Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 4, April 2015, pg.59 – 64

RESEARCH ARTICLE

Secure File Transmission using Split & Merge Technique

Neha Koli[1], Siddhant Parkar[2], Karan Shah[3], Asst. Prof Manisha Giri[4]

[1]Department of Comp. Engineering, Atharva College of Engineering, India

[2]Department of Comp. Engineering, Atharva College of Engineering, India

[3]Department of Comp. Engineering, Atharva College of Engineering, India

[4]Department of Comp. Engineering, Atharva College of Engineering, India

[1] nehakoli2841994@gmail.com; [2] siddparkar@gmail.com; [3] Karan_shah108@yahoo.com; [4] manisha.giri1@gmail.com

***Abstract*— This paper deals with the development of an application under the data security domain. In today's age, security of data has become paramount. Continuous flow of information in removable storage and/or internet has made data susceptible to mischief. Convenient, safe and fast transmission of data is the aim behind developing this application. The application splits a large file into multiple chunks making it practically unreadable or incomprehensible to achieve data security. It is impractical to upload a large raw file with slow internet connection. Such files can be compressed using the application. An email module added to the application makes the transfer even easier.**

***Keywords*— Splitting, Merging, Compression, Decompression, Encryption, Decryption**

I. INTRODUCTION

The application to be developed primarily deal with data security. The basic idea behind this application is split and merges. Suppose if you have media file of 3 GB and have two removable storages of 2 GB each. A dilemma arises of effectively transferring the said media file. Apart from the security aspect of the application to be developed an easy way to transfer the file is achieved. This big file can be split into multiple smaller pieces of user defined sizes. Now these can be conveniently stored to the removable storage devices. Along with providing convenience of transfer, it also secures the media file. The file extension would be changed because of the split operations performed on them. This renders the file naturally unreadable which makes it safe from prying eyes. Alas this approach isn't very helpful when the security of text files is involved. With this in mind, encryption is introduced. Encoding messages or information in such a way that only authorized parties can read it is known as encryption. Converting plain text into ciphertext with or without the help of a key is the core idea behind encryption. Ciphertext is encrypted or encoded information because it contains a form of the original plaintext that is unreadable by a human or computer without the proper cipher to decrypt it. To understand or access the encrypted data one need to decrypt it which also requires a key. The process of decryption is the reverse of encryption in which the cipher text is subject to the process of deciphering which converts the cipher

text into plain text which the receiver can read. When the internet speed is slow, it is impractical to send large media files. Compression allows us to decrease the file size so it can be sent faster over a network. Compression can be done using two methods lossy and lossless. In lossy compression some redundant data is lost therefore lossless is usually preferred over lossy. Lossy Compression is usually used for media files like image, audio, video, etc and lossless compression is usually used for documents in text format. An email module is used to transfer the file.

II. ALGORITHMS & TECHNIQUES USED

We consider and discuss algorithms and techniques which have been zeroed in on, to use in the software development.

1. Encryption Algorithm

1.1 AES (Advanced Encryption Standard).

For the encryption purpose AES[2] has been considered. AES[2] is a symmetric encryption algorithm. It belongs to the class of block ciphers. The name Rijndael is derived from its two creators Vincent Rijmen and John Daemen. Because of its symmetric nature key management is very essential as the encrypted data can be decoded using the same key which was used to encrypt it in the first place. AES[2] is based on the design principle of both substitution and permutation. The Advanced Encryption Standard[2] is operated on a fixed block size of 128 bits , along with varying key sizes of 128 bits, 196 bits and 256 bits. Depending upon the key size 256 ,128 and 196 it takes 14, 10 and 12 cycles respectively to get encrypted.

2. Data compression Techniques

2.1 G ZIP

For compression the gzip algorithm is considered [3]. A variation of LZ77 is the deflation algorithm used by gzip. In the input data duplicated strings are found. A pointer to the previous string replaces second occurrence of a string, in the form of a 1pair (length, distance). 32K bytes is the distance, and and 258 bytes the length. A sequence of literal bytes is emitted when a string does not occur anywhere in the previous 32K bytes. `String' is not restricted to printable characters and must be taken as an arbitrary sequence of bytes.

III.MODULES

1. Split and Merge

Both the split and merge modules are combined. The file to be secured is selected by the user. With the help of file stream the contents of the file are read. The contents need to be manipulated. For that we write the data into a byte array using BinaryWriter. The split size of the file chunk can be specified by the user. The application should run a loop where the byte array in which the contents are stored are retrieved at every pass and save in a new file with extension .dat Also the size of the array is decreased subsequent for each chunk retrieved. The loop goes on until the buffer is empty. We get the output as split chunks of a readable file. The chunks so formed are unreable to humans. In this way files can be securely transmitted from one system to another. Now if the user is at receivers end i.e. he has received the split chunks for security purpose and now wishes to read/write the file. He will have to have access to the whole file. For that he has to merge the files. To achieve that, the user has to have each and every piece of the split file. The user has to select the first piece and the target directory to save the merged file.

2. Encryption and Decryption

The provision has to be made so that the user is able to select the file that has to be encrypted. Once the file gets encrypted the user can save the file on the system or the encrypted file can be directly transferred using the email module. Splitting generally makes the file unreadable for human eyes. But for text files, we can even view split files using a notepad. So for this we introduced an encryption

module using AES. AES will make text files unreadable. These files can then be safely split. In the encryption module we select the text file using open. Then we select encrypt. When the text is encrypted we save it.

3. Compression and Decompression

GZip[3] is used for compression. Compression is of two types. One is lossy compression and one is lossless compression. Lossy compression is mainly used for images and videos. Lossless compression is used for compression of application files. Compression facilitates quick upload and transfer of files over the network. We select the file to be compressed and then we select the destination folder. The compressed file is stored in a zip format. For decompression we select the zip file and select the output folder. The compressed file is then decompressed.

4. Email

Transfer of file is ubiquitous in today's environment. The world has become a smaller place due to the advent of the internet. It goes without saying that when an application involving encryption and compression is developed. And email module would be added to it. The email module facilitates an easy transfer of data over the network. Email allows attachments of maximum size 25mb. When the file is split and compressed using our application, an email module is not just an afterthought. The developer can connect the email module being developed to any portal. Simple Message Transport Protocol can be used to connect the said module to the email server.

IV. PSEUDO CODE

A pseudo code for the application being developed is described as follows.

```
public bool beginSplit()
{
BinaryReader breader = null;
BinaryWriter bwriter = null;
byte[] buffer = null;
long fSize = 0;
int curFileSplit = 0;
long numLeftOverBytes = 0;
long bytesRead = 0;
long curSplitSize = 0;
string curSplitFileName = null;
string baseName = null;
if (!checkInputs(true, true)) {
return false;
}
fSize = FileSystem.FileLen(targFileName);
if (fSize < MaxSplitSize) {
return false;
}
targfileACL = new FileInfo(targFileName).GetAccessControl();
buffer = new byte[BufferSize];
baseName = targFileName.Substring(targFileName.LastIndexOf("\\") + 1, targFileName.Length -
(targFileName.LastIndexOf("\\") + 1));
curSplitFileName = outputDir + "\\" + baseName + splitFirstName + splitReplaceToken + splitExt;
breader = new BinaryReader(new FileStream(targFileName, FileMode.Open, FileAccess.Read,
FileShare.None, BufferSize, FileOptions.SequentialScan));
bwriter = new BinaryWriter(File.Create(curSplitFileName.Replace(splitReplaceToken,
curFileSplit.ToString()), BufferSize, FileOptions.None, targfileACL));
bytesRead = breader.Read(buffer, 0, BufferSize);
while (bytesRead > 0) {
if ((curSplitSize + bytesRead) > MaxSplitSize) {
numLeftOverBytes = MaxSplitSize - curSplitSize;
bwriter.Write(buffer, 0, Convert.ToInt32(numLeftOverBytes));
bwriter.Flush();
```



```
    bwriter.Close();
    curFileSplit += 1;
    bwriter = new BinaryWriter(File.Create(curSplitFileName.Replace(splitReplaceToken,
curFileSplit.ToString()), BufferSize, FileOptions.SequentialScan | FileOptions.WriteThrough, targfileACL));
    curSplitSize = bytesRead - numLeftOverBytes;
    if (onProgressStep != null) {
    double temp789 = breader.BaseStream.Position / fSize;
    double temp147 = curFileSplit / curStats.NumSplits;
    onProgressStep(Convert.ToInt32(Math.Floor((temp789) * 100)), Convert.ToInt32(Math.Floor((temp147)
* 100)));
    }
    if (!(numLeftOverBytes == 0)) {
    bwriter.Write(buffer, Convert.ToInt32(numLeftOverBytes - 1), Convert.ToInt32(curSplitSize));
    } else {
    bwriter.Write(buffer, Convert.ToInt32(numLeftOverBytes), Convert.ToInt32(curSplitSize));
    }
    } else {
    bwriter.Write(buffer, 0, Convert.ToInt32(bytesRead));
    curSplitSize += bytesRead;
    if (onProgressStep != null) {
    double temp258 = breader.BaseStream.Position / fSize;
    double temp369 = curFileSplit / curStats.NumSplits;
    onProgressStep(Convert.ToInt32(Math.Floor((temp258) * 100)), Convert.ToInt32(Math.Floor((temp369)
* 100)));
    }
    }
    bytesRead = breader.Read(buffer, 0, BufferSize);
    }
    return true;
    }
    private void DoSplit()
    {
    boolean b = SpliterClass.beginSplit();
    if (b == false)
    {
    print("Cannot Split File");
    }
    else
    {
    print("Split Done Successfully");
    }
    }
    function bool beginReassembly()
    {
    BinaryReader breader = null;
    BinaryWriter bwriter = null;
    byte[] buffer = null;
    int curFileSplit = 0;
    long bytesRead = 0;
    string curSplitFileName = "";
    string baseName = "";
    System.Security.AccessControl.FileSecurity targfileACL = null;
    //store file access control list, used on each split file
    long fSize = 0;
    if (!checkInputs(true, false))
    {
    return false;
    }
    targfileACL = FileInfo(targFileName).GetAccessControl();
    buffer = new byte[BufferSize];
```



```
    baseName = targFileName.Substring(targFileName.LastIndexOf("\\") + 1, targFileName.Length -
(targFileName.LastIndexOf("\\") + 1));
    baseName = baseName.Substring(0, baseName.LastIndexOf(splitFileFirstName));
    curSplitFileName = outputDir + "\\" + baseName + splitFirstName + splitReplaceToken + splitExt;
    bwriter = new BinaryWrite(File.Create(outputDir + "\\" + baseName, BufferSize,
System.IO.FileOptions.None, targfileACL));
    while (File.Exists(curSplitFileName.Replace(splitReplaceToken, curFileSplit.ToString()))) {
    breader = new BinaryReader(new System.IO.FileStream(curSplitFileName.Replace(splitReplaceToken,
curFileSplit.ToString()), FileMode.Open, FileAccess.Read,FileShare.None,
BufferSize,FileOptions.SequentialScan));
    bytesRead = breader.Read(buffer, 0, BufferSize);
    fSize = FileSystem.FileLen(curSplitFileName.Replace(splitFileReplaceToken, curFileSplit.ToString()));

    while (bytesRead > 0) {
    bwriter.Write(buffer, 0, Convert.ToInt32(bytesRead));
    if (onProgressStep != null) {
    double temp14 = (breader.BaseStream.Position / fSize) * 100;
    double temp15 = (curFileSplit / curStats.NumSplits) * 100;
    onProgressStep(Convert.ToInt32(Math.Floor(temp14)), Convert.ToInt32(Math.Floor(temp15)));
    }
    bytesRead = breader.Read(buffer, 0, BufferSize);
    }
    breader.Close();
    breader = null;
    //avoid exception in finally block
    return true;
    }
    private void DoAssemble()
    {
    boolean b = beginReassembly();
    if (b == false)
    {
    print("Cannot Merge File");
    }
    else
    {
    print("Merge Done Successfully");
          }
```

V. FEATURES AND APPLICATIONS

The application developed will have a variety of uses in the domain of data security. Particularly for day to day users who do not have access to sophisticated methods of securing data. The proposed application would have possible use in Silence Tracking Radar and Region of Interest.

VI. FUTURE WORK

The application under development will have several life applications. These applications can be extended to Silence Tracking. Silence tracking involves recording by radar over a period of time. The recording if needed to be monitored can be split using our application and check. Another application of the application can be finding ROI or Region of Interest in an image like an X Ray.

VII. CONCLUSIONS

The developed application will work on windows platform. Black box and white box testing will be carried out to make sure all the arising errors are ruled out. The unified GUI will make operations easy for the end user by bringing together the four modules of Split Merge, Compress Decompress, Encrypt Decrypt, and Email. Splitting and/or encryption make every known format practically unreadable.

REFERENCES

[1] Amandeep Singh Sidhu [M.Tech]1, Er. Meenakshi Garg [M.Tech]2 (2014)" An Advanced Text Encryption & Compression System Based on ASCII Values & Arithmetic Encoding to Improve Data Security"

[2] Joan Daemen, Vincent Rijmen(2002\) "The Design of Rijndael: AES - The Advanced Encryption Standard"

[3] Franceschini Robert.; Dept. of Computer Sci., Univ. of Central Florida, Orlando, FL, USA ; Mukherjee Amar (1996) "Data compression using encrypted text"

[4] Muhanad Hayder (2009) "Design and Implementation of a File Splitter and Merger Software"

[5] Siddhant Parkar, Neha Koli, Karan Shah, Manisha Giri (2015) "Study of Different Algorithms and Techniques for Secure File Transmission"

[6] Rauschert, P. Fac. of Electr., Inf. & Media Eng., Wuppertal Univ., Germany Klimets, Y. ; Velten, J. ;Kummert, A. (2004) "Very fast GZIP compression by means of content addressable memories"

[7] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting(20002) "Improved Cryptanalysis of Rijndael"

[8] Henri Gilbert and Marine Minier "A collisions attack on the 7-rounds Rijndael"

[9] Fredrik A. Dahl, Margreth Grotle, Jūratė Šaltytė Benth, Bård Natvig(2008) "Data splitting as a countermeasure against hypothesis fishing: with a case study of predictors for low back pain"

