Enhancing File Transfer Security and Efficiency through Compression, Fragmentation and Reassembling Techniques

Ngbede Barnabas Michael
Faculty of Computing, Department of Science, Federal University Computer of Lafia, Nigeria.

Karim Usman, PhD
Department of Mathematics and Computer Science, Benue State University, Makurdi, Nigeria

ABSTRACT
The transfer of digital files faces significant challenges, including security vulnerabilities, interruptions during transmission, and variable transfer speeds. This research investigates a robust mechanism to enhance file security during transmission, minimize completion times, and ensure the swift transfer of data in unreliable service environments. The proposed approach involves the encryption and compression of file content, fragmentation of the compressed files, and reassembly, as well as decompression of the sub-files upon transmission. Additionally, this method leverages the secure transmission capabilities of the File Transfer Protocol (FTP) to protect sensitive information transmitted over the internet. A Java Enterprise Application was developed to implement this method, and seven different file formats, each with a size of 10MB, were tested to evaluate system performance. Key performance metrics recorded included the execution time for compression, fragmentation, reassembly, and decompression, the size of the compressed files, and the number of fragments post-compression. The results indicated that performance metrics varied depending on the file format, with optimal performance observed for text-based files. Consequently, the developed system is recommended for secure file sharing over the web, particularly for confidential or security-sensitive data files.

General Terms
Cryptography, Cybersecurity, File Transfer Protocol, Compression, Fragmentation, Reassembling, Decompression

Keywords
File Transfer, File Security, Throughput, File Fragment, File Security

1. INTRODUCTION
As digital data files continue to be generated rapidly in the world of technology and from day-to-day usage of technological devices, file sharing becomes commonplace among users of these technologies. These files can also be shared between users in remote locations and are required to be transferred through a transmission medium/sets of media, giving rise to the need for secure and rapid file transfers, as well as retrieval techniques at both ends (senders' and receivers' ends respectively). Challenges arise when files to be transferred between remote users are relatively large and need to be transferred via a particular transmission medium in a secure mode to prevent a malicious attack or hijack of such file(s); hence the need for the provision of secure and fast-medium for data transfer to facilitate the rate of transfer and retrieval of such large digital files. Secure file transfer across transmission media is crucial in today's digital landscape. Cryptography plays a vital role in ensuring data confidentiality and integrity during transmission [1]. This research study was conceived to critically look into proffering sustainable improvement of secure and fast transfer of a digital file(s) between users in remote locations using compression, fragmentation, reassembling, and decompression techniques.

A major drawback in the transfer of digital files include the security of transmitted files, the restart feature and the speed with which such files are transmitted, where there is a possibility of probable file hijack, a file is interrupted on transmission, that is, in the middle of its processing, the entire file transfer needs to restart from the beginning, i.e., the work that is partially completed is lost and slow rate of transfer respectively. The primary challenge confronting contemporary digital systems is the substantial size of data, necessitating efficient compression techniques to optimize storage capacity and reduce processing time [3].

The integration of data compression and encryption techniques in digital data transmission presents a multifaceted approach with significant advantages. Firstly, the amalgamation of compression and encryption facilitates efficient data transmission by reducing the data size, thereby enhancing the speed and effectiveness of information exchange over networks. This streamlined process optimizes resource utilisation and contributes to faster transmission speeds, ultimately improving the overall performance of digital communication systems. Moreover, the incorporation of encryption ensures data security by safeguarding sensitive information from unauthorized access during transmission, thereby upholding confidentiality and integrity in the communication process [2].

A standard mechanism for improving file security on transmission, reducing completion times and ensuring the rapid transfer of data files in an unreliable service environment is compression (possibly encrypt file content), fragmentation, reassembling and decompression of such sub-files on transmission. This technique compresses, and fragments a data file into smaller ordered blocks of sub-files, which can be transmitted through a data distribution/transmission channel using File Transfer Protocol on a TCP/IP network (e.g. the internet), reassembles the ordered fragmented data blocks in accordance to the original ordering scheme to produce the original data file and then decompresses the file. "The file

conversion Software comprises a file fragmentation utility employed in a computer at the transmitting end of the System that segments the huge data file into a plurality of smaller blocks or files. A file reassembly utility is employed in a computer at the receiving end of the System. The file reassembly utility reconstructs the original file by concatenating each of the smaller ordered files or blocks following the original ordering Scheme to produce the original data file. (United States Patent - US 6,233,252 B1)".

According to [13], a significant challenge in the transfer of digital files, particularly large ones, is the "RESTART" feature. This feature necessitates that if a file transfer is interrupted mid-process, the entire transfer must restart from the beginning, resulting in the loss of any partially completed work. To mitigate this issue and reduce completion times in unreliable service environments, fragmentation is employed. For instance, a file transmitted over an unreliable channel is divided into packets. Similarly, in computing environments, a file or job is fragmented through the introduction of ordered checkpoints. This fragmentation ensures that in the event of a server failure, only the work associated with the fragment currently being processed is lost.

The necessity to address the drawbacks in the transfer of digital files, particularly in the areas of cybersecurity and network throughput efficiency, led to the development of a novel approach. This approach combines file compression, fragmentation, reassembling, and decompression techniques to enhance file transfer security and efficiency.

2. REVIEW OF RELATED LITERATURE

[4] proposed the TranSecure Model, which integrates file compression to minimize size and enhance bandwidth efficiency before encryption by the Trusted Party Agent (TPA). The combined application of compression and encryption secures data during both transmission and storage, ensuring the privacy and integrity of files. It effectively addresses the challenges posed by unreliable service environments and potential security breaches. The TranSecure Model, despite its focus on file security and efficient data transfer, can still encounter transmission delays due to various factors like compression and encryption overhead, network latency, server processing time, file size, and system resource limitations. To address these delays and enhance file transmission efficiency, integrating file fragmentation and reassembly techniques can be beneficial. By breaking down large files into smaller fragments for transmission and reassembling them at the destination.

The integration of compression and encryption techniques provides a robust defence mechanism against potential security threats in digital data transmission. By compressing data before encryption, the resulting encrypted information becomes more resistant to unauthorized decryption attempts, thereby enhancing protection against data breaches. This dual approach not only strengthens security measures but also reduces the overall cost of data transmission, promoting cost-effective communication. The strategic combination of compression and encryption techniques emphasizes the importance of balancing efficiency, security, resource optimization, and cost-effectiveness in contemporary communication systems, underscoring its critical role in ensuring secure and efficient digital data transmission [2]. Moreover, the additional incorporation of file fragmentation and reassembly techniques can further augment file security during transmission.

The study by [5] enhances file transfer security and efficiency in mobile ad hoc networks (MANETs) by employing a combination of compression, fragmentation, and reassembling techniques. By utilizing encryption algorithms such as ES-LZW and various compression methods, the research ensures data security during transmission and reduces the size of data packets to enhance bandwidth efficiency and conserve network resources. The fragmentation of data into multiple parts and their transmission over disjoint paths increases resilience against node compromise attacks. Additionally, the use of compression and encryption techniques addresses memory constraints in MANETs by minimizing the memory requirements for ciphertext. The efficient reassembling of data fragments on the receiver side guarantees the integrity and completeness of the transferred data, ultimately improving the overall efficiency of file transfer in MANET environments. Transferring the methods from the study on secure and efficient data transfer in MANETs to a web application highlights significant distinctions arising from the structured network infrastructure, resource availability, security concerns, data transmission protocols, user interaction, scalability, and performance requirements specific to web environments. Despite encountering security threats common to both MANETs and web applications, the diverse nature of these threats underscores the need for customized security strategies in web applications.

[6] developed an advanced cryptographic system incorporating compression, fragmentation, and XOR operations to enhance the security and efficiency of data transmission. This system, tailored for fast and efficient data transmission, particularly in mobile devices with limited resources, employs several methodologies to bolster security. Firstly, data is compressed prior to encryption. Next, the data is converted into byte code and divided into chunks. Multiple XOR operations are then performed on these chunks to increase randomness and enhance security. Despite its strengths, the study notes a limitation in terms of slower data transmission rates on wireless networks.

[15] introduces a Java-based file splitter and merger tool designed to facilitate the distribution of large files via removable media and to reassemble these files on the destination machine. Hayder emphasizes that file splitting is a fundamental concept in computer science, playing a critical role in various applications.

[16] addressed the challenge of digital forensics in the reassembly of document fragments using statistical modelling tools applied in data compression. Their research focused on developing a general process model for the automatic analysis of a collection of fragments to reconstruct the original document by correctly ordering the fragments. This process involves assigning probabilities to the likelihood that two given fragments are adjacent in the original file, employing context modelling techniques from data compression. Given the inherently scattered nature of digital evidence, forensic analysts often encounter situations where they must recover and reassemble deleted files from randomly scattered data blocks on storage media. Traditional tools struggle with reassembling non-contiguous data blocks in the correct order without the appropriate file table entries. This study proposed building a system with a predefined fragmentation scheme capable of reassembling file fragments back into the original file, regardless of the file content's structure.

The problem investigated by [11] was that of reassembling a scattered document as stated thus: suppose a set $\{A_0, A_1, \ldots, A_n\}$

of fragments of document **A**. A permutation $\prod(\text{theta})$ would be computed such that **A** = $A = A\prod(0) \,||\, A\prod(1) \,||\, A\prod(2) \,||\, \ldots ||A\prod(n)$, where ‖ denotes the concatenation operator. In other words, to determine the order in which fragments $\boldsymbol{A_i}$ need to be concatenated to yield the original document **A.** They assumed fragments were recovered without loss of data, that is, concatenation of fragments in the proper order yields the original document intact. To determine the correct fragment reordering, they needed to identify fragment pairs that are adjacent to the original document. The technique they employed is to use a dictionary of the underlying language of the document. With this approach, a fragment $\boldsymbol{A_j}$ would be considered to be a likely candidate fragment to follow $\boldsymbol{A_i}$ if the word that straddles the boundaries of $\boldsymbol{A_i}$ and $\boldsymbol{A_j}$ is found in the dictionary.

However, a dictionary-based approach is language-specific and it is therefore not feasible for the variety of documents that will be encountered by a Forensic Analyst in the field. Furthermore, for non–textual files, like executable files, a dictionary may not be readily available or easy to reconstruct. Finally, if more than one dictionary matches are found then how does a Forensic Analyst select between the two? They observed a technique to identify the correct reassembly with high probability and formally compute the permutation $\prod$ (theta) such that the value:

$$\Pi_{i=0}^{n-1}\, C(\Pi(i), \Pi(n+1))$$

is maximized over all possible permutations $\Pi(theta)$ of degree **n**. Accordingly, this permutation is most likely to be the one that leads to a correct reconstruction of the document.

They came up with the general approach of reassembling document fragments. First, they built an ***n-order*** context model for a given document by accumulating the context models of individual fragments into a single model. Using the model for any ordered pair of fragments. Consider a sliding window of size **n**. Place the window at the end of the first fragment and slide the window in one position estimating the probability of the upcoming character (n + 1) by using the character in the window as the current context and using the context model obtained from the total pool of fragments for probability estimation. Continuing this sequence for **d** sequence characters, where $d \leq n$, gives the candidate probability for the fragment pair where the edges quantify candidate probabilities of the adjacency of the corresponding node in the original document. After this a heuristic solution is to compute a few near-optimal re-orderings of the fragments.

According to [14], the daily transmission of large volumes of data in cyberspace is increasingly accompanied by a rise in cyber threats and attacks. Consequently, there is a pressing need for highly secure and reliable mediums for data transmission between remote users across the web. Due to cybersecurity concerns, the transmission of data, particularly sensitive and confidential information, is often deemed unreliable. Even well-trusted technological firms explicitly state that data provided or transmitted using their services is at the user's own risk.

The implementation of a secure file transfer system using compression, fragmentation, reassembling and decompression techniques investigated in this research study improves both the security of transmitted data files and reduces the time taken for the transmission of such large files over a network.

[7] asserted that with the development of computer technology, encryption algorithms, or ciphers, must be bit-oriented. This requirement arises because the data to be encrypted encompasses not only text but also numbers, graphics, audio, and video. Converting these diverse data types into a bit stream, encrypting the stream, and subsequently transmitting the encrypted stream is a convenient and effective approach.

In their 2003 study, "File Splitting, Scalable Coding, and Asynchronous Transmission in Streamed Data Transfer," Ted et al. posited that files can be segmented and transmitted in parts to a client via a communication channel. They observed that at least some of these transmitted segments can be locally cached, facilitating subsequent streaming playback of the file. This method reduces bandwidth usage by transmitting only the uncached portions of the file, which are then combined with the locally cached segments.

[15] conducted research resulting in the development of a File Splitter and Merger Software designed to facilitate the distribution of files via removable media such as floppies and CDs. This software splits large files, which cannot fit onto a single USB flash drive or other media, into smaller files that can be easily accommodated. Additionally, it allows these split files to be merged back on the destination machine. The system operates across major operating systems on personal computers (PCs). However, the research had limitations, notably the lack of consideration for mobile (hand-held) devices and web users due to their lower popularity at the time. Given the current prominence of mobile devices and file transfer over the internet in rapid data generation and transmission, a system that implements the secure transfer of files through compression, fragmentation, and reassembly techniques would significantly enhance and build upon this earlier research.

File fragmentation and reassembly are crucial for the fast, efficient, and secure transfer of digital files using FTP within the TCP/IP transmission model. This study introduces a technique that compresses data files, segments them into discrete packets, simultaneously transmits these packets to the receiving computer, and reassembles them into the original file.

Prior research on file fragmentation and reassembly often limited their methods by considering file content and structure. This research advances the field by implementing an ordering scheme for fragmenting files into sub-files, regardless of their content or structure, and reassembling these fragments using the same scheme.

Numerous studies have explored file splitting and merging (fragmentation and defragmentation). However, the novelty of this research lies in its ability to compress and fragment a data file into sub-files, reassemble them at the recipient's end, and apply this methodology in developing a web-based application. Unlike previous studies that focused on point-to-point (P2P) systems, distributed systems, and other communication channels, this research develops an enterprise application that compresses and fragments files and then reassembles them back into the original file, irrespective of the file content's structure.

3. METHODOLOGY

The proposed methodology for this research introduces a comprehensive system aimed at enhancing the security and efficiency of digital file transfer through the integration of compression, fragmentation, reassembling, and decompression

techniques. The system, developed as a web-based enterprise application, is accessible from any device and facilitates secure file exchange by minimizing file sizes, splitting them into manageable fragments, and reassembling them at the destination. Key modules of the system include the Compression Module, which reduces file size and encodes content; the Fragmentation Module, which recursively divides files into smaller fragments based on a predefined algorithm; the Reassembling Module, which merges the fragments in reverse order; and the Decompression Module, which restores the file to its original state. This methodology has practical applications in digital forensic analysis, particularly in reconstructing recovered file fragments, and in the efficient streaming of file content to users in remote locations.

3.1 Analysis of the proposed system

Given the critical need for secure and efficient digital file transfer, as highlighted in the reviewed literature, this study develops an enterprise web application designed to compress files, fragment them, reassemble the fragments, and decompress the files back into their original form. This web-based enterprise application is accessible from any device, facilitating the secure transfer of digital files through the use of compression, fragmentation, reassembling, and decompression techniques.

i. *Compression Module:* The compression module is responsible for reducing the size of digital files and encoding their content.

ii. *Fragmentation Module:* This module is a core component of the proposed system, executing the recursive file fragmentation algorithm on the selected file. The process begins by recursively splitting the file from a pre-determined maximum size for a component section and continues until the smallest component section's size is equal to or less than the specified minimum size. Fragmentation is performed according to the ordering scheme defined in the fragmentation algorithm.

iii. *Reassembling Module:* Another fundamental component of the proposed system, the reassembling module manages the merging of file fragments back into the original file. This is achieved by applying the reverse of the ordering scheme used during fragment*ation.*

iv. *Decompression Module:* This module handles the decompression of the reassembled file, restoring it to its original size and decoding its content.

This methodology has been applied in two major areas: (a) Digital Forensic Analysis for the reassembling of recovered document fragments. (b) Streaming of digital file content between users in remote locations, after which the streamed chunks of data are reassembled into a single digital file.

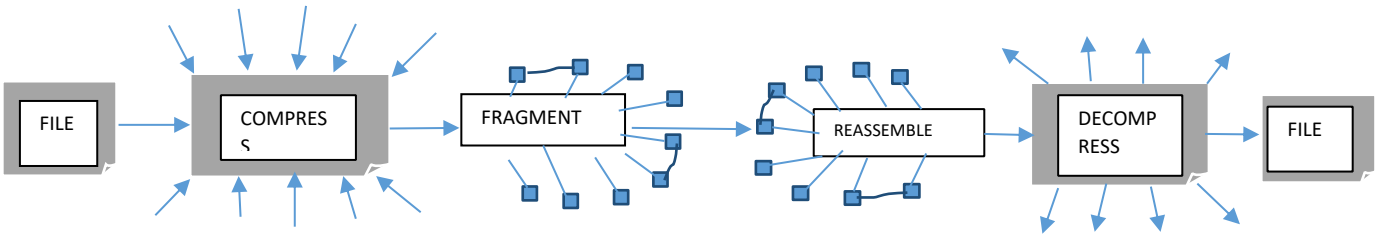


Figure 1 Diagrammatic Representation of Proposed System

3.2 System Design

Upon initiating the system, the user is prompted to select a file for transmission. Once the file is selected, the system first applies the compression algorithm to reduce the file's size, enhancing the efficiency of its transfer. Following compression, the system executes the fragmentation process, wherein the compressed file is divided into several uniquely addressed blocks, each conforming to predefined size parameters. These blocks, each individually identifiable by their unique addresses, are then transmitted to the destination. Upon successful reception, the system reverses the process by reassembling the fragments in their correct sequence using their unique identifiers. Finally, the reassembled file is decompressed, restoring it to its original format and ensuring that the file integrity is maintained throughout the transfer process. This systematic approach optimizes both security and transmission efficiency while ensuring seamless file reconstruction at the destination.

3.3 Program Flowchart

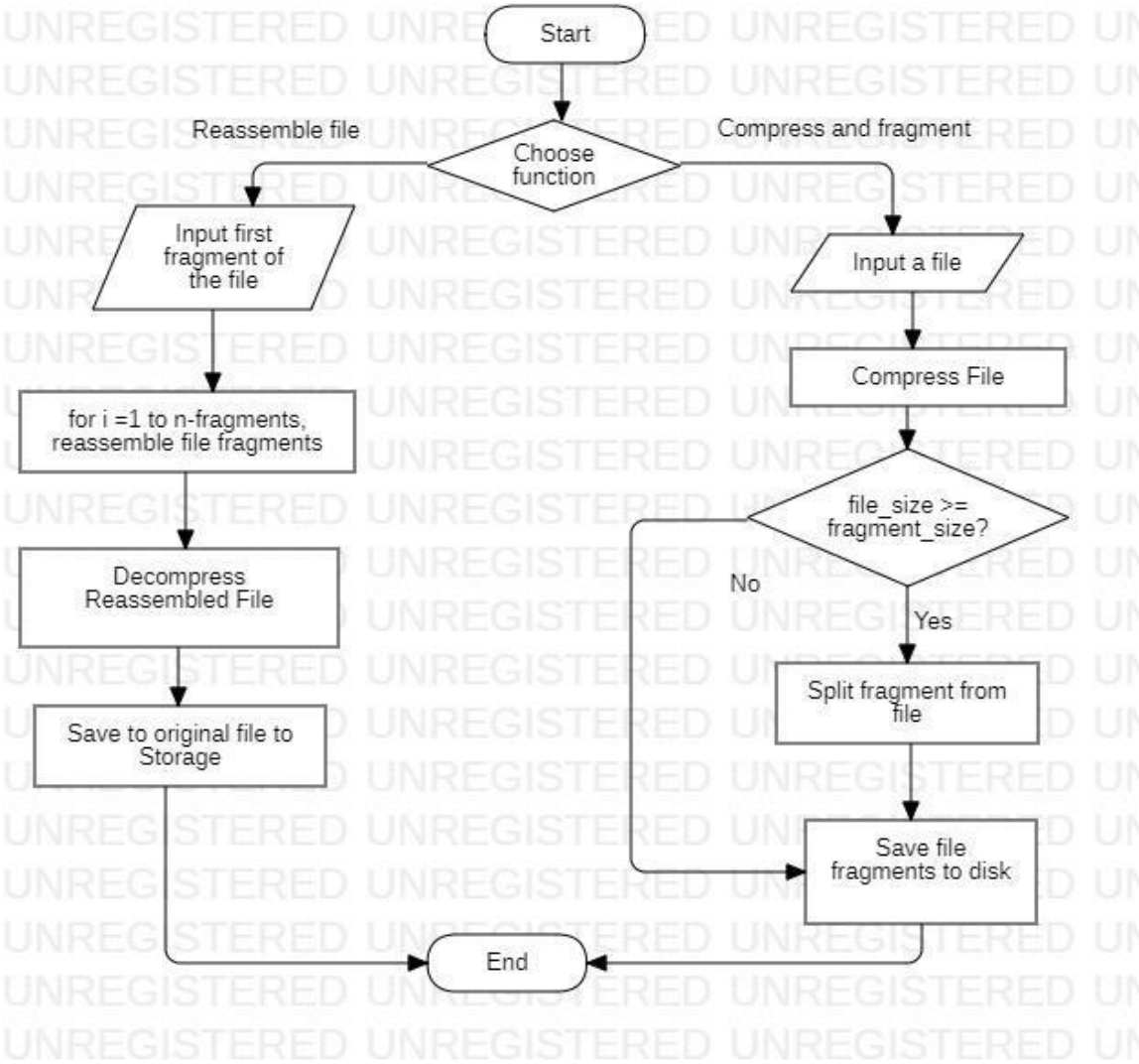


Figure 2: Proposed system flowchart

4. RESULTS

The development environment utilized for creating the system comprised a Windows 10 (64-bit) operating system, JDK 8, NetBeans 8.2 IDE, JBoss (WildFly 14) application server, StarUML for model prototyping, and Google Chrome Browser for testing. The system was developed on a Personal Computer (PC) with the following hardware specifications: Intel® Core™ i5-3230M CPU, 2.60GHz processor, and 8GB of installed memory (RAM).

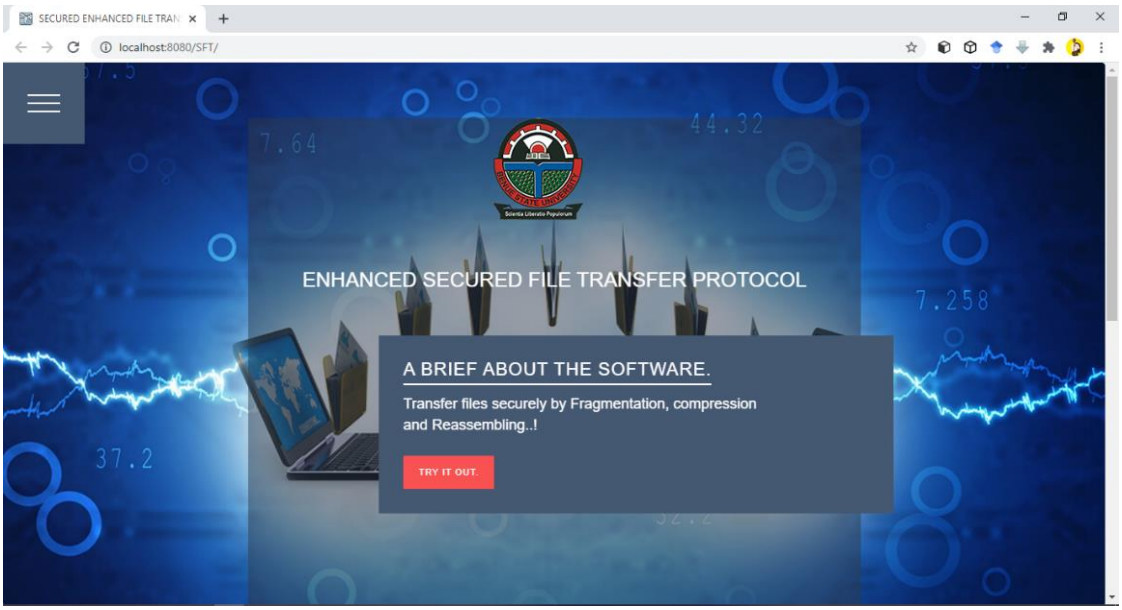
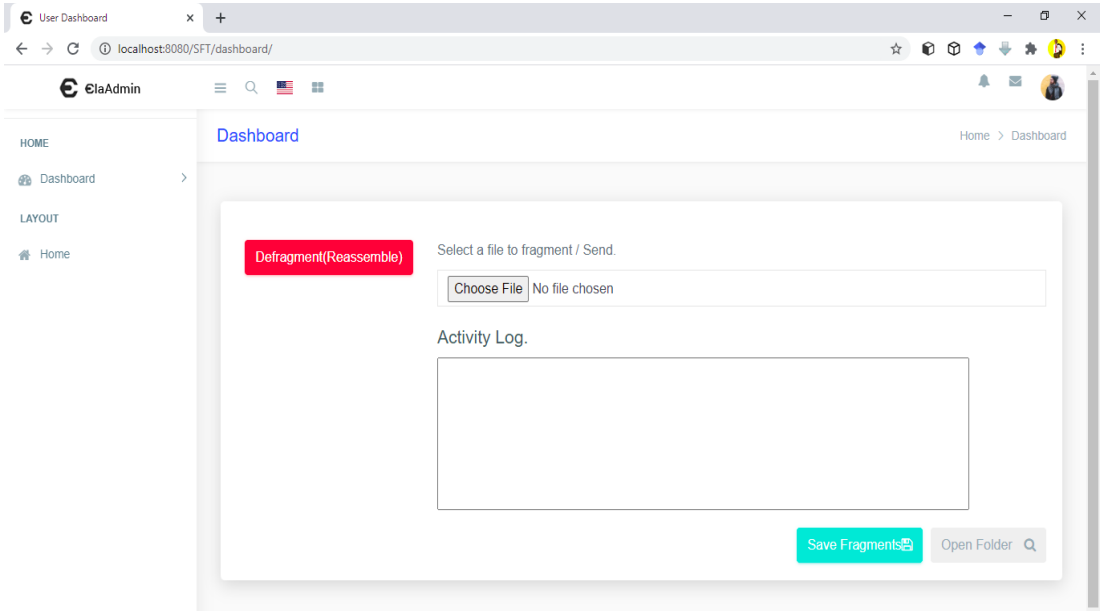


Figure 3: Homepage



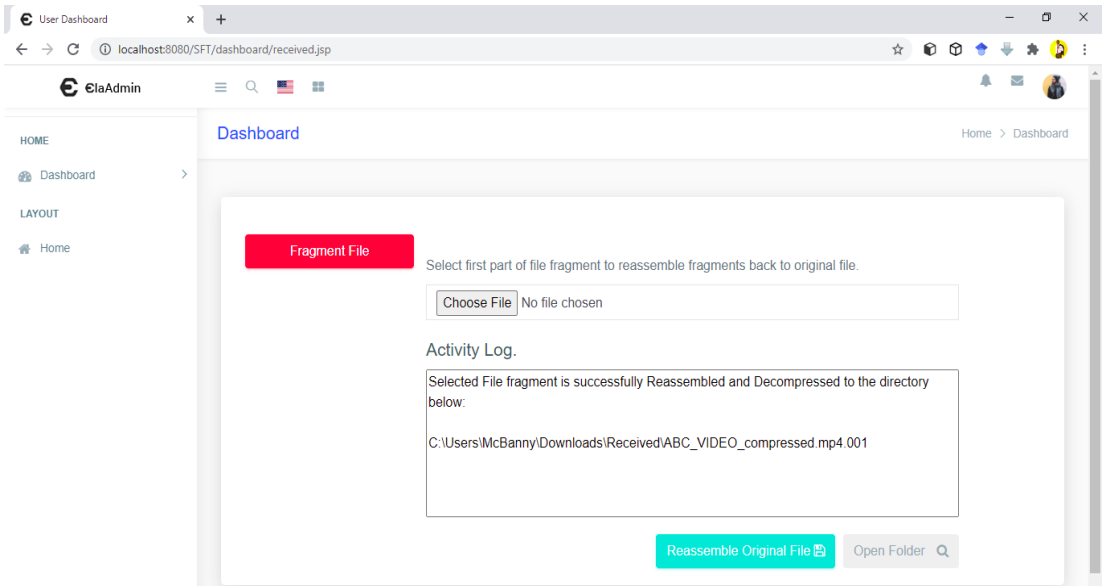


Figure 4: Fragmentation and Reassembly Pages

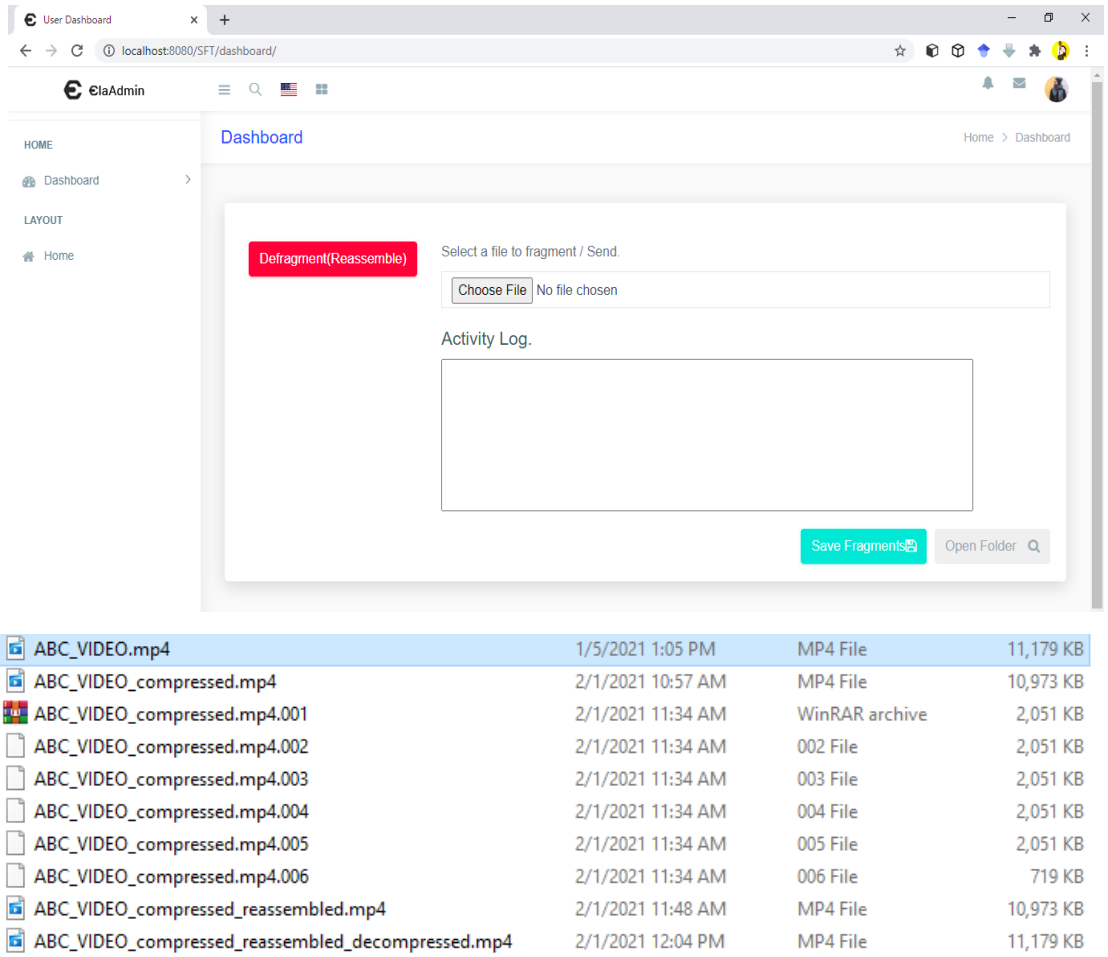


Figure 5: Compression and Fragmentation Result Pages (Folder after compression, reassembling and decompression)

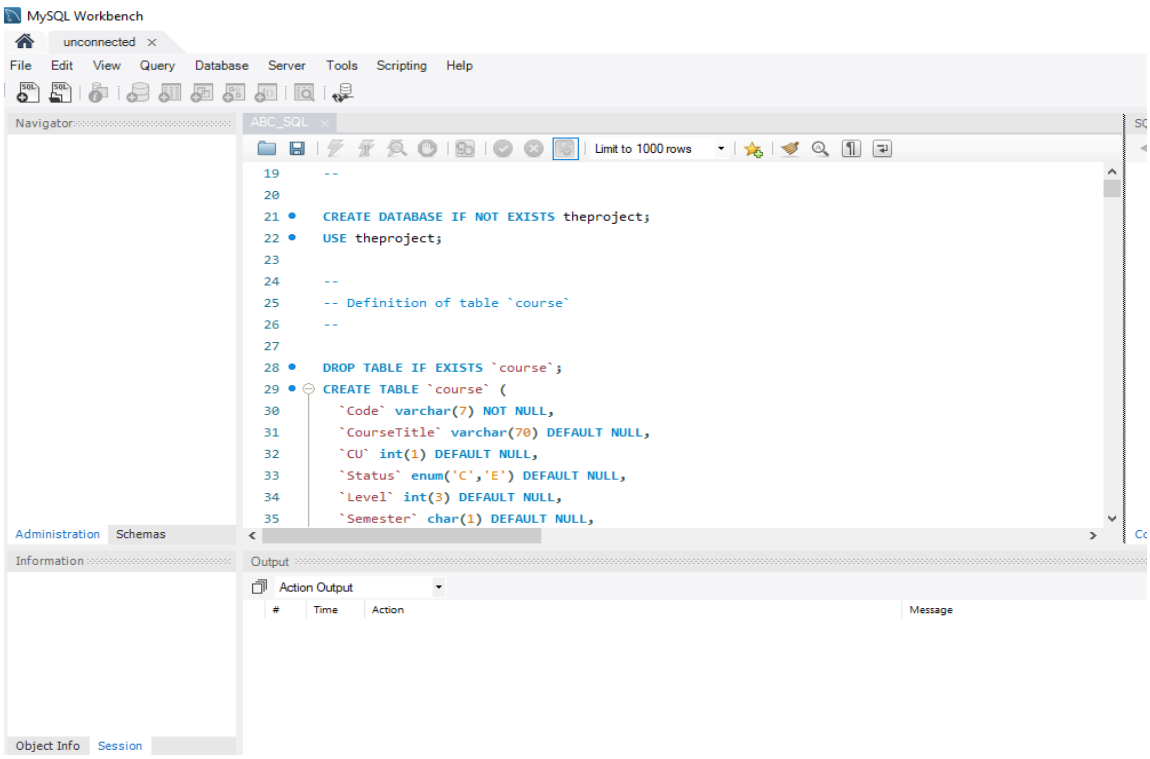


Figure 6: SQL file before compression

Figure 7: SQL file after compression.

The methods of compression, fragmentation, reassembling, and decompression were tested on seven file types, each with a size of 10MB, to evaluate system performance. Various performance metrics were recorded, including the execution time for compression, fragmentation, reassembling, and decompression, the size of the compressed files, and the number of fragments generated after compression. The results indicated that the performance metrics varied depending on the file format, with optimal performance observed for text-based files.

Table 1: Result of file Compression, Fragmentation, Reassembling and Decompression of files

File Types	Original File Size	Compressed Size	No. of Fragments (2MB each except the last)	Reassembled File Size	Time of Execution (Seconds)
Text File (.txt)	10 MB	0.01 MB	1	0.01 MB	< 1s

PDF (.pdf)	10 MB	6.05 MB	4	6.05 MB	< 1s
Database File (.sql)	10 MB	0.72 MB	1	0.72 MB	< 1s
Audio File (.mp3)	10 MB	9.68 MB	5	9.68 MB	< 1s
Video File (.mp4)	10 MB	9.8 MB	5	9.8 MB	< 1s
Image File (.jpg & .png)	10 MB	9.84 MB	5	9.84 MB	< 1s
Document File (.doc)	10 MB	0.01 MB	1	0.01 MB	< 1s

Table 1 presents the results of the compression files. Fragments are created from the compressed version of the file, which are subsequently reassembled into the original compressed file, maintaining the same size, and then decompressed to the original file size of 10MB. Within a file transfer system, the compressed and fragmented sub-files are transmitted independently and are then reassembled and decompressed at the recipient's end.

Chart 1: Graph representing the Original file sizes and their corresponding compressed sizes.

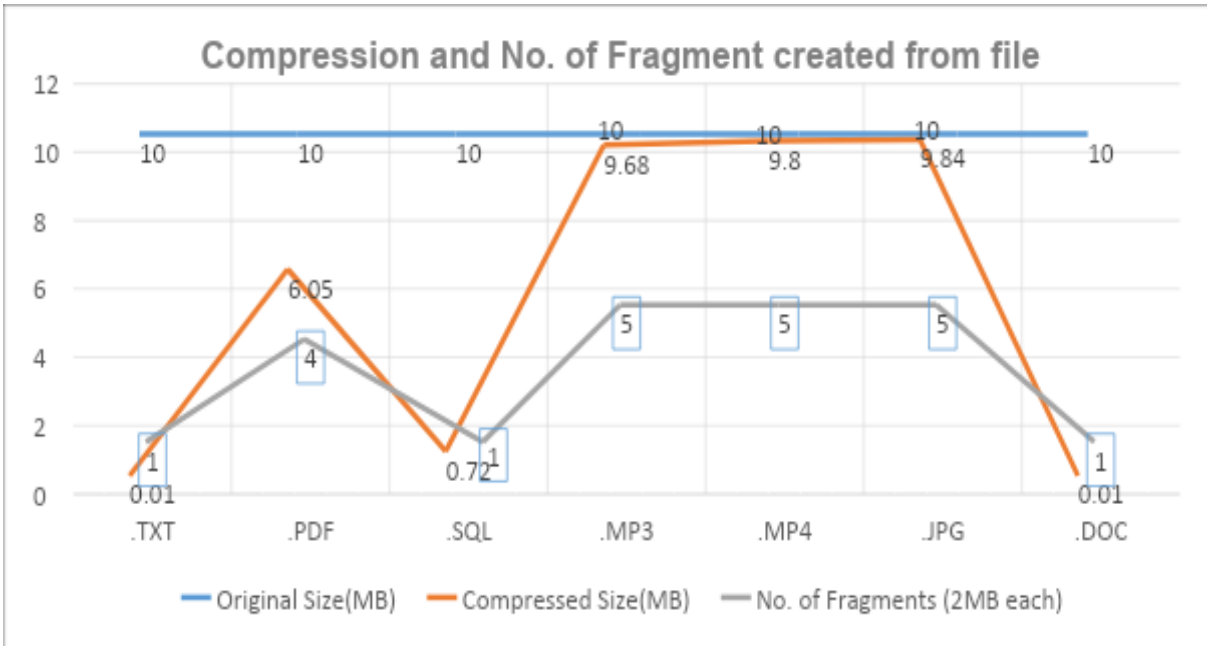


Chart 2: Graph representing the compression performance (reduction of the original file size of 10 MB each) of different file types, the number of fragments created from each compressed file that can be reassembled and decompressed back to the original file.

Graph 3: Graph representing the compression performance (reduction of the original file size of 10 MB each) of different file types.

4.1 DISCUSSION

The research explores the enhancement of file transfer security and efficiency through the combined application of compression, fragmentation, reassembly, and decompression techniques. This integrated approach addresses several critical issues in the field of digital data transmission, particularly in the context of cybersecurity and network throughput efficiency.

4.1.1 Security Enhancement

The integration of compression and encryption techniques significantly bolsters the security of data during transmission and storage. By compressing data before encryption, the encrypted content becomes more resistant to unauthorized access, thereby enhancing protection against potential cyber threats. This layered approach ensures that even if a fragment is intercepted, it remains challenging for malicious entities to decipher the content without the complete data set and the necessary decryption key.

4.1.2 Efficiency and Bandwidth Optimization

Compression reduces the overall size of the data, which directly impacts the efficiency of data transmission. Smaller data sizes require less bandwidth, resulting in faster transmission times and reduced

network congestion. This is particularly beneficial in mobile ad hoc networks (MANETs) and other environments where bandwidth is a limited resource. The reduction in data size also translates to cost savings in data transmission, highlighting the economic advantages of this approach.

4.1.3 Fragmentation and Resilience

Fragmentation enhances the resilience of data transmission by splitting the data into smaller, manageable parts. This method not only facilitates the handling of large files but also increases the robustness of the transmission process. Transmitting fragmented data over disjoint paths reduces the risk of complete data loss in the event of a node compromise or network failure. Moreover, the fragmentation process can be executed without significant consideration of the file content structure, provided an appropriate ordering scheme is employed.

4.1.4 Reassembly and Data Integrity

The reassembly process is critical to ensuring the integrity and completeness of the transmitted data. By using the reverse of the ordering scheme applied during fragmentation, the system can accurately reconstruct the original file from its fragments. This method guarantees that the data received at the destination is identical to the data sent, thus maintaining the reliability of the file transfer process. The study's findings demonstrate that the system performs optimally with text-based files, but further research is needed to assess its effectiveness with other file types.

4.2 Limitations and Future Work

Despite the advantages, the study acknowledges certain limitations. One major drawback is the dependency on a pre-specified ordering scheme for reassembly. Files not fragmented according to this scheme pose significant challenges for reassembly, potentially leading to data loss or corruption. Additionally, the current system may face performance issues with non-text-based files, which warrants further investigation.

Future research should focus on developing more sophisticated algorithms for fragmentation and reassembly that can adapt to various file types and structures. Exploring advanced encryption techniques that work synergistically with compression and fragmentation processes could also enhance security further. Additionally, evaluating the system's performance in real-world scenarios, particularly in diverse and dynamic network environments, would provide valuable insights into its practical applications.

5. CONCLUSION

This research underscores the potential of combining compression, fragmentation, reassembly, and decompression techniques to enhance file transfer security and efficiency. This also constitutes a highly secure method for safeguarding digital files against hijacking, both locally on client machines and during transmission between remote users. The processes of file fragmentation and reassembly can be effectively implemented without significant consideration of the file content structure. This is achieved by specifying an ordering scheme that can also be utilized to reassemble the file fragments back into the original file. However, a notable limitation is that files not fragmented using the pre-specified ordering scheme are challenging to reassemble into their original form.

By addressing both cybersecurity concerns and network throughput challenges, the proposed system offers a comprehensive solution for secure and efficient digital data transmission. Continued advancements in this area promise to further improve the robustness and reliability of file transfer systems, ensuring the safe and efficient handling of digital information in an increasingly interconnected world.

6. ACKNOWLEDGEMENT

The diligent tutelage and support through the process of this research study by Dr Usman Karim is next to none and is laudable. This would not have been a huge success without his help.

Also, to the team of reviewers and editors at the International Journal of Computer Applications (IJCA), their relentless and timely review and professional support in advancing excellent scholarly endeavours in Computing is remarkable.

7. REFERENCES

[1] Rb, M., Chhetri, S., Kc, A., & Jain, H. (2021). Secure File Storage & Sharing on Cloud Using Cryptography. *International Journal of Computer Science and Mobile Computing*, *10*(5), 49–59. https://doi.org/10.47760/ijcsmc.2021.v10i05.005

[2] Carpentieri, B. (2018). Efficient Compression and Encryption for Digital Data Transmission. *Security and Communication Networks*, *2018*, 1–9. https://doi.org/10.1155/2018/9591768

[3] [3] Toradmalle, D., Muthukuru, J., Sathyanarayana, B., & Sri Krishnadevaraya (2019). Compression Techniques: Key to Effective Data Transmission. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(3), 3179–3181. https://doi.org/10.35940/ijrte.C4906.098319

[4] Chidambaram, N., Vineela, K., Jebin, M. J., Balaji, J., & Raj, P. (2018). Fortified File Transfer Framework—A TranSecure Model. *2018 International Conference on Computer Communication and Informatics (ICCCI)*, 1–6. https://doi.org/10.1109/ICCCI.2018.8441210

[5] Kausar, S., Habib, M., Shabir, M. Y., Ullah, A., Xu, H., Mehmood, R., Bie, R., & Iqbal, M. S. (2020). Secure and efficient data transfer using spreading and assimilation in MANET. Software: Practice and Experience, 50(11), 2095–2109. https://doi.org/10.1002/spe.2782

[6] Verma, S., Kapoor, V., & Maheshwari, R. (2019). An Enhanced Cryptographic System for Fast and Efficient Data Transmission. In R. Kamal, M. Henshaw, & P. S. Nair (Eds.), International Conference on Advanced Computing Networking and Informatics (Vol. 870, pp. 287–297). Springer Singapore. https://doi.org/10.1007/978-981-13-2673-8_31

[7] Aamo Iorliam & Harold Kpojime (2019). *Computer Security*. Department of Mathematics and Computer Science, Benue State University, Makurdi, Nigeria.

[8] Jaradat, Abdel-Rahman & Abbadi, Mansour & Nassar, Talha. (2006). A File Splitting Technique for Reducing the Entropy of Text Files. World Academy of Science, Engineering and Technology.

[9] Anandabrata Pal, Husrev T. Sencar, Nasir Memon (2008): *Detecting file fragmentation point using sequential hypothesis testing*. Digital Investigation 5 (2008) S2–S13

[10] File Transfer Protocol, Wikipedia, https://en.wikipedia.org/wiki/File_Transfer_Protocol [accessed 03/02/2021].

[11] H.C.G Leitao & J. Stolfi (2000)**:** *A Multi-Scale method for the reassembly of fragmented objects.* Proc. British Machine Vision Conference – BMVC 2:705 – 714

[12] Ian Sommerville (2016)**:** *Software Engineering*. 10th Edition, Pearson Education Limited, Edinburgh Gate, England, PP:169.

[13] Jayakrishnan Nair, Martin Andreasson **(**2014)**:** *On Channel Failures, File Fragmentation Policies, and Heavy-Tailed Completion Times*. IEEE/ACM Transactions on Networking

[14] Karim Usman, Hyacinth C. Inyiama & Risikatu Karim (2019)**:** *Securing remote backup files from a man in the middle attacks by using encryption and compression mechanisms*. International Journal of Advanced Studies in Computer Science and Engineering, Volume 8, Issue 3.

[15] Muhanad Hayder (2009)**:** *Design and implementation of a file splitter and merger software*. Journal of Kerbala University, Vol. 7 No. 4 Scientific.

[16] Shanmugasundaram, Kulesh & Memon, Nasir. (2003). Automatic Reassembly of Document Fragments via Context Based Statistical Models. Department of Computer and Information Science, Polytechnic University, Brooklyn, New York 11201.

[17] Suchita Tayde & Seema Siledar (2015)**:** *File Encryption, Decryption Using AES Algorithm in Android Phone.* International Journal of Advanced Research in Computer Science and Software Engineering. Volume 5, Issue 5.