***Практичне заняття №9-10***

*«*Асиметричне шифрування як засіб забезпечення конфіденційності інформації*»*

Виконала студентка групи МІТ-21 **Борук Дарина**

Мета: розібратися із електронним цифровим підписом, його використанням для підпису документів та перевірка. Написати програму, яка генерує електронний цифровий підпис з використанням асиметричного шифрування RSA та його перевірку. Відкритий ключ береться із XML-файлу, а секретний – з контейнеру.

Хід роботи:

* Для того, щоб написати дану програму, спочатку треба розібратись із тим, як працює алгоритм генерації і підпис ЕЦП. Тож, спочатку ми пропишемо усі функції, які будуть необхідні для вищеперерахованих задач. Такими функціями будуть:

1. Генерація ключів, використовуючи асиметричне шифрування RSA і запис їх у файл .
2. Безпосередня генерація електронного цифрового підпису.
3. Перевірка підпису.

Для функції, що генерує пару ключів ми використаємо вже відому нам функцію при асиметричному шифруванні, яка зберігає секретний ключ у контейнері (це визначає рядок коду rsa.PersistKeyInCsp = true), а публічний у файл з розширенням xml (File.WriteAllText(publicKeyPath, rsa.ToXmlString(false))). Параметром даної функції буде виступати шлях, у якому буде створено файл із записаним публічним ключем. Окрім цих команд, у функції буде прописаний екземпляр класу CspParameters, який виконує роль так званого криптографічного провайдера. Функції генерації ключів ми поміщуємо у створений екземпляр RSA, використовуючи у якості параметрів створений екземпляр класу CspParameters.

Після того, як ключі були згенеровані і збережені у відповідні їм місця, справа переходить до безпосереднього підписування електронним підписом. Передаватись у функцію буде масив байтів даних, які потрібно підписати. У цій функції ми прописуємо певні параметри для нашого контейнера і створюємо об’єкт класу RSACryptoServiceProvider із використанням параметрів класу CspParameters. На цьому етапі ми прописуємо такі значення:

* rsa.PersistKeyInCsp = false – означає, що ключ вже не зберігається у контейнері);
* var rsaFormatter = new RSAPKCS1SignatureFormatter(rsa) – клас для створення цифрового підпису на основі RSA шифруванні;
* rsaFormatter.SetHashAlgorithm(nameof(SHA256)) – встановлює алгоритм хешування, який буде використовуватись у процесі створення підпису (у даній роботі було обрано алгоритм SHA256, так як він є стандартним і має переваги над деякими іншими алгоритмами (такими як MD5));
* return rsaFormatter.CreateSignature(data) – повертає на вихід зашифрований ЕЦП.

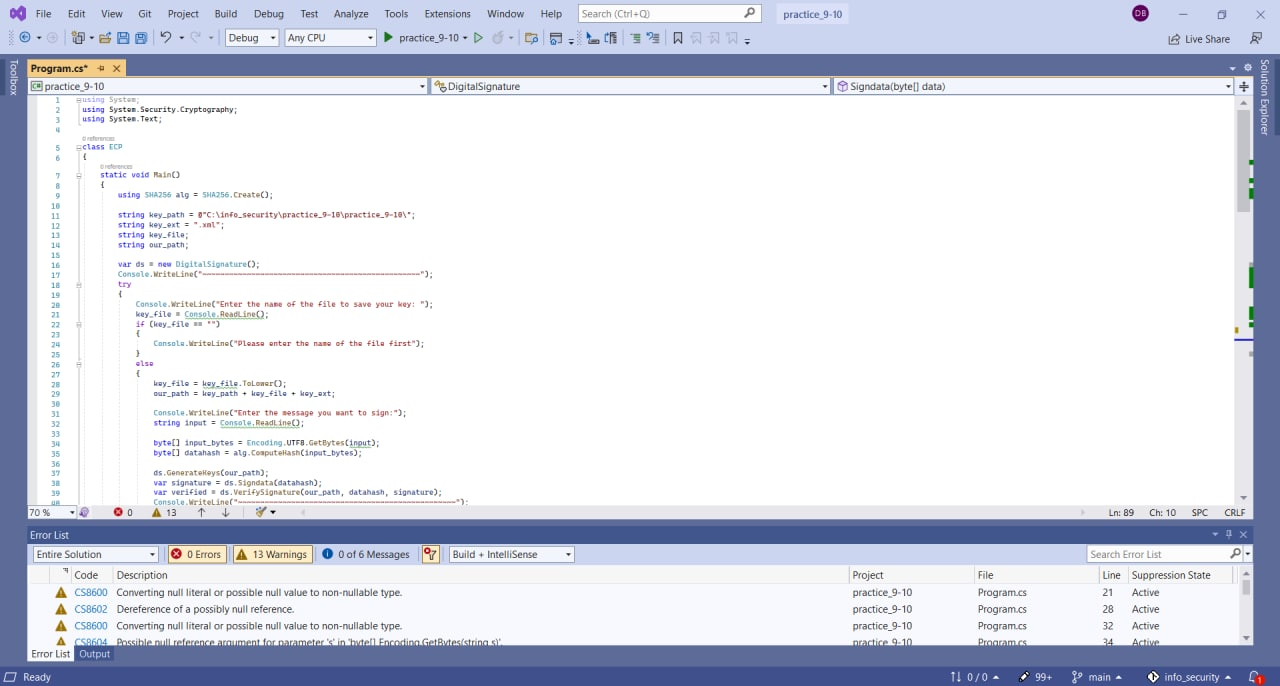
Розібравшись із головними функціями для створення електронного цифрового підпису, можна перейти до написання функції перевірки його валідності. Ця функція є схожою до попередньої, єдиними відмінностями є те, що ми використовуємо не секретний ключ. а публічний, який знаходиться в нас у файлі (rsa.FromXmlString(File.ReadAllText(publicKeyPath)) і те, що замість створення цифрового підпису (RSAPKCS1SignatureFormatter), функція перевіряє цифровий підпис з використанням алгоритму RSA (RSAPKCS1SignatureDeformatter). Далі для самої перевірки валідності підпису ми викликаємо метод VerifySignature, який повертає булеве значення true або false в залежності від результату порівняння хешу даних і створеного підпису. Параметри функції – шлях, хеш даних для підпису і згенерований підпис.

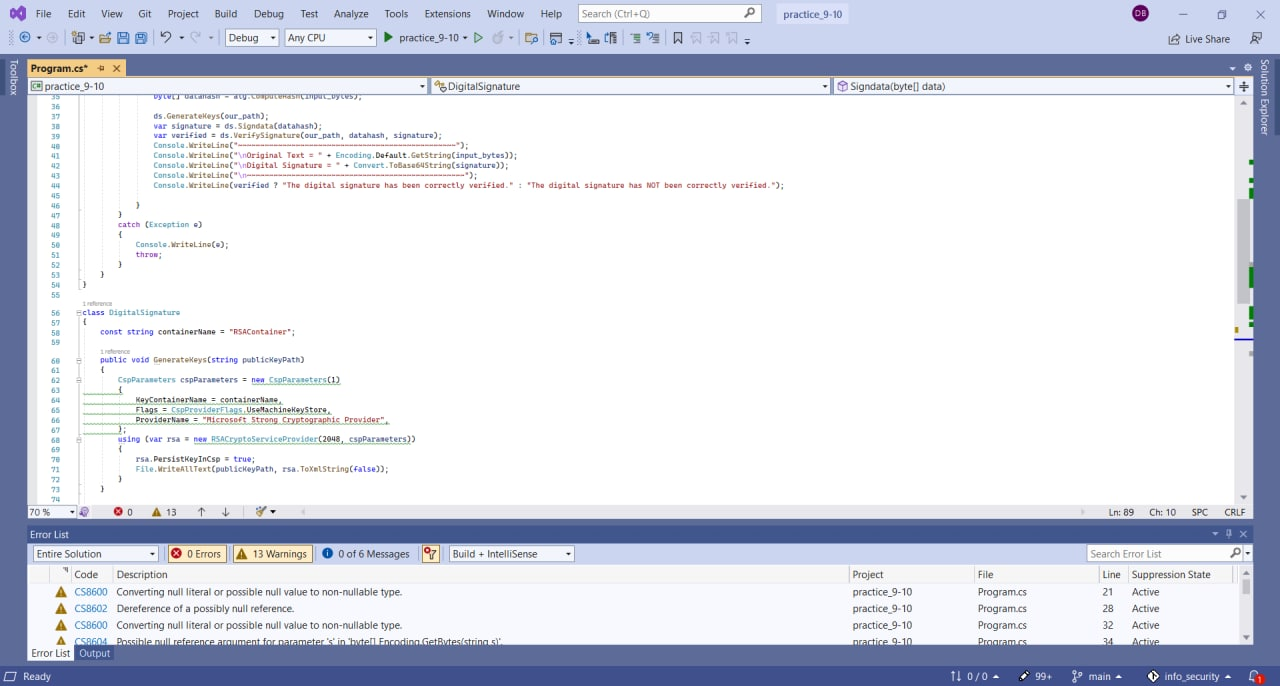
* Написавши функції для програми і розібравшись з тим як вони працюють, можна переходити до написання інтерфейсу користувача із залученням використання наших функцій.

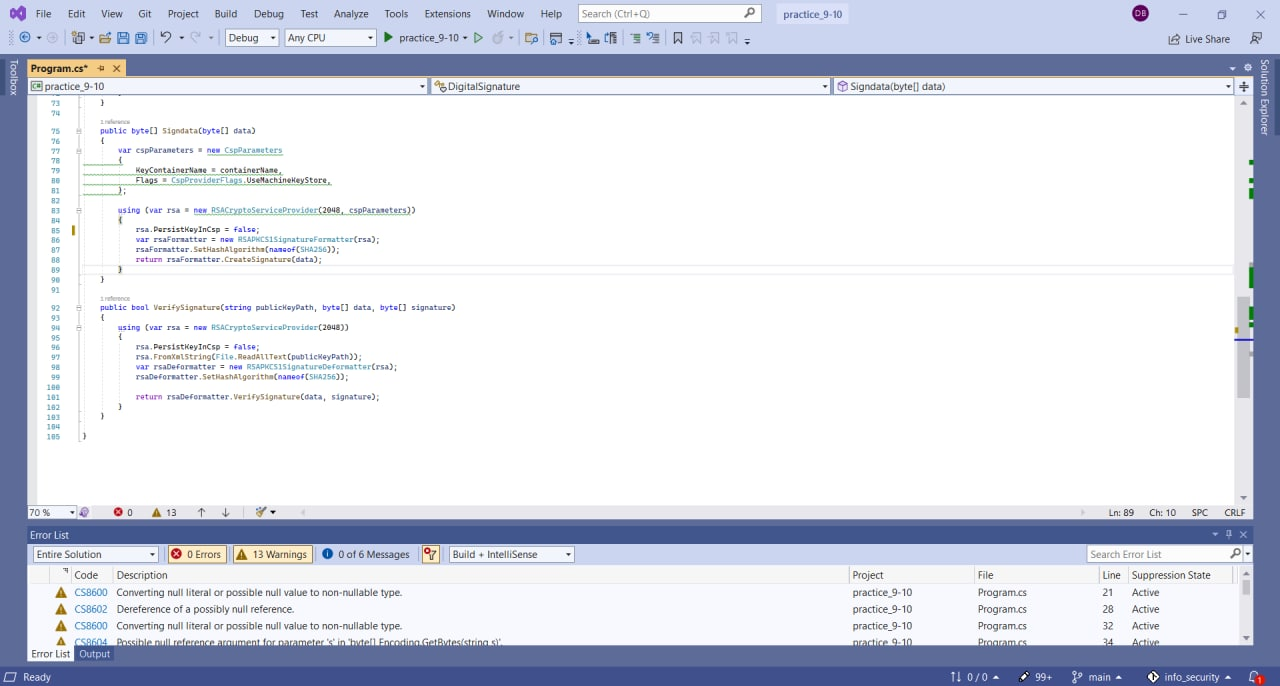
Для початку прописуються змінні для несення і збереження певної інформації: змінна зі шляхом до папки, у яку буде записано файл з публічним ключем, змінна з розширенням даного файлу і дві змінні, які будуть відповідати для створення повного шляху до файлу з ім'ям, яке введе користувач. Перше, що робить користувач – вводить назву файлу, у який буде занесено дані ЕЦП. Далі створюється шлях: шлях до папки + назва файлу + розширення, який буде передаватись у параметри наших функцій. Наступним кроком користувач вводить повідомлення, яке хоче підписати. Це повідомлення потім представляють масивом байтів і піддають хешуванню, використовуючи алгоритм хешування SHA256. Створивши об'єкт класу з прописаними функції, першим ділом ми викличемо функцію генерації ключів. Передаємо у параметр наш створений шлях. Після цього генеруємо підпис задаючи новому об’єкту результуюче значення функції із параметром хешу нашого повідомлення. Далі створимо об'єкт, у який буде занесено результат перевірки підпису. У кінці програма виведе всю необхідну інформацію користувачеві на екран:

1. Оригінальне повідомлення.
2. Електронний цифровий підпис.
3. Висновок щодо валідності даного підпису.

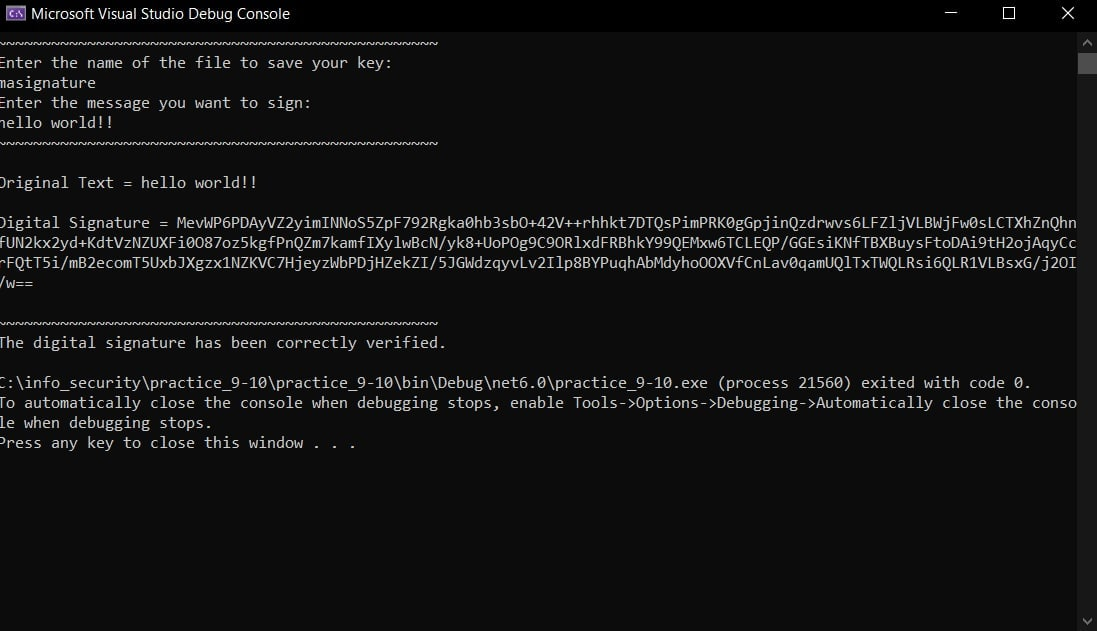
У результаті маємо даний код:



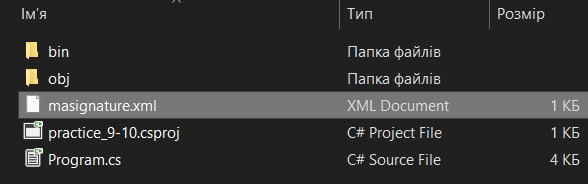




І результат виконання програми:



Було створено файл для збереження публічного ключа під назвою masignature:



Висновки: у ході практичної роботи були отримані знання про електронний цифровий підпис, його вимоги. Також було досліджено етапи підпису документів за участі ЕЦП і його перевірка.