***Практичне заняття №6***

*“Асиметричне шифрування як засіб забезпечення конфіденційності інформації»*

Виконав студент групи МІТ-21 Черевач Юрій

Мета: розібратися із механізмами асиметричного шифрування і також реалізувати їх у програмі

**Хід роботи**

Завдання:

1. Написати програму, яка виконує зашифровування та розшифровування даних з використанням алгоритмів асиметричного шифрування RSA. Пара ключів зберігається у пам’яті.
2. Для програми з п.1. реалізувати можливість збереження відкритого ключа у файлі. Реалізувати можливість зашифровувати повідомлення за допомогою файлів відкритих ключів інших користувачів.

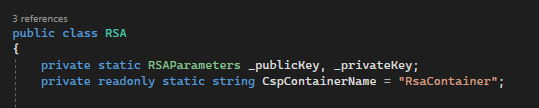
Виконання:

1. Створюємо нове рішення у VS, називаємо його pz6. Для виконання 1 завдання створюємо відповідно проєкт pz6\_1. Всередині проєкту маємо два файли з кодом - RSA i Program. В першому прописаний алгоритм асиметричного шифрування, який я застосую для виконання завдання, а в другому файлі - власне, сама програма з меню.

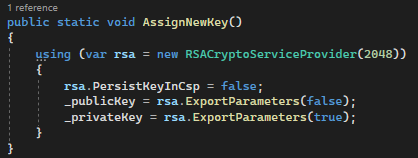
У файлі RSA я реалізовував все так, як було вказано в методичних рекомендаціях. Для пов’язаності об’єктів в проєкті я використовував namespace pz6\_1.

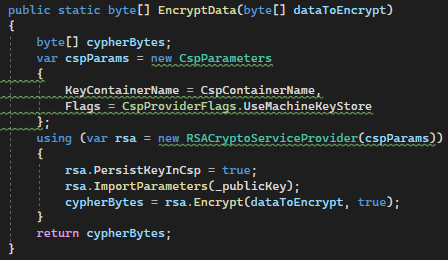
Я створив клас RSA, приватний і публічний ключі, а також контейнер для їхнього зберігання.

Виглядає це все ось так:

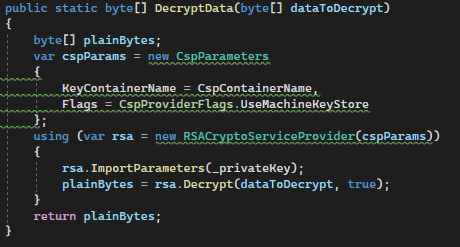


Далі в мене йдуть по-стандарту функції створення нових ключів, шифрування, дешифрування.

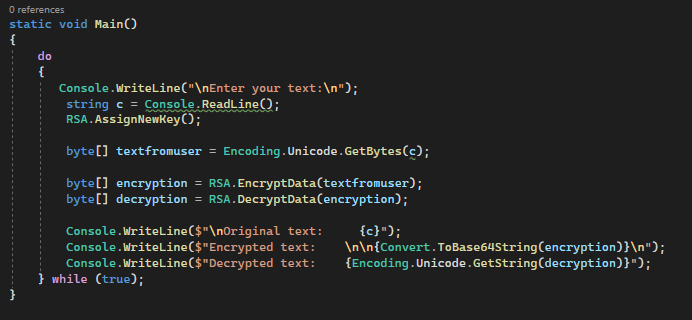




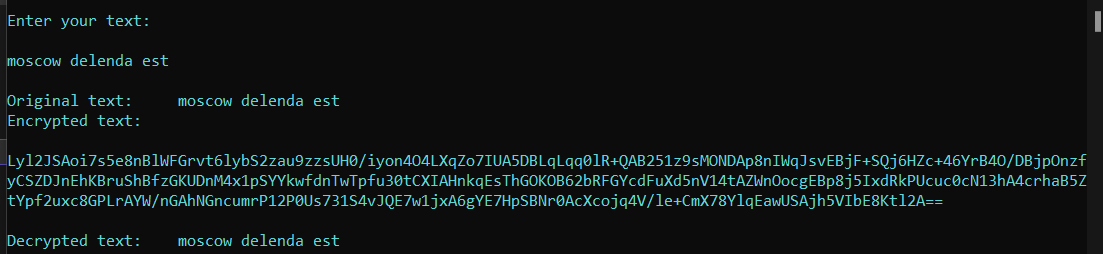
Як бачимо, функція повертає сукупність байтів - це наш зашифрований текст. Шифрується він за допомогою публічного ключа - це прописано у рядку **rsa.ImportParameters(\_publicKey);.** Функція дешифрування дуже схожа до цієї, але повертає вона уже дешифровану інформацію, яка отримується уже за допомогою приватного ключа. Виглядає вона ось так:



У файлі з менюшкою ми підключаємо наш простір імен, щоби мати змогу користуватися кодом із файлу RSA. Програма максимально проста - вона просить від користувача текст, а потім зашифровує і дешифрує його. Код меню:

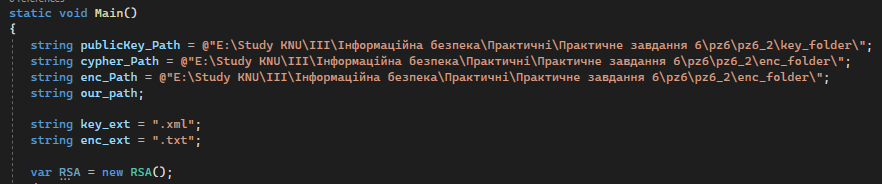


В роботі все виглядає ось так:

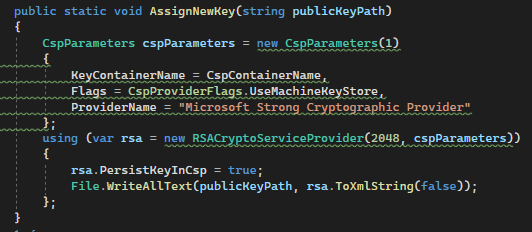


Як бачимо, трішки перефразований відомий вислів “**Carthago delenda est**” **moscow delenda est** було зашифровано у рядок хешу, а потім дешифровано. Всі умови виконані, завдання 1 також.

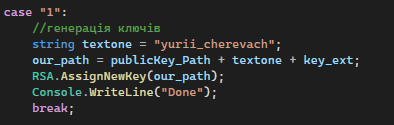
1. Для зручності я прописую шляхи, де будуть зберігатися ключ, текст, а також розширення файлів:



У завданні у нас просять зробити функцію зберігання ключа. Що ж, я реалізував це наступним чином, трохи переінакшивши функцію створення ключів:

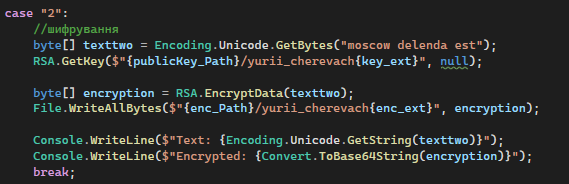


Ми бачимо функцію створення публічного ключа, який буде зберігатися у .xml форматі. В меню вона прописана ось так:



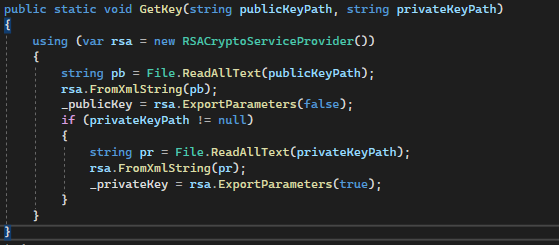
Тобто, ми бачимо формування шляху до файлу, який матиме назву name\_surname і запускаємо функцію генерації ключів, описану вище.

Далі, у нас є функція шифрування.



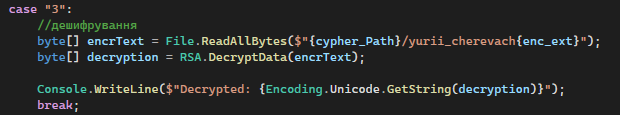
Шифруємо уже знайому фразу **moscow delenda est**. Використовуємо функцію GetKey, де вказуємо шлях до нашого публічного ключа, який ми створили у попередньому завданні, а замість аргумента, що потребує шлях до приватного ключа, вказуємо null.

Код самої функції GetKey:

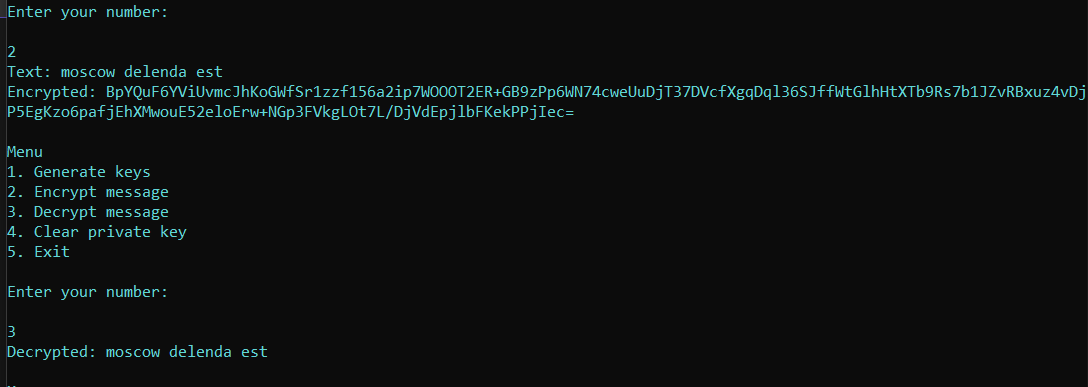


Далі, наш зашифрований текст зберігається у .txt файлі, і на екран виводиться хеш тексту.

Функція дешифрування працює схожим чином, тільки займається розшифруванням цього текстового файлу.



В роботі все виглядає так:



**Висновок**: виконуючи практичне завдання, я розібрався із алгоритмами асиметричного шифрування, а також реалізував їх у програмі.