Lab Answer Key: Module 13: Encrypting and Decrypting Data

Lab: Encrypting and Decrypting the Grades Report

Exercise 1: Encrypting the Grades Report

Task 1: Create an asymmetric certificate

1. Start the MSL-TMG1 virtual machine if it is not already running.
2. Start the 20483B-SEA-DEV11 virtual machine.
3. Log on to Windows® 8 as **Student** with the password **Pa$$w0rd**. If necessary, click **Switch User** to display the list of users.
4. Switch to the Windows 8 **Start** window and then type Explorer.
5. In the **Apps** list, click **File Explorer**.
6. Navigate to the **E:\Mod13\Labfiles\Databases** folder, and then double-click **SetupSchoolGradesDB.cmd**.
7. Close File Explorer.
8. Switch to the Windows 8 **Start** window.
9. Click **Visual Studio 2012**.
10. In Microsoft® Visual Studio®, on the **File** menu, point to **Open**, and then click **Project/Solution**.
11. In the **Open Project** dialog box, browse to **E:\Mod13\Labfiles\Starter\Exercise 1**, click **Grades.sln**, and then click **Open**.
12. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.

13. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

14. In Solution Explorer, expand the **Grades.Utilities** node, and then double-click the **CreateCertificate.cmd** file.

15. Review the contents of this file.

16. Switch to the Windows 8 **Start** window.

17. In the **Start** window, right-click the background to display the task bar.

18. On the task bar, click **All apps**.

19. In the **Start** window, right-click the **VS2012 x86 Native Tools Command** icon.

20. On the task bar, click **Run as administrator**.

21. In the **User Account Control** dialog box, in the **Password** box, type **Pa$$w0rd**, and then click **Yes**.

22. At the command prompt, type the following, and then press Enter.

```
E:
```

23. At the command prompt, type the following, and then press Enter.

```
cd E:\Mod13\Labfiles\Starter\Exercise 1\Grades.Utilities
```

24. At the command prompt, type the following, and then press Enter.

```
CreateCertificate.cmd
```

25. Verify that the command returns a success message, and then close the command window.

Task 2: Retrieve the Grade certificate

1. In Visual Studio, on the **View** menu, click **Task List**.
2. In the **Task List** window, in the **Categories** list, click **Comments**.
3. Double-click the **TODO: Exercise 1: Task 2a: Loop through the certificates in the X509 store to return the one matching _certificateSubjectName** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

```
foreach (var cert in store.Certificates)
  if
(cert.SubjectName.Name.Equals(this._certificateSubjectName,
StringComparison.InvariantCultureIgnoreCase))
    return cert;
```

Task 3: Encrypt the data

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3a: Get the public key from the X509 certificate** task.
2. In the code editor, delete the following line of code:

```
throw new NotImplementedException();
```

3. In the blank line below the comment, type the following code:

```
var provider =
(RSACryptoServiceProvider)this._certificate.PublicKey.Key;
```

4. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3b: Create an instance of the AesManaged algorithm** task.

5. In the code editor, click in the blank line below the comment, and then type the following code:

```
using (var algorithm = new AesManaged())
{
```

6. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3c: Create an underlying stream for the unencrypted data** task.

7. In the code editor, click in the blank line below the comment, and then type the following code:

```
using (var outStream = new MemoryStream())
{
```

8. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3d: Create an AES encryptor based on the key and IV** task.

9. In the code editor, click in the blank line below the comment, and then type the following code:

```
using (var encryptor = algorithm.CreateEncryptor())
{
  var keyFormatter = new
RSAPKCS1KeyExchangeFormatter(provider);
  var encryptedKey =
keyFormatter.CreateKeyExchange(algorithm.Key,
algorithm.GetType());
```

10. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3e: Create**

byte arrays to get the length of the encryption key and IV task.

11. In the code editor, click in the blank line below the comment, and then type the following code:

```
var keyLength = BitConverter.GetBytes(encryptedKey.Length);
var ivLength = BitConverter.GetBytes(algorithm.IV.Length);
```

12. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3f: Write the following to the out stream** task.

13. In the code editor, click in the blank line below the comment block, and then type the following code:

```
outStream.Write(keyLength, 0, keyLength.Length);
outStream.Write(ivLength, 0, ivLength.Length);
outStream.Write(encryptedKey, 0, encryptedKey.Length);
outStream.Write(algorithm.IV, 0, algorithm.IV.Length);
```

14. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3g: Create a CryptoStream that will write the encrypted data to the underlying buffer** task.

15. In the code editor, click in the blank line below the comment, and then type the following code:

```
using (var encrypt = new CryptoStream(outStream, encryptor,
CryptoStreamMode.Write))
{
```

16. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3h: Write all the data to the stream** task.

17. In the code editor, click in the blank line below the comment, and then type the following code:

```
encrypt.Write(bytesToEncrypt, 0, bytesToEncrypt.Length);
encrypt.FlushFinalBlock();
```

18. In the **Task List** window, double-click the **TODO: Exercise 1: Task 3i: Return the encrypted buffered data as a byte[]** task.
19. In the code editor, click in the blank line below the comment, and then type the following code:

```
        return outStream.ToArray();
        }
      }
    }
  }
```


Task 4: Write the encrypted data to disk

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4a: Write the encrypted bytes to disk** task.
2. In the code editor, click in the blank line below the comment, and then type the following code:

```
File.WriteAllBytes(filePath, encryptedBytes);
```

Task 5: Build and test the application

1. On the **Build** menu, click **Build Solution**.

2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
4. In the **Class 3C** view, click **George Li**.
5. In the **Report Card** view, click **save report**.
6. In the **Save As** dialog box, browse to the **E:\Mod13\Labfiles\Reports** folder, in the **File name** box, type **GeorgeLi**, and then click **Save**.
7. In the **Report Card** view, click **Back**.
8. In the **Class 3C** view, click **Kevin Liu**.
9. In the **Report Card** view, click **save report**.
10. In the **Save As** dialog box, browse to the **E:\Mod13\Labfiles\Reports** folder, in the **File name** box, type **KevinLiu**, and then click **Save**.
11. In the **Report Card** view, click **Log off**, and then close the application.
12. On the **File** menu, click **Close Solution**.
13. Open Windows Internet Explorer®, and in the address bar, type **E:\Mod13\Labfiles\Reports\KevinLiu.xml**, and then press Enter.
14. Note the page is blank because the file is encrypted, and then close Internet Explorer.
15. Open File Explorer, and then browse to the **E:\Mod13\Labfiles\Reports** folder.
16. Right-click **KevinLiu.xml**, and then click **Edit**.
17. Review the encrypted data, close Notepad, and then close File Explorer.

Results: After completing this exercise, you should have updated the Grades application to encrypt generated reports.

Exercise 2: Decrypting the Grades Report

Task 1: Decrypt the data

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod13\Labfiles\Starter\Exercise 2**, click **School-Reports.sln**, and then click **Open**.
3. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1a: Get the private key from the X509 certificate** task.
4. In the code editor, delete the following line of code:

```
throw new NotImplementedException();
```

5. In the blank line below the comment, type the following code:

```
var provider =
(RSACryptoServiceProvider)this._certificate.PrivateKey;
```

6. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1b: Create an instance of the AESManaged algorithm which the data is encrypted with** task.
7. In the blank line below the comment, type the following code:

```
using (var algorithm = new AesManaged())
{
```

8. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1c: Create a stream to process the bytes** task.

9. In the blank line below the comment, type the following code:

```
using (var inStream = new MemoryStream(bytesToDecrypt))
{
```

10. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1d: Create byte arrays to get the length of the encryption key and IV** task.
11. In the blank line below the comment, type the following code:

```
var keyLength = new byte[4];
var ivLength = new byte[4];
```

12. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1e: Read the key and IV lengths starting from index 0 in the in stream** task.
13. In the blank line below the comment, type the following code:

```
inStream.Seek(0, SeekOrigin.Begin);
inStream.Read(keyLength, 0, keyLength.Length);
inStream.Read(ivLength, 0, ivLength.Length);
```

14. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1f: Convert the lengths to ints for later use** task.
15. In the blank line below the comment, type the following code:

```
var convertedKeyLength = BitConverter.ToInt32(keyLength, 0);
var convertedIvLength = BitConverter.ToInt32(ivLength, 0);
```

16. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1g: Determine the starting position and length of data** task.
17. In the blank line below the comment, type the following code:

```
var dataStartPos = convertedKeyLength + convertedIvLength +
keyLength.Length +
ivLength.Length;
var dataLength = (int)inStream.Length - dataStartPos;
```

18. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1h: Create the byte arrays for the encrypted key, the IV, and the encrypted data** task.
19. In the blank line below the comment, type the following code:

```
var encryptionKey = new byte[convertedKeyLength];
var iv = new byte[convertedIvLength];
var encryptedData = new byte[dataLength];
```

20. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1i: Read the key, IV, and encrypted data from the in stream** task.
21. In the blank line below the comment, type the following code:

```
inStream.Read(encryptionKey, 0, convertedKeyLength);
inStream.Read(iv, 0, convertedIvLength);
inStream.Read(encryptedData, 0, dataLength);
```

22. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1j: Decrypt the encrypted AesManaged encryption key** task.
23. In the blank line below the comment, type the following code:

```
var decryptedKey = provider.Decrypt(encryptionKey, false);
```

24. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1k: Create an underlying stream for the decrypted data** task.

25. In the blank line below the comment, type the following code:

```
using (var outStream = new MemoryStream())
{
```

26. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1l: Create an AES decryptor based on the key and IV** task.
27. In the blank line below the comment, type the following code:

```
using (var decryptor = algorithm.CreateDecryptor(decryptedKey,
iv))
{
```

28. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1m: Create a CryptoStream that will write the decrypted data to the underlying buffer** task.
29. In the blank line below the comment, type the following code:

```
using (var decrypt = new CryptoStream(outStream, decryptor,
CryptoStreamMode.Write))
{
```

30. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1n: Write all the data to the stream** task.
31. In the blank line below the comment, type the following code:

```
decrypt.Write(encryptedData, 0, dataLength);
decrypt.FlushFinalBlock();
```

32. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1o: Return**

the decrypted buffered data as a byte[] task.

33. In the blank line below the comment, type the following code:

```
            return outStream.ToArray();
        }
    }
}
}
}
```

Task 2: Build and test the solution

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, click **Browse**.
4. In the **Browse For Folder** dialog box, browse to the **E:\Mod13\Labfiles\Reports** folder, and then click **OK**.
5. Click **Print**.
6. In the **Save Print Output As** dialog box, browse to the **E:\Mod13\Labfiles\Reports\ClassReport** folder, in the **File name** box, type **3CReport**, and then click **Save**.
7. In the **The School of Fine Arts** dialog box, click **OK**, and then close the application.
8. Open File Explorer, and browse to the **E:\Mod13\Labfiles\Reports\ClassReport** folder.
9. Right-click **3CReport.oxps**, and then click **Open**.
10. Review the unencrypted report, and then close the XPS Viewer.

Results: After completing this exercise, you should have a composite unencrypted report that was generated from the encrypted reports.