CBA · NAU

College of Business Administration Northern Arizona University
Box 15066
Flagstaff AZ 86011

Data File Security Through Cryptography Using the Advanced Encryption Standard (RIJNDAEL)

Working Paper Series 05-03— March 2005

John D. Haney
Clinical Professor of CIS
(928) 523-7352
john.haney@nau.edu

Northern Arizona University
College of Business Administration
Box 15066
Flagstaff, AZ 86011-5066

CBA·NAU

College of Business Administration
Northern Arizona University
Box 15066
Flagstaff AZ 86011

Data File Security Through Cryptography Using the Advanced Encryption Standard (RIJNDAEL)

John D. Haney

INTRODUCTION

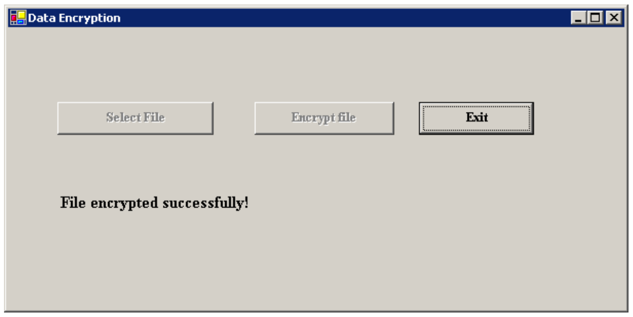
The security of data has become an increasing concern for both businesses and consumers alike; therefore, there is an increasing demand to keep data private. Cryptography, which involves two processes: encryption and decryption, is one method of achieving that goal. The former converts plain text into cipher text, and the later converts cipher text back into plain text. Ciphers are cryptography algorithms, or mathematical functions, used for the encryption and decryption of data (Slain, 2004).

There are two types of encryption, asymmetric and symmetric. In the asymmetric encryption method, one key (public) is used to encrypt the data and a different key (private) is used to decrypt the data. With symmetric encryption the same key (secret) is used to encrypt and decrypt the data. Data that has been encrypted can only be decrypted by using the same key. The risk is that if the key is discovered the data can easily be decrypted (Slain, 2004).

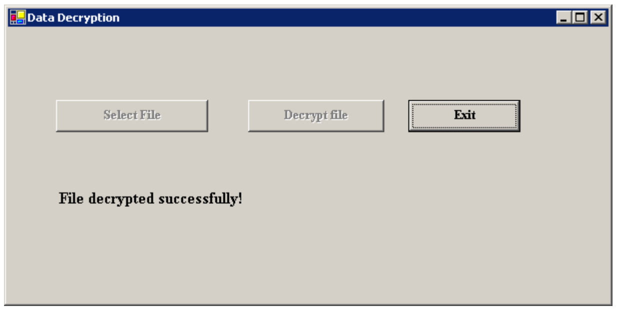
As an example of how cryptography can add to the security of data, a text file that contains two fields, a name and a test score is encrypted and written to a new data file. Then the encrypted data file is decrypted and written to a new data file. These two processes are accomplished in two separate programs, which are written in Visual Basic within Microsoft's .NET environment.

In Microsoft's .NET framework the System.Security.Cryptography namespace provides classes for the following: Symmetric Encryption, Asymmetric Encryption, Hashing, Digital Signatures, Digital Certificates, and XML Signatures. Within Symmetric Encryption there are four options: Data Encryption Standard (DES) which was developed by IBM and adopted as a national standard in 1977; Triple DES which is similar to DES but the process is repeated three times; Rivest Cipher (RC2) designed by Ron Rivest and is considerably faster that DES; and Rijndael which was developed by Joan Daemen and Vincent Rijmen and has been adopted as the Advanced Encryption Standard (AES) by the National Institute of Standards and technology (NIST) (Slain, 2004).

Symmetric Encryption using the Rijndael block cipher method is used in this example for encrypting and decrypting the data. Again encryption is accomplished in one program and decryption is accomplished in another. Through this method the secret key that is used to encrypt the data in the first program must be available in the second program to decrypt the data. The graphical user interfaces for these two programs are shown below.



File encrypted successfully!



File decrypted successfully!

THE RIJNDAEL BLOCK CIPHER AND THE ADVANCED ENCRYPTION STANDARD (AES)

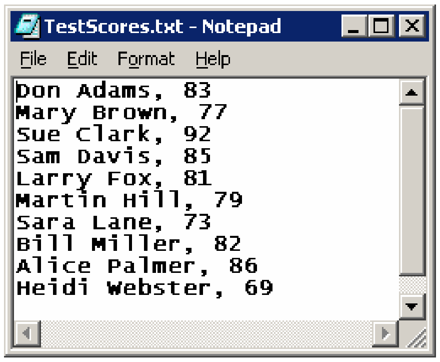
Rijndael, a block cipher whose design was strongly influenced by the block cipher Square, is currently the Advanced Encryption Standard (AES) (Wikipedia, 2005). The AES was adopted by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197 in November 2001, and was preceded by the Data

Encryption Standard (DES) (Wikipedia, 2005). The cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. The name Rijndael is comprised of the names of the inventors, which is pronounced "Rhine dahl." Rijndael is a substitution-permutation network that is fast, requires little memory, and is relatively easy to implement. As a new encryption standard, Rijndael is being deployed on a very large scale.

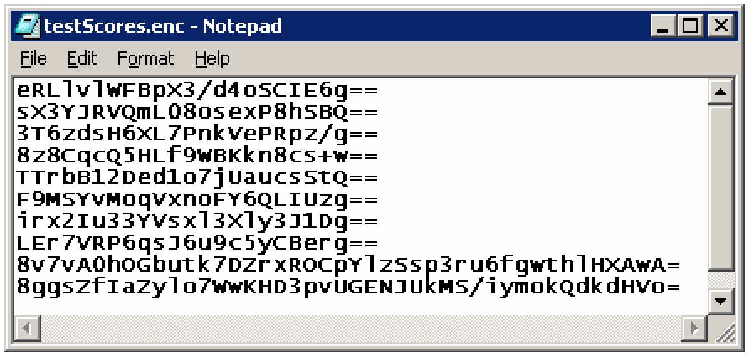
Technically, AES is not exactly Rijndael because Rijndael supports a larger range of block and key sizes. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits. AES operates on a 4x4 array of bytes, termed the state. For encryption, each round consists for four stages: SubBytes- a non-linear substitution where each byte is replaced by another according to a lookup table; ShiftRows- where each row of the state is shifted cyclically a certain number of steps; MixColumns- a mixing operation on the columns of the state that combines the four bytes in each column using a linear transformation; AddRoundKey- each byte of the state is combined with the round key, where each round key is derived from the cipher key using a key schedule. The final round omits the MixColumns stage (Wikipedia, 2005).

DATA FILE ENCRYPTION

For the example of encryption, a sequential text data file is read in, the data is encrypted, and then the encrypted data is written to a new data file. The method of encryption used is the RijndaelManaged object of the .NET System.Security.Cryptography class. Following is the data before encryption and after encryption.



```
Don Adams, 83
Mary Brown, 77
Sue Clark, 92
Sam Davis, 85
Larry Fox, 81
Martin Hill, 79
Sara Lane, 73
Bill Miller, 82
Alice Palmer, 86
Heidi Webster, 69
```



```
eRLlvlWFBpX3/d4oSCIE6g==
sX3YJRVQmLO8osexP8hSBQ==
3T6zdsH6XL7PnkVePRpz/g==
8z8CqcQ5HLf9WBKkn8cs+w==
TTrbB12Ded1o7jUaucsStQ==
F9MSYvMoqVxnoFY6QLIUzg==
irx2Iu33YVsxl3Xly3J1Dg==
LEr7VRP6qsJ6u9c5yCBerg==
8v7vA0hoGbutk7DZrxROCpYlzSsp3ru6fgwthlHXAwA=
8ggsZfIaZylo7WwKHD3pvUGENJUkMS/iymokQdkdHVo=
```

The program has two classes: the form class and the encryption class that contains the encryption function where the actual encryption takes place. The function is called from the form class and supplied three arguments: (1) the text that will be encrypted (strPlain); (2) the encryption key (keyMain); (3) and the initial vector (IV). The encrypted text (strText) is then returned to the form class.

The code for the function is shown below. The function creates an instance of the RijndaelManaged cryptography object (RMcrypto). A memory stream (outStream) is created, and an encrypted stream (CryptStream) is created to hold the encrypted data using the encryption key and initial vector, which is wrapped around the memory stream. An instance of the StreamWriter (SWriter) is created to hold the encrypted data, which is wrapped around the encrypted stream. Using the data argument (strPlain) the encryption takes place by writing to the instance of the StreamWriter (SWriter). The StreamWriter and memory stream are then flushed to ensure that complete encryption of the data takes place. The encrypted data is converted into a byte array (byteEncrypt) and converted to base64 (strCrypt). All of the streams are closed and the encrypted text (strCrypt) is returned to the calling code.

```
'Encryption Class
Public Class Encrypt
 'Function returns a new string that is encrypted and encoded in base64
Function EncryptString(ByVal strPlain As String, ByVal keyMain() As Byte, ByVal IV() As Byte) As String
    Dim RMcrypt As New System.Security.Cryptography.RijndaelManaged
    'Create a memory stream to hold the encrypted data
    Dim outStream As New System.IO.MemoryStream
    'Create the crypto stream to encrypt and hold the encrypted data
    Dim CryptStream As New System.Security.Cryptography.CryptoStream(outStream,
   RMcrypt.CreateEncryptor(keyMain, IV), System.Security.Cryptography.CryptoStreamMode.Write)
```

```
        'Create a StreamWriter to write the plain text data and do the encryption into the outStream buffer
        Dim SWriter As New System.IO.StreamWriter(CryptStream)
        ‘Now call the write command to do the encryption
        SWriter.Write(strPlain)
        SWriter.Flush()
        CryptStream.FlushFinalBlock()
        ‘convert the outStream buffer back to a string
        Dim byteEncrypt() As Byte = outStream.ToArray
        ‘Convert to base64 and return it
        Dim strCrypt As String = Convert.ToBase64String(byteEncrypt)
        SWriter.Close()
        outStream.Close()
        CryptStream.Close()
        Return strCrypt
    End Function
End Class
```

The code for the form class is abbreviated to show only the code that is relevant to the encryption process. An Import statement is required to reference the StreamReader and StreamWriter.

Imports System.IO

Class data variables are defined for the StreamReader, StreamWriter, a string variable for holding the record read from the input file, and a string variable for holding the encrypted record that will be written to the output file.

```
Private Input As StreamReader
Private Output As StreamWriter
Private strLine As String = " "
Private strEncrypt As String = " "
```

Following is the code for the subroutine that performs the data encryption. The input and output data files are opened. An instance of the encryption class that contains the encryption method is created (encryptInstance), and the encryption key and initial vector key are defined, where the values are arbitrarily assigned. Both of the keys are 16 bytes in length or 128 bits.

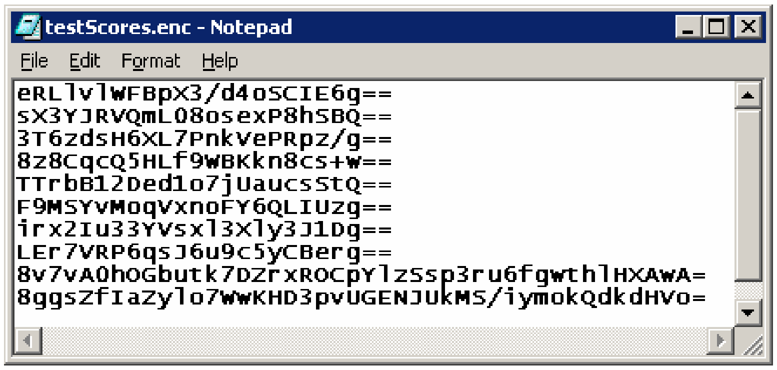
Next is the record cycle loop. A record is read from the plain text data file (strLine) into a string data variable. The record (strLine), along with the encryption key (keyMain), and initial vector key (IV) is sent to the encryption function. The function returns an encrypted value (strEncrypt). The encrypted data is then written to the output data file.

At the conclusion of the record cycle a message is displayed in a label that the encryption was successful, and both data files are closed. If an error occurs during encryption an error message is displayed in the label.

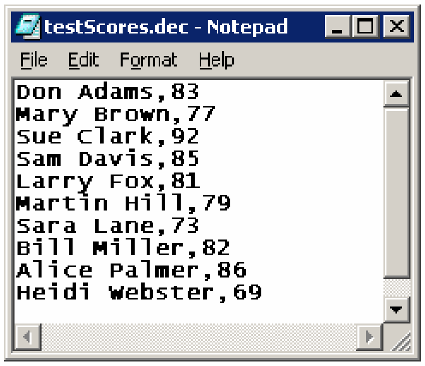
```
 Private Sub btnEncrypt_Click(. . . . .) Handles btnEncrypt.Click

        Try
            Input = New StreamReader(“testScores.txt”)
            Output = New StreamWriter("testScores.enc")
            Dim encryptInstance As New Encrypt
            Dim keyMain As Byte() = {&H92, &H3A, &HC1, &H89, &HB6, &H43, &HCD, &H3F, &H5C, &H6C,
           &H92, &HE4, &H72, &H89, &HA8, &HD1}
            Dim IV As Byte() = {&H14, &H21, &H97, &H44, &HFC, &HC7, &H48, &H8F, &HC4, &HE3, &H2D,
           &H45, &HC3, &H14, &H34, &H62}

            Do Until Input.Peek = -1
                strLine = Input.ReadLine()
                strEncrypt = encryptInstance.EncryptString(strLine.Trim, keyMain, IV)
                Output.WriteLine(strEncrypt)
            Loop

            lblMessage.Text = "File encrypted successfully!"
            Input.Close()
        Output.Close()
      Catch
        lblMessage.Text = "Error on encryption!"
      End Try
    End Sub
```

DATA FILE DECRYPTION

For the example of decryption, the encrypted data created in the example above is read in, the data is decrypted, and then the decrypted data is written to a new data file. The method of decryption as in the case of encryption uses the RijndaelManaged object of the .NET System.Security.Cryptography class. Following is the data before decryption and after decryption.



```
eRLlvlWFBpX3/d4oSCIE6g==
sX3YJRVQmL08osexP8hSBQ==
3T6zdsH6XL7PnkVePRpz/g==
8z8CqcQ5HLf9WBKkn8cs+w==
TTrbB12Ded1o7jUaucsStQ==
F9MSYvMoqVxnoFY6QLIUzg==
irx2Iu33YVsxl3Xly3J1Dg==
LEr7VRP6qsJ6u9c5yCBerg==
8v7vA0hoGbutk7DZrxROCpYlzSsp3ru6fgwthlHXAwA=
8ggsZfIaZylo7WwKHD3pvUGENJUkMS/iymokQdkdHVo=
```



```
Don Adams,83
Mary Brown,77
Sue Clark,92
Sam Davis,85
Larry Fox,81
Martin Hill,79
Sara Lane,73
Bill Miller,82
Alice Palmer,86
Heidi Webster,69
```

As with decryption, the program has two classes: the form class and the decryption class that contains the decryption function. The function is called from the form class and supplied three arguments: (1) the text that will be decrypted (strCrypt); (2) the encryption key (keyMain); (3) and the initial vector (IV). The decrypted text (strDecrypt) is then returned to the form class.

The code for the function is shown below. The function creates an instance of the RijndaelManaged cryptography object (RMcrypto), and the base64 string (strCrypt) is converted into a byte array (byteEncrypt). The byte array is converted into a memory stream (inStream), and a cryptography stream (CryptStream) is created that is wrapped around the memory stream which uses the same encryption key and initial vector with which the data was encrypted. An instance of the StreamReader (SReader) is created to hold the decrypted data, which is wrapped around the decrypted stream. The decrypted data is then read into the decrypted string variable (strDecrypt). All of the streams are closed and the decrypted text (strDecrypt) is returned to the calling code.

```
‘Decryption function
Public Class Decrypt
  Function DecryptString(ByVal strCrypt As String, ByVal keyMain() As Byte, ByVal IV() As Byte) As String
    Dim RMcrypto As New System.Security.Cryptography.RijndaelManaged
     ‘Convert the Base64 string into a byte array
     Dim byteEncrypt() As Byte
     byteEncrypt = System.Convert.FromBase64String(strCrypt)
     ‘convert the bytes into a memory stream
     Dim inStream As New System.IO.MemoryStream(byteEncrypt)
     ‘Create the decryption structure
    Dim CryptStream As New System.Security.Cryptography.CryptoStream(inStream,
        RMcrypto.CreateDecryptor(keyMain, IV), System.Security.Cryptography.CryptoStreamMode.Read)
     ‘Read the incoming stream and decrypt it
     Dim SReader As New System.IO.StreamReader(CryptStream)
     ‘Decrypt and convert the stream back to a string
     Dim strDecrypt As String
     strDecrypt = SReader.ReadToEnd()
```

```
        SReader.Close()
        inStream.Close()
        CryptStream.Close()
        Return strDecrypt
    End Function
End Class
```

The code for the form class is abbreviated to show only the code that is relevant to the decryption process. An Import is required to reference the StreamReader and StreamWriter.

Imports System.IO

Class data variables are defined for the StreamReader; StreamWriter; a string variable for holding the record read from the input file; an array for holding the fields from the record; a string variable for holding the encrypted data; and a string variable for holding the decrypted data that will be written to the output file.

```
Private Input As StreamReader
Private Output As StreamWriter
Private strLine As String = " "
Private strFields(2) As String
Private strEncrypt As String = " "
Private strDecrypt As String = " "
```

Following is the code for the subroutine that performs the data decryption. The input and output data files are opened, and an instance of the decryption class that contains the decryption method is created (decryptInstance). The encryption key and initial vector key are defined where the values are arbitrarily assigned, but must match the encryption key and initial vector key when the data file was encrypted. Both of the keys are 16 bytes in length or 128 bits. A delimiter is defined for separating the fields within the record once the record is decrypted.

Next is the record cycle loop. A record is read from the encrypted data file into a string data variable (strEncrypt). The decryption function is called sending the data that will be decrypted, along with the encryption key (keyMain), and initial vector key (IV). The function returns the decrypted data (strDecrypt), and is then split into fields (strFields) using the delimiter (,). A composite record (strDecrypt) is then built and written to the output file.

At the conclusion of the record cycle a message is displayed in a label that the decryption was successful, and both data files are closed. If an error occurs during decryption an error message is displayed in the label.

```
Private Sub btnEncrypt_Click(. . . . . ) Handles btnDecrypt.Click

        Try
            Input = New StreamReader(“testScores.enc”)
            Output = New StreamWriter("testScores.dec")
            Dim decryptInstance As New Decrypt
           Dim keyMain As Byte() = {&H92, &H3A, &HC1, &H89, &HB6, &H43, &HCD, &H3F, &H5C, &H6C,
           &H92, &HE4, &H72, &H89, &HA8, &HD1}
            Dim IV As Byte() = {&H14, &H21, &H97, &H44, &HFC, &HC7, &H48, &H8F, &HC4, &HE3, &H2D,
           &H45, &HC3, &H14, &H34, &H62}
            Dim chrDelimiter As Char
            chrDelimiter = Convert.ToChar(",")

            Do Until Input.Peek = -1
                strEncrypt = Input.ReadLine()
                strLine = decryptInstance.DecryptString(strEncrypt.Trim, keyMain, IV)
                strFields = strLine.Split(chrDelimiter)
                strDecrypt = strFields(0) & "," & Convert.ToInt32(strFields(1))
                Output.WriteLine(strDecrypt)
            Loop

            lblMessage.Text = "File decrypted successfully!"
            nput.Close()
            Output.Close()
        Catch
            lblMessage.Text = "Error on decryption!"
        End Try
    End Sub
```

SUMMARY AND CONCLUSIONS

The awareness of security has placed an increasing focus on the need for data security. One means of ensuring data security is the use of cryptography. Through this mechanism data can be encrypted for security and then later decrypted for additional use. The process of encryption used has been the Rijndael block cipher, a symmetric encryption method that uses a secret key for both encryption and decryption. The strength of the Rijndael block cipher has led the national institute of Standards and Technology to adopt Rijndael and the advanced encryption standard (AES).

To show how cryptography can be used to add security to data files, the Rijndael cipher is used to encrypt an unsecured data file, and then later decrypt the secured data file. This has been accomplished by using two different programs. The first program reads the unsecured data file and creates a secure encrypted file. The second program reads the secured encrypted file and creates a new unsecured data file. The encryption and decryption processes use the same secret key. Without the secret key, the encrypted data cannot be decrypted. However, if the secret key becomes known the secured data is no longer secure. One further security measure could be to encrypt the secret key, but a risk always remains. In any event, the use of cryptography adds a considerable level of security to otherwise unsecured data.

References

Cloudscape, CodeGuru. “Symmetric Encryption” (2004), Online: http://www.codeguru.com/Csharp/.NET/net_security/encryption/article.php/c8511/

Daemen, Joan & Rijmen, Vincent. (2002). The Design of Rijndael. New York, NY: Spring-Verlag.

Daemen, Joan & Rijmen, Vincent. “The Rijndael Block Cipher: AES Proposal” (1999), Online: http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

Daemen, Joan & Rijmen, Vincent. “The Block Cipher Rijndael” (2004), Online: http://www.esat.kuleuven.ac.be/~rijmen/rijndael

Djordjevic, Nenad. “File Encryption” (2002), Online: http://www.csharphelp.com/archives2/archive315.html

Intelligent Solution Inc. “Rijndael Encryption in VB.NET” (2003), Online: http://www.freevbcode.com/ShowCode.Asp?ID=4520

Network of Excellence in Cryptology. ”AES Lounge” (2005), Online: http://www.iaik.tu-graz.ac.at/research/krypto/AES/

Slain, Deon. “Symmetric Key Encryption using Rijndael and C#” (2004), Online: http://www.sadeveloper.net/Articles_View.aspx?articleID=157

Wikipedia, the free encyclopedia. “Advanced Encryption Standard” (2005), Online: http://en.wikipedia.org/wiki/AES