey);

		return dsa.SignData(data);
	}

	public async Task<bool> Verify(string data, IFormFile SignatureFile, string publicKey)
	{
		var signatureBytes = await GetFileBytesAsync(SignatureFile);

		return Verify(Encoding.UTF8.GetBytes(data), signatureBytes, publicKey);
	}

	public async Task<bool> Verify(IFormFile file, IFormFile SignatureFile, string publicKey)
	{
		publicKey = WebUtility.HtmlDecode(publicKey);

		var fileData = await GetFileBytesAsync(file);
		var signatureBytes = await GetFileBytesAsync(SignatureFile);

		return Verify(fileData, signatureBytes, publicKey);
	}

	public bool Verify(byte[] data, byte[] signature, string publicKey)
	{
		publicKey = WebUtility.HtmlDecode(publicKey);

		using var dsa = new DSACryptoServiceProvider();
		dsa.FromXmlString(publicKey);

		return dsa.VerifyData(data, signature);
	}

	public async Task<byte[]> Sign(string data, IFormFile privateKey)
	{
		return Sign(data, await GetFileTextAsync(privateKey));
	}

	public async Task<byte[]> Sign(IFormFile file, IFormFile privateKey)
	{
		return await Sign(file, await GetFileTextAsync(privateKey));
	}

	public async Task<bool> Verify(string data, IFormFile SignatureFile, IFormFile publicKey)
	{
		return await Verify(data, SignatureFile, await GetFileTextAsync(publicKey));
	}

	public async Task<bool> Verify(IFormFile file, IFormFile SignatureFile, IFormFile publicKey)
	{
		return await Verify(file, SignatureFile, await GetFileTextAsync(publicKey));
	}
```

```
        public async Task<bool> Verify(string data, string SignatureText, IFormFile publicKey)
        {
            return Verify(Encoding.UTF8.GetBytes(data), Convert.FromHexString(SignatureText), await GetFileTextAsync(publicKey));
        }

        public async Task<bool> Verify(IFormFile file, string SignatureText, IFormFile publicKey)
        {
            return Verify(await GetFileBytesAsync(file), Convert.FromHexString(SignatureText), await GetFileTextAsync(publicKey));
        }

        private static async Task<byte[]> GetFileBytesAsync(IFormFile file)
        {
            using var stream = file.OpenReadStream();
            using var memoryStream = new MemoryStream();
            await stream.CopyToAsync(memoryStream);
            return memoryStream.ToArray();
        }

        private static async Task<string> GetFileTextAsync(IFormFile file)
        {
            using var stream = file.OpenReadStream();
            using var reader = new StreamReader(stream);
            return WebUtility.HtmlDecode(await reader.ReadToEndAsync());
        }
}

public record DSSKeyResult(string PublicKey, string PrivateKey);
```

Результати роботи

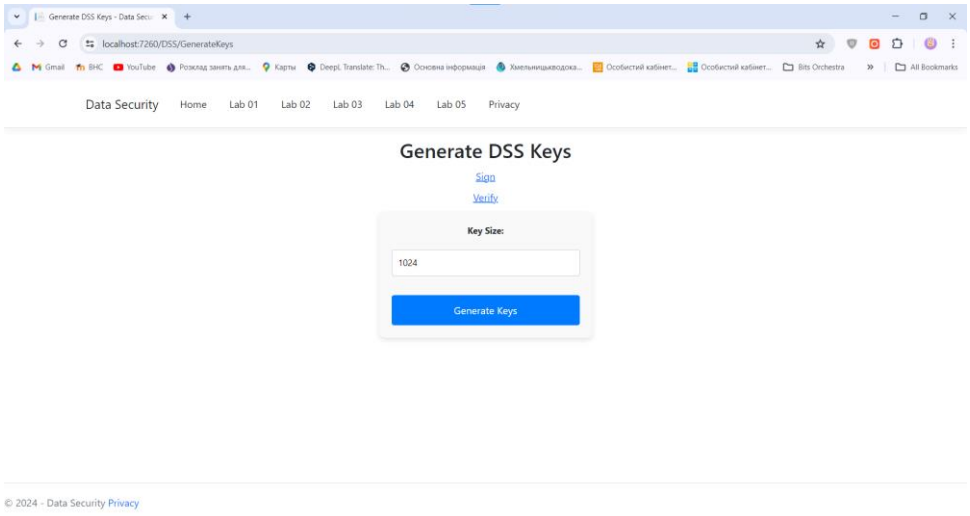

Рис. 1 Головне вікно програми

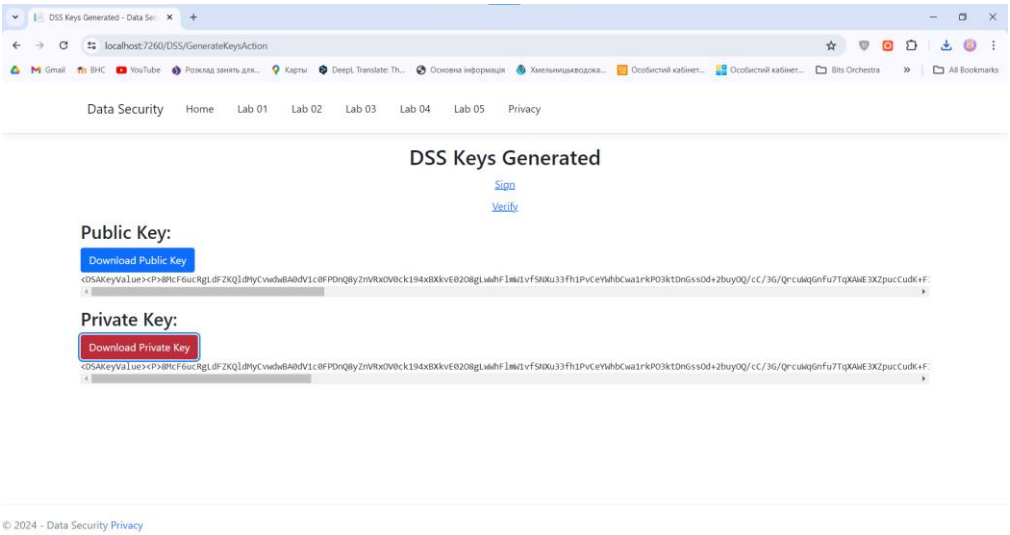

Рис. 2 Згенеровані ключі

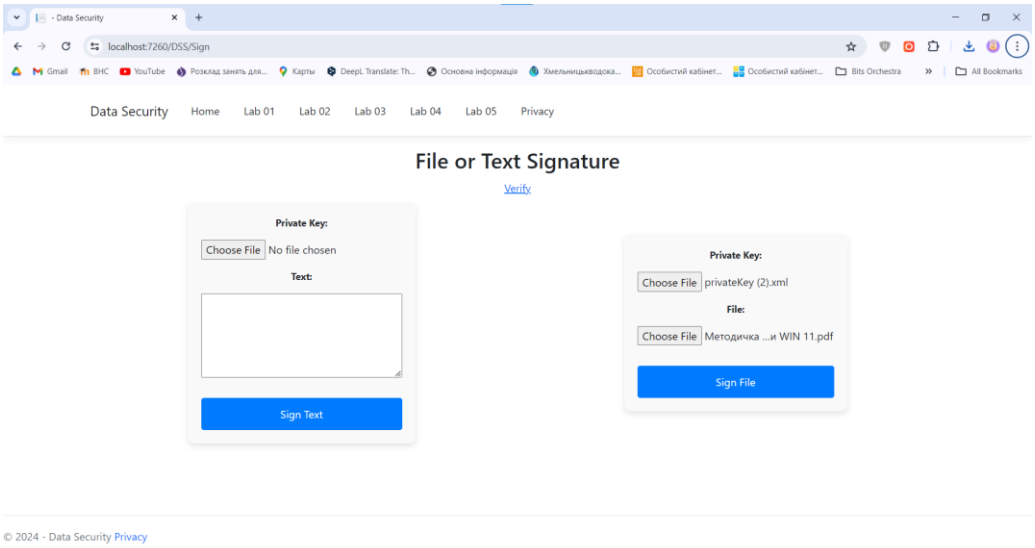

Рис. 3 Цифровий підпис файлу

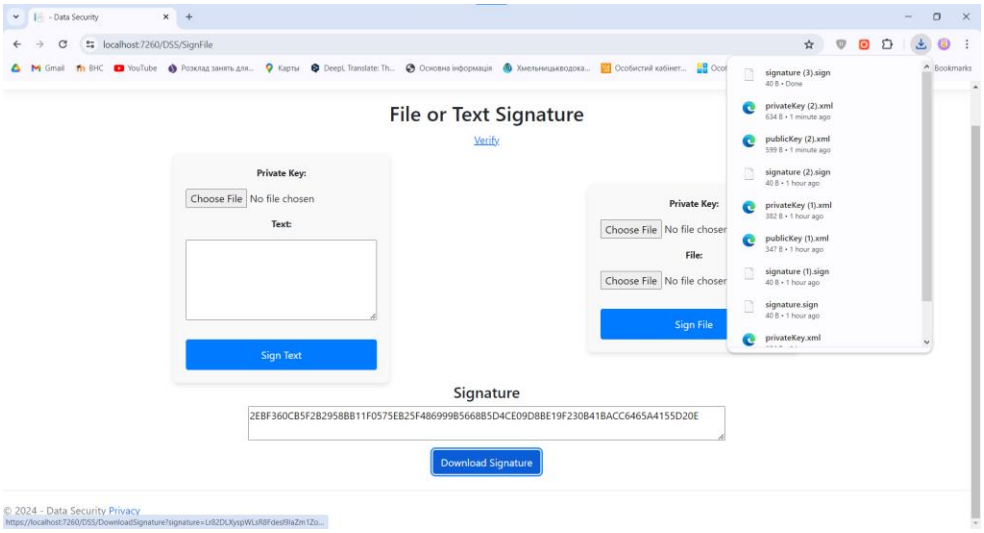

Рис. 4 Результат цифрового підпису файлу

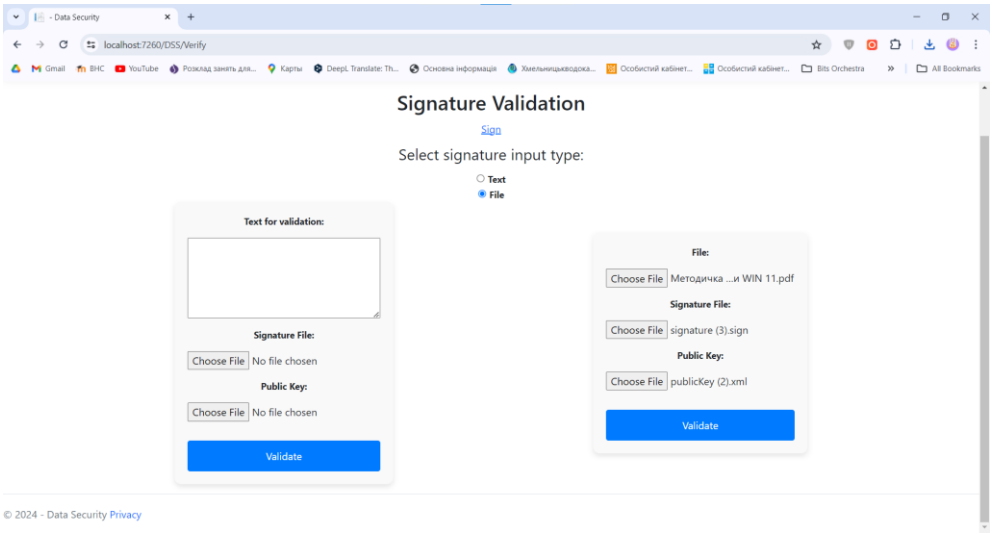

Рис. 5 Перевірка цифрового підпису файлу

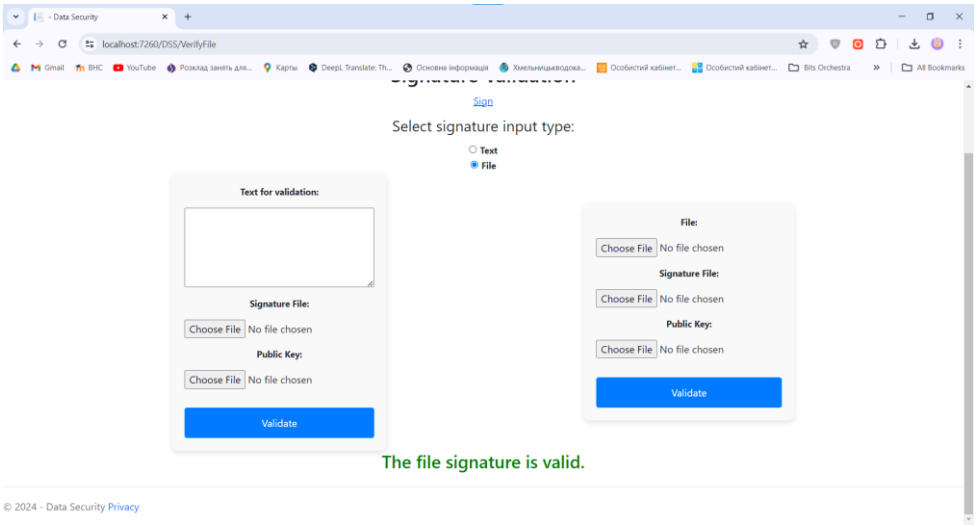

Рис. 6 Результат перевірки цифрового підпису файлу

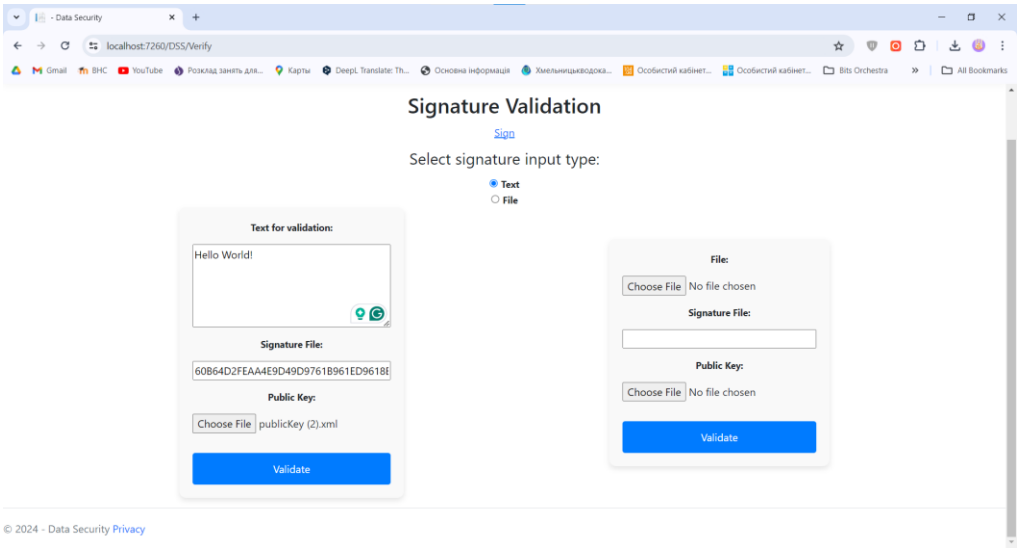

Рис. 7 Перевірка цифрового підпису тексту

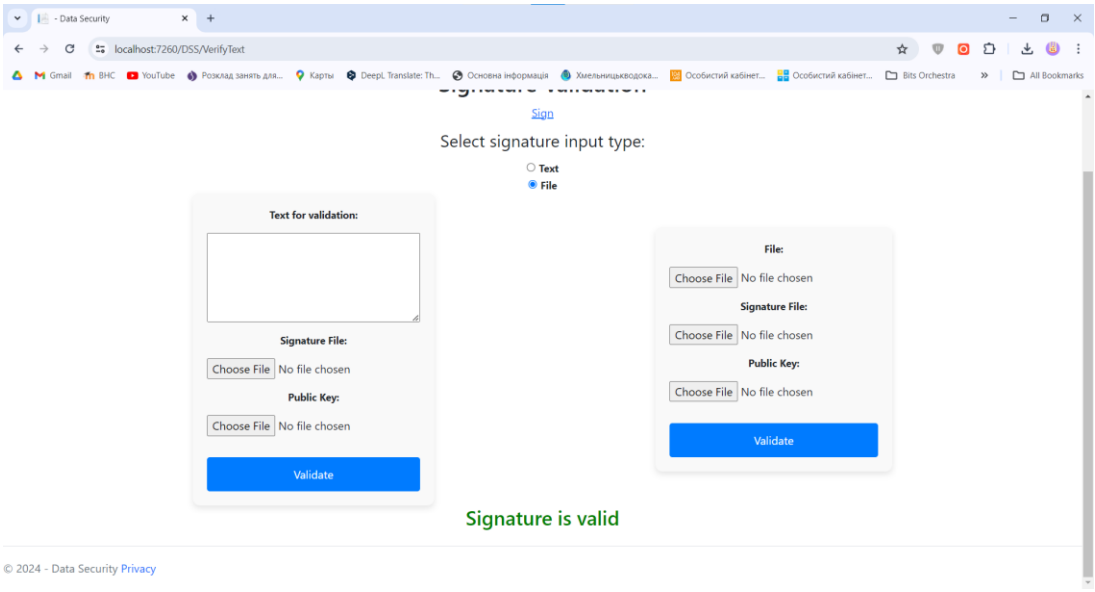

Рис. 8 Результат перевірки цифрового підпису тексту

Висновки

Отже, під час виконання даної лабораторної роботи я ознайомився з методами криптографічного забезпечення цифрового підпису, навчився створювати програмні засоби для цифрового підпису з використанням криптографічних інтерфейсів.