Lab 7: Private Key Encryption

Details

Aim: To provide a foundation in data encryption.

Activities

If Visual Studio is installed on your machine, download the following solution [1]:

http://www.dcs.napier.ac.uk/~bill/encryption.zip

1. The .NET environment provides a number of cryptography classes. An excellent method is to use a code wrapper, which provides a simple method of accessing these classes [1]. It provides encryption algorithms such as DES, 3DES and BlowFish, and also to hash algorithms such as MD5 and SHA. The following is a simple example using the 3DES algorithm:

```
using System;
using XCrypt;
// Program uses XCrypt library from http://www.codeproject.com/csharp/xcrypt.asp
namespace encryption
{
   class MyEncryption
   {
      static void Main(string[] args)
      {
         XCryptEngine xe = new XCryptEngine();
         xe.InitializeEngine(XCryptEngine.AlgorithmType.TripleDES);
// Other algorithms are:
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.BlowFish);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.Twofish);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.DES);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.MD5);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.RC2);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.Rijndael);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.SHA);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.SHA256);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.SHA384);
//       xe.InitializeEngine(XCryptEngine.AlgorithmType.SHA512);
         xe.Key = "MyKey";

         Console.WriteLine("Enter string to encrypt:");
         string inText = Console.ReadLine();

         string encText = xe.Encrypt(inText);

         string decText = xe.Decrypt(encText);

         Console.WriteLine("Input: {0}\r\nEncr: {1}\r\nDecr: {2}",
                                                 inText,encText,decText);
         Console.ReadLine();
      }
   }
}
```

A sample run shows:

```
Enter string to encrypt:
test
Input: test
Encr: uVZLHJ3Wr8s=
Decr: test
```

By changing the method to SHA gives:

```
Enter string to hash:
test
Input: test
Hash: qUqP5cyxm6YcTAhz05Hph5gvu9M=
```

2. Implement a program for the MD5, SHA, SHA (256-bit), SHA (384-bit), SHA (512-bit) and complete the following table (for the first few characters of the signature):

Text	MD5	SHA	SHA (256)	SHA (384)	SHA (512)
apple					
Apple					
apples					
This is it.					
This is it					

How many characters does each of the types have?

3. Add the following method, and thus convert MD5 and SHA-1 Base-64 hash signatures to hex format:

```
public static string Base64ToHex(string input)
 {
   StringBuilder sb = new StringBuilder();
   byte [] inputBytes = Convert.FromBase64String(input);
   foreach(byte b in inputBytes)
   {
    sb.Append(string.Format("{0:x2}", b));
   }
   return sb.ToString();
 }
```

And change the main program so that it uses the method, such as:

```
xe.InitializeEngine(XCryptEngine.AlgorithmType.MD5);
Console.WriteLine("Enter string to encrypt:");
string inText = Console.ReadLine();

string encText = Base64ToHex(xe.Encrypt(inText));
```

Determine the hash signature for "hello", and check it again a standard MD5 program, such as from: http://pajhome.org.uk/crypt/md5/

4. Prove that the following program can decrypt an encrypted message with the correct encryption key, while an incorrect one does not. Change the program so that the user enters the encryption key, and also the decryption key:

```
xe.Key = "MyKey";

Console.WriteLine("Enter string to encrypt:");
```

```
        string inText = Console.ReadLine();

        string encText = xe.Encrypt(inText);

        xe.Key = "test"; // should not be able to decrypt as the key differs
        try
        {
           string decText = xe.Decrypt(encText);

           Console.WriteLine("Input: {0}\r\nEncr: {1}\r\nDecr: {2}",
                          inText,encText,decText);
        }
        catch { Console.WriteLine("Cannot decrypt");}  ;
        Console.ReadLine();
```

5. The following program uses a single character as an encryption key, and then searches for the encryption key, and displays it. Modify it so that it implements a 2-character encryption key, and then a 3-character one:

```
using System;
using XCrypt;
// Program uses XCrypt library from http://www.codeproject.com/csharp/xcrypt.asp
namespace encryption
{
   class MyEncryption
   {
      static void Main(string[] args)
      {
         XCryptEngine xe = new XCryptEngine();
         xe.InitializeEngine(XCryptEngine.AlgorithmType.TripleDES);
         // Other algorithms are:
         //       xe.InitializeEngine(XCryptEngine.AlgorithmType.BlowFish);
         //       xe.InitializeEngine(XCryptEngine.AlgorithmType.Twofish);
         //       xe.InitializeEngine(XCryptEngine.AlgorithmType.DES);
         //       xe.InitializeEngine(XCryptEngine.AlgorithmType.RC2);
         //       xe.InitializeEngine(XCryptEngine.AlgorithmType.Rijndael);
         xe.Key = "f";

         Console.WriteLine("Enter string to encrypt:");
         string inText = Console.ReadLine();

         string encText = xe.Encrypt(inText);
         for (char ch ='a'; ch<='z'; ch++)
         {
            try
            {
               xe.Key=ch.ToString();
               string decText = xe.Decrypt(encText);
               if (inText==decText) Console.WriteLine("Encryption key found {0}",xe.Key);
            }
            catch {}  ;
         }
         Console.ReadLine();

      }
   }
}
```

An example test run is:

```
Enter string to encrypt:
test
Encryption key found f
```

Note

C# programs can be created without the need for Visual Studio. To compile them, either go to the .NET framework directory, such as:

```
c:\> cd \WINDOWS\Microsoft.NET\Framework\v1.1.4322
```

```
C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322> csc myprog.cs
```

which produces an executable file named **myprog.exe** or create a batch file, with the contents:
c:\windows\microsoft.net\framework\v1.1.4322\csc %1

and call it compile.bat, and then run compile myprog.cs, and it produces the exe.

[1] This code is based around the Xcrypt libraries provided at http://www.codeproject.com/csharp/xcrypt.asp.