Asynchronous File Processing and Encryption Algorithms

Holger Findling

In the book Programming F# the author Chris Smith provides us with an example programming Asynchronous file IO using F# Async Workflows. The function `asyncProcessFile (filePath: string) (processBytes: byte[] -> byte[])` has two parameters; one that takes the file path pointing to our data and the other takes an algorithm to perform work on the data. In the examples provided below, we are using asyncProcessFile to encrypt data using two different types of encryption algorithms, shift Key encryption and a Symmetric algorithm.

In the first example, we are copying data from one file to another without encryption. The algorithm used to perform this task is the lambda identity function. Identity is truly parametric polymorphic, and in this example takes a byte stream as input and returns the byte stream. The lambda identity function always returns its input argument. 'b -> 'b.

```
let proc = asyncProcessFile filename (fun b -> b)
```

In the second example we are copying data from one file and encrypting the data before writing it to another file. The encryption algorithm used, shift key encryption, is also known as substitution cypher. Symmetric cyphers have been used since the Roman Empire. Shift key encryption or substitution cipher can easily be deciphered using a brute attack or using some form of character frequency analysis.

The purpose of the examples provided in this paper is to demonstrate asynchronous file processing using encryption and decryption. We make no attempt to discuss the merits of the algorithms or their data security provided.

Function shiftEncryptAlgorithm key data requires two arguments, one is the number of bytes (key) to shift the data and the other is the input byte stream. We can curry the function and then apply it to `asyncProcessFile`.

```
let key = 4
let SEA = shiftEncryptAlgorithm key
let proc = asyncProcessFile filename SEA
```

The second encryption algorithm provides symmetric encryption using the Cryptographic Application Programming Interfaces (CAPI) implementation of the Advanced Encryption Standard (AES) algorithm. In this example we curry the function as well.

```
let SymmA = SymmetricAlgorithm aesCSP
let proc = asyncProcessFile filename2 SymmA
```

```
//*********************************************************************
// Project: Asynchronous Processing using Encryption Algorithms
// Comp Intel
// MyCYGroup.com
//*********************************************************************
open System
open System.IO
open System.Security.Cryptography
open System.Text


//*********************************************************************
// Shift Encryption Algorithm
//*********************************************************************
let shift (m:string) o =
   List.ofArray (Encoding.ASCII.GetBytes(m))
   |> List.map (fun x ->
      Convert.ToChar(((((int)x+o)%255)+255)%255).ToString())
   |> List.reduce (fun a x -> a + x)


let shiftEncryptAlgorithm key data =
   let x = shift (Encoding.ASCII.GetString(data)) key
   Encoding.ASCII.GetBytes(x)


//*********************************************************************
// Symmetric Algorithm
//*********************************************************************
let SymmetricAlgorithm (aesp:AesCryptoServiceProvider) (data:byte[]) =
   let  xfrm: ICryptoTransform = aesp.CreateEncryptor()
   let encryptFile: byte[]  = xfrm.TransformFinalBlock(data, 0, data.Length)
   encryptFile


let DecryptSymmetricAlgorithm(aesp:AesCryptoServiceProvider) (data:byte[]) =
   let xfrm: ICryptoTransform = aesp.CreateDecryptor()
   let decryptFile: byte[] = xfrm.TransformFinalBlock(data, 0, data.Length)
   decryptFile


//*********************************************************************
// Asynchronuous Processing
//*********************************************************************
let asyncProcessFile (filePath: string) (processBytes: byte[] -> byte[]) =
   async {
      let inputFile = Path.GetFileName(filePath)
      printfn "Processing Input File: %s " inputFile
      printfn " "

      // Open the input file
      let fptr = new FileStream (filePath, FileMode.Open)
      let bytesToRead = int fptr.Length

      // Read the input file
      let! data = fptr.AsyncRead(bytesToRead)
      printfn "File: %s, Number of bytes read %d " inputFile data.Length
      printfn "data: %A" data

      // Process the data
```

```
        let data' = processBytes data

        // Write to the Output file
        let resultFile = filePath + ".results"
        printfn "Output File: %s " resultFile
        let fptrO = new FileStream (resultFile , FileMode.Create)
        do! fptrO.AsyncWrite(data', 0, data'.Length)

        // Close the output file
        fptr.Close()
        fptrO.Close()

        printfn "Finished processing file [%s]" <| Path.GetFileName (filePath)
    } |> Async.Start



//**********************************************************************
// Program
//**********************************************************************
[<EntryPoint>]

//**********************************************************************
// Function: main
//**********************************************************************
let main (args: string[]) =

    // File to be processed.
    let filename = @"../../bin/Log.txt"

    //**************************************************************
    // Copying a file to Log.txt.results
    //**************************************************************
    let proc = asyncProcessFile filename (fun b -> b)

    Console.WriteLine("Copying the file ...")
    Console.WriteLine("Hit <Enter> to continue")
    Console.WriteLine("")
    Console.ReadKey(true) |> ignore

    //**************************************************************
    // Using Shift Encryption to encrypt a file
    //**************************************************************
    let key = 4
    let SEA = shiftEncryptAlgorithm key
    let proc = asyncProcessFile filename SEA

    Console.WriteLine("Copying the file using Shift Encryption ...")
    Console.WriteLine("Hit <Enter> to continue")
    Console.WriteLine("")
    Console.ReadKey(true) |> ignore

    // Shift Decrypt
    let filename = @"../../bin/Log.txt.results"
    let SEA = shiftEncryptAlgorithm -key
    let proc = asyncProcessFile filename SEA

    Console.WriteLine("Decrypting the file ...")
    Console.WriteLine("Hit <Enter> to continue")
    Console.WriteLine("")
```

```
    Console.ReadKey(true) |> ignore


    //**************************************************************
    // Using Symmetric Algorithm to encrypt a file
    //**************************************************************
    let filename2 = @"../../bin/Log1.txt"

    // create a .Net AES Crypto Service Provider
    let aesCSP : AesCryptoServiceProvider = new AesCryptoServiceProvider();
    aesCSP.GenerateKey();
    aesCSP.GenerateIV();

    let SymmA = SymmetricAlgorithm aesCSP
    let proc = asyncProcessFile filename2 SymmA

    Console.WriteLine("Copying the file using Syymetric Algorithm Encryption ...")
    Console.WriteLine("Hit <Enter> to continue")
    Console.WriteLine("")
    Console.ReadKey(true) |> ignore

    // Decrypt Symmetric Algorithm
    let filename = @"../../bin/Log1.txt.results"
    let DSymmA = DecryptSymmetricAlgorithm aesCSP
    let proc = asyncProcessFile filename DSymmA

    Console.WriteLine("Decrypting the file ...")
    Console.WriteLine("Hit <Enter> to continue")
    Console.WriteLine("")
    Console.ReadKey(true) |> ignore


    Console.ReadKey(true) |> ignore

    // Program exit code
    0;;


(*****************************************************************)
```