LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

Assignment 4

Encryption and Decryption tool

Objective: Create a GUI tool in WPF that takes in a message and encrypts the text into ciphertext and decrypts the ciphertext back into its original text. The application will have two modes, one for AES (shared key and iv) and RSA (private and public key). The user can also use custom keys by importing and exporting them using the tool.

Gettings Started:

- Create a WPF .NET Core App and called it Assignment4
- Create a class called Crypto
- Create a Rename the MainWindow to EncryptionToolWindow
- Add new window and name it KeysWindow
- Add new window and name it SelectionWindow

Crypto is a wrapper class that will handle both RSA and AES encryption.

Crypto.cs.

- public enum CryptoAlgorithm {RSA, AES};
- private members: AesCryptoServiceProvider aes; RsaCryproServiceProvider rsa, CryptoAlgorithm mode
- public Initialize(CryptoAlgorithm)
    - sets the CryptoAlgorithm mode
    - Initialize the selected CryptoServiceProvider i.e. rsa or aes
    - Note when we initialize it, it automatically generated keys
- public void SaveK1(string path)
    - rsa mode: saves the private key (XML)
        - by writing the private key xml string to a file
        - Hint: rsa.ToXmlString(false) returns the private key xml as a string
        - Hint: File.WriteAllText(path, ...) writes all the text to a file
    - aes mode: saves the shared key (BINARY)
        - by writing the shared key in byte[] to a file
        - Hint: File.WriteAllBytes(path, ...); write a byte array to a file
        - Hint: aes.Key is the property to access the shared key in byte[]
        - Note: the SharedKey and IV are saved as binary files, you will need to call File.WriteAllBytes(...) and not StreamWriter because StreamWriter will not write byte[] properly as it'll call ToString to it resulting in a System.Byte[] in the file instead of the the actual bytes.
- public void SaveK2(string path)

VGP 232 Game Tools and Pipeline

    - rsa mode: saves the public key (XML)
        - Hint: rsa.ToXmlString(true) returns the public key xml as a string
    - aes mode: saves the iv (BINARY) i.e. aes
        - Hint: aes.IV is the property to access the initialization vector in byte[]
- public void LoadK1(string path)
    - rsa mode: loads the private key (XML) and public key (XML)
        - by loading the file and assign it into a variable
        - Hint: File.ReadAllText(path, ...) reads all the text of the file and returns a string
        - Hint: rsa.FromXmlString(xmlString) loads the xml string and assigns the private and public keys
    - aes mode: shared key (BINARY)
        - by assigning the byte[] from the loaded file to the aes shared key
        - Hint: File.ReadAllBytes(...) loads a file and returns the byte[]
        - Note: the shared key and IV are binary files and you will need to call File.ReadAllBytes(...) to load it and not the StreamReader
- public void LoadK2(string path)
    - rsa mode: loads the public key (XML)
    - aes mode: loads the shared key (BINARY)
- public byte[] Encrypt(byte[] input)
    - generate the cipher bytes from input bytes
    - Invoke aes or rsa encrypt
- public byte[] Decrypt(byte[] input)
    - generate the plain bytes from the input bytes
    - Invoke aes or rsa decrypt

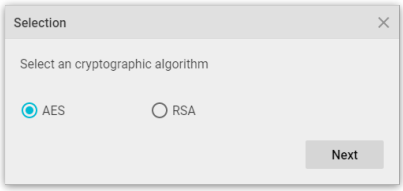
There are 3 windows: SelectionWindow, KeysWindow, and EncryptionToolWindow

LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

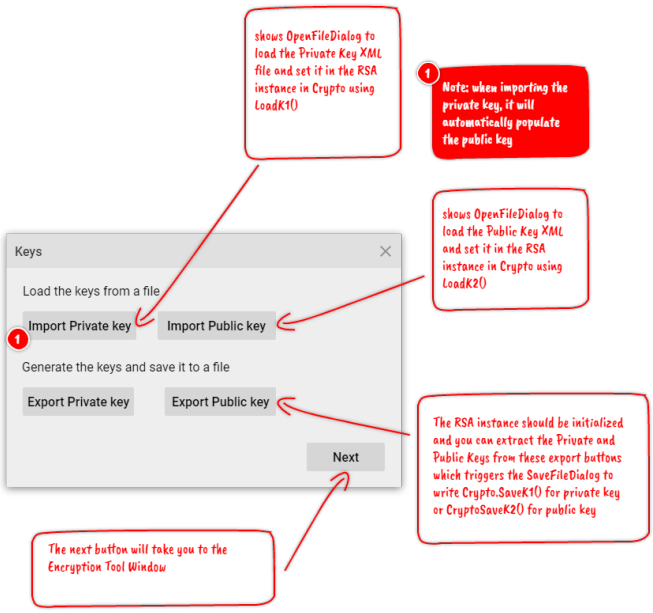
Selection Window:

The radio button will be used to determine which mode of encryption (AES or RSA) to use. The next button will take the user to the Keys Window with the AES or RSA instances initialized for Crypto.

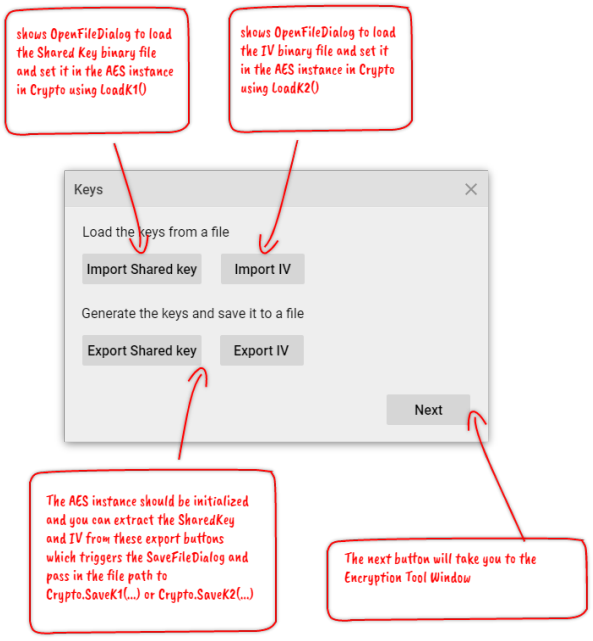


Keys Window:

In RSA mode, with the RSA instance initialized the private the public keys are already generated. However, you are able to override the existing keys using the Import Buttons for the Private and Public Keys which will show the OpenFileDialog to select a path to pass into Crypto using LoadK1 (private key & public key) or LoadK2 (only public key). Note: when you load a private key, it will automatically populate the public key as well. The Export Buttons will show a SaveFileDialog where you can save the key as an XML file using Crypto.SaveK1 (private key) or Crypto.SaveK2 (public key).



In AES Mode, the aes instance should have the SharedKey and IV generated, and you can also Import the Shared Key and Initialization Vector as well using the OpenFileDialog to select the path to pass into the Crypto.LoadK1 (for Shared Key) and Crypto.LoadK2 (for IV). The Export Buttons show the SaveFileDialog to set the path to pass into Crypto.SaveK1 (for Shared Key) or Crypto.SaveK2 (for IV).



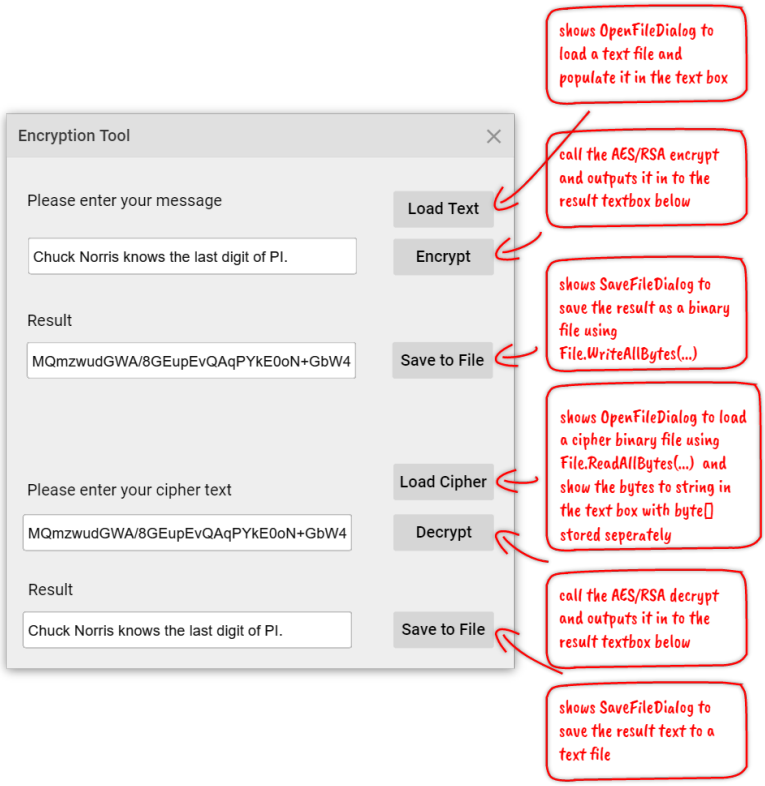
Revised on November 18 2020

LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

Encryption Tool Window:

In this window, the selected encryption algorithm instances will be initialized, and the user can now enter a string in the plain textbox and encrypt it to the result textbox, or decrypt the cipher text back to plain text in the result textbox.



Please note, when you use Convert.FromBase64String to get the bytes array, it has to be divisible by 4 and it cannot have spaces.

Be careful when you decrypt text from the cipher textbox, there are many exceptions that can be thrown so you want to wrap it in a try catch and display the exception in a MessageBox. The encrypted result when saved should be saved as a binary file using File.WriteAllBytes and the cipher text to be loaded should use File.ReadAllBytes instead of using the StreamReader.

There are a few exceptions that can be thrown due to bad input or encryption or decryption failures. Please handle failure so the user is prompted with a MessageBox containing a meaningful error so the app does not crash.

Due date

Next class, week 7 (at the start of class 6:30pm)

Submission

Implement the answers in a new project with the name “Assignment4”. This project should be submitted through committing and pushing through GIT to your VGP232 repository which you shared with the instructor.

Revised on November 18 2020

LaSalle College
Vancouver

VGP 232 Game Tools and Pipeline

Grading

Marks: Out of 100 (10% of final grade)

(60) Functional: Does it compile? Does it meet the requirements and work? Does it give the correct results?

(25) Error Handling: Does your application handle bad input? If so, does it handle failures and exceptions gracefully or does it crash?

(15) Naming convention & Comments: Does it follow the coding standards? Are your variables and method names descriptive? Did you leave descriptive comments on methods that does not have obvious functionality?

Coding Standard

C#
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions
https://www.c-sharpcorner.com/UploadFile/75a48f/rsa-algorithm-with-C-Sharp2/

Late Assignments

After the due date, the student will no longer be able to submit their assignment and will receive a 0 as I will go over the solution in the next class.

Revised on November 18 2020