Valid Document

From <http://support.microsoft.com/kb/307010>

Cryptography program can be used as is, but you need to create the file MyData.Text, store it to a location that does not need admin privileges, and then amend the file paths in main method.

using System;

using System.IO;

using System.Security;

using System.Security.Cryptography;

using System.Runtime.InteropServices;

using System.Text;

namespace CSEncryptDecrypt

{

class Class1

{

// Call this function to remove the key from memory after use for security

[System.Runtime.InteropServices.DllImport("KERNEL32.DLL", EntryPoint = "RtlZeroMemory")]

public static extern bool ZeroMemory(IntPtr Destination, int Length);

// Function to Generate a 64 bits Key.

static string GenerateKey()

{

// Create an instance of Symetric Algorithm. Key and IV is generated automatically.

DESCryptoServiceProvider desCrypto = (DESCryptoServiceProvider)DESCryptoServiceProvider.Create();

// Use the Automatically generated key for Encryption.

return ASCIIEncoding.ASCII.GetString(desCrypto.Key);

}

static void EncryptFile(string sInputFilename,

string sOutputFilename,

string sKey)

{

FileStream fsInput = new FileStream(sInputFilename,

FileMode.Open,

FileAccess.Read);

FileStream fsEncrypted = new FileStream(sOutputFilename,

FileMode.Create,

FileAccess.Write);

DESCryptoServiceProvider DES = new DESCryptoServiceProvider();

DES.Key = ASCIIEncoding.ASCII.GetBytes(sKey);

DES.IV = ASCIIEncoding.ASCII.GetBytes(sKey);

ICryptoTransform desencrypt = DES.CreateEncryptor();

CryptoStream cryptostream = new CryptoStream(fsEncrypted,

desencrypt,

CryptoStreamMode.Write);

byte[] bytearrayinput = new byte[fsInput.Length];

fsInput.Read(bytearrayinput, 0, bytearrayinput.Length);

cryptostream.Write(bytearrayinput, 0, bytearrayinput.Length);

cryptostream.Close();

fsInput.Close();

fsEncrypted.Close();

}

static void DecryptFile(string sInputFilename,

string sOutputFilename,

string sKey)

{

DESCryptoServiceProvider DES = new DESCryptoServiceProvider();

//A 64 bit key and IV is required for this provider.

//Set secret key For DES algorithm.

DES.Key = ASCIIEncoding.ASCII.GetBytes(sKey);

//Set initialization vector.

DES.IV = ASCIIEncoding.ASCII.GetBytes(sKey);

//Create a file stream to read the encrypted file back.

FileStream fsread = new FileStream(sInputFilename,

FileMode.Open,

FileAccess.Read);

//Create a DES decryptor from the DES instance.

ICryptoTransform desdecrypt = DES.CreateDecryptor();

//Create crypto stream set to read and do a

//DES decryption transform on incoming bytes.

CryptoStream cryptostreamDecr = new CryptoStream(fsread,

desdecrypt,

CryptoStreamMode.Read);

//Print the contents of the decrypted file.

StreamWriter fsDecrypted = new StreamWriter(sOutputFilename);

fsDecrypted.Write(new StreamReader(cryptostreamDecr).ReadToEnd());

fsDecrypted.Flush();

fsDecrypted.Close();

}

static void Main()

{

// Must be 64 bits, 8 bytes. But remember a 64 Bit key is very weak and can easily be broken with parallel computing.

// The thing to ask here is - does the key remain the same if we iteratively encrypt using the same plaintext.

// How could we break this encryption?

// Is this a one off key?

// Distribute this key to the user who will decrypt this file - the receiver

string sSecretKey;

// Get the Key for the file to Encrypt. Generate the key automatically. User does not enter key or passphrase.

sSecretKey = GenerateKey();

// For additional security Pin the key. Make sure that the key will not be garbage collected before it's usefulness is completed.

GCHandle gch = GCHandle.Alloc(sSecretKey, GCHandleType.Pinned);

// Encrypt the file using the input file, output file and secret key

EncryptFile(@"C:\Data\Temp\MyData.txt",

@"C:\Data\Temp\Encrypted.txt",

sSecretKey);

// Decrypt the file. Using the ciphertext and the output file for plain text and the secret key.

DecryptFile(@"C:\Data\Temp\Encrypted.txt",

@"C:\Data\Temp\Decrypted.txt",

sSecretKey);

// Remove the Key from memory. This is clever - put a watch on the variable string sSecretKey above and observe the results in a step through.

ZeroMemory(gch.AddrOfPinnedObject(), sSecretKey.Length \* 2);

gch.Free();

}

}

}