Original Research Article

A Smarter Way to Compress and Decompress Data for Cloud Storage

.

ABSTRACT

|  |
| --- |
| The present work contributes an adaptive lossless compression-decompression technique for different data types, such as text, image, audio, and video files. The primary objective is to optimize cloud storage by maximize space savings while maintaining data confidentiality and reducing data loss. A Comparative analysis of experimental research of various compression methods is presented in the paper. The proposed method used search-based compression techniques using linear, binary, and interpolation search, to get effective lossless compression. The performance of method is computed using a dataset of different file formats, observing key parameters, including compression ratio, decompression time and space savings. The results are presented through graphs and tables and compared the computed results with existing methods available in the literature.The method provides a higher compression ratio and faster decompression time as compared to existing methods. It effectively reduces storage space on cloud servers without reducing data integrity. Interpolation-based compression is performing as compared to linear and binary search approaches, achieving up to 85% compression for audio and video files and 79% for text-based files while maintaining information integrity. An efficient and scalable lossless compression method works on different file formats and Its adaptability makes it a practical solution for cloud storage optimization, improving storage efficiency, retrieval speed, and cost-effectiveness. Comparative analysis confirms its superiority over existing techniques, highlighting its potential for large-scale cloud applications. |

*Keywords: Cloud storage; Information retrieval, Search algorithms, Data compression, Data decompression.*

1. INTRODUCTION

From the literature, it is seen that there is tremendous rise in the growth of multimedia contents; therefore, it is essential to manage the cloud storage handling large-scale multimedia contents. The compression approach of data provides significant idea for optimizing the cloud storage and transmission efficiency with data integrity. Some traditional compression techniques, such as Huffman coding, Lempel-Ziv, and Shannon-Fano (SF), improve the cloud storage. The modern cloud storage gives more flexible and scalable methods that work with wide variety of file types, including text, image, audio and video, while ensuring minimal delays and cost.

Several studies had been proposed to boost up the cloud storage, such as Hadoop Distributed File System (HDFS), but it lacks to provide compression method for all file types [1]. Classical compression methods were unfit for cloud data, especially related to medical records, as it requires full data recovery. Lossless method like Lempel–Ziv–Welch (LZW) had utilized for medical data, making sure for complete data retrieval with optimized space [2]. Hadoop Distributed File System (HDFS) reduces similar data but lack in strong compression method. ID-based encryption had been proposed; still its computational complexity was challenging task for big datasets. Self-managed cloud storage improved scheduling but does not work well on compression [3]. Techniques such as Boneh-Lynn-Shacham (BLS) cryptography and Automatic Blocker Protocol (ABP) raise security but lack on storage reduction [4]. Multi-keyword searchable encryption with Locality Sensitive Hashing (LSH) provides easy document access, but its ability to data compression remains untested. Additionally, Convergent Encryption (CE) and Proof of Ownership (PoW) techniques worked deduplication [5] but are ineffective on data compression. Many studies have analyzed the performance of used in compression techniques, such as Huffman, LZ77, and BID, based on compression ratio, speed, and storage efficiency. The methods remain standard in data compression, modern cloud environments that help in all file formats, minimizing cost [6-7].

**Table 1. Compression techniques for text, image, audio and video files**

|  |  |  |  |
| --- | --- | --- | --- |
| **Text/Doc/Pdf** | **Image** | **Audio** | **Video** |
| 7zip, Huffman Coding, Zip, Gzip, Bzip2, Shannon-Fano Coding, Arithmetic Coding, Lempel-Ziv Coding (LZ), Lempel-Ziv-Welsh (LZW), Golomb Coding. | EZW, DCT, Wavelet Transform, Run-length Encoding, Predictive Coding and DPCM, Area Image Compression, DEFLATE, Chain Codes, Adaptive Dictionary Algorithms, Entropy Encoding, Decreasing the Color Space, Transform coding, Chroma Subsampling, Fractal Compression | LPAC (Lossless predictive audio compression), ALAC (Apple lossless audio codec), Huffman shift coding, Intrinsic mode function, WMA (Windows media audio lossless), NERO AAC CODEC, FAAC (Freeware advance audio coder), IEEE 1857.2. | JPEG,MPEG,H.261,H.263, MPEG-1, MPEG-2, MPEG-4, MPEG-7, MPEG-21 and H.264. |

The latest literature highlights the significant gaps in compression methods for cloud storage. It is observed that various data compression techniques are described and some of the techniques are listed in the Table 1. While deep learning-based models [8] had improved efficiency using high computational resources; making unfit for real-time applications. Adaptive algorithms, like as dictionary-based techniques [9], were adaptable but less optimize for multimedia contents. Graph entropy based compression methods had shown good results in less data redundancy, yet implementation on cloud server remains limited [10]. Hybrid encryption-compression techniques combine security with compression, but lead to increased processing cost. Furthermore, domain-specific approaches, such as medical image compression and genomic data compression, had provided high efficiency but are not for multi-format cloud storage [11]. An efficient data compression and storage method had been utilized by Regressive Probabilistic Key Encryption (RPKE) for data encryption and Lempel-Ziv-77-Huffman coding (LZ77-HM) for compressing huge datasets [12]. This method gave significant improvements in compression ratio and rate, but computational overhead with RPKE limit scalability in real-time applications. SCOPe, a unified pipeline focused on optimizing data storage and less costs in cloud environments is also used [13]. With combination of data placement, compression scheme and access pattern-aware data partitioning, SCOPe presented significant cost savings. However, the complexity and accurate access may create challenges for dynamic storage scenarios. A power-efficient image storage solution combines image compression with Super-Resolution Generative Adversarial Networks (SRGAN) which reduces image sizes and reconstructs after retrieval [14]. SRGANs may provide increased computational demands during image reconstruction. A detailed review of several distinct data compression strategies was conducted, including data quality-based, coding-based and application-based and data type-based techniques [15]. The important references provide ideas to create models, methods, tools and security approaches that make software creation easier [16-18]. Existing methods sometimes lacks in compressing bigger text files on handheld devices [19], improving lossless compression over variety of file formats [20], and choosing suitable method for text data [21]. Image compression methods are usually restrict to some formats like BMP [22], while generic file compression struggles domain-specific algorithms and restrict for resource-constrained scenarios [23]. Also, video compression methods like adaptive fuzzy inference systems use evaluation against common codecs to handle impact on quality and efficiency [24]. Cloud storage is well explained in [25-26]. Resolving above gaps, it is important for forming more adaptable and effective compression methods over various applications.

The above-mentioned limitations demonstrate the requirement for adaptive compression methods that emphasize cloud storage across all sorts of data formats while ensuring minimal computational cost. The present work demonstrates a compression-decompression method which shall increase cloud storage, retrieval speed and good performance. Focusing on storage issues such as delays and computational cost, the presented approach is a flexible technique that works well with modern cloud servers. The research work presented in this paper fills the above-mentioned gaps by introducing flexible, efficient and resource-aware model, ensuring improved cloud storage management for various data applications.

2. material and methods

By compressing a file, its size may be minimized and a smaller file size makes it simpler to manage large multimedia contents. By compressing the data, unnecessary information may be eliminated using tools like video conferencing, file storage, and printers. The suggested method makes use of search-based data compression, even if adaptive modeling might not be the best compression methodology for natural language texts in full-text content retrieval scenarios. Data is compressed using search methods like linear, binary, and interpolation searches. Compressing file types like .pdf, .jpg, .txt, .doc, audio and video before storing them on the cloud is a good idea. Once the information is stored, it may be simply accessed. Initially, the compressed information decompresses and is then obtained by the authorized client.

The following compression techniques are used and elaborated below.

## **Linear Search**

## In order to apply the linear search approach, a file must first be converted into binary format. Then, the algorithm searches for 0 and 1 values.

***Algorithm 1:*** *Linear\_Search()*

***Input:***

1. *Select file of format .txt, .pdf, .doc, .jpeg*

*begin*

1. *file \_format=pdf||doc||txt||jpg*
2. *data = content*
3. *(For Compression)*
4. *bin\_value= ' '.join(format(ord(i), '08b') for i in str(data))*
5. *list.sort(bin\_value)*
6. *linear\_Sort()*
7. *rank\_zero=[n for n, val in enumerate(m) if val==0]*
8. *zero = len([0 for x in t if x==0])*
9. *rank\_one = [n for n,val in enumerate (m) if val==1]*
10. *one = len([0 for x in t if x==1])*

***Output:***

1. *file\_size= sys.getsizeof (compressed\_content)byte)*

*end*

1. **Binary Search**

Binary Search is an effective technique for narrowing your choices from an ordered set of options. It continuously bisects the portion of the list that may contain an item until the item's position is collapsed. The binary search algorithm is given below:

***Algorithm 2:*** *Binary\_Search()*

***Input:***

1. *Select file of format txt, pdf, doc, jpeg*

*Begin*

1. *file \_format=.pdf||.doc||.txt||.jpg*
2. *data = content*

*(For Compression)*

1. *bin\_value= ' '.join(format(ord(i), '08b') for i in str(data))*
2. *list.sort(bin\_value)*
3. *binary\_search(arr,low=none,high=none)*
4. *if arr[high]==0:*
5. *return 0*
6. *if arr[low]==1:*
7. *return high-low+1*
8. *return binary\_sort(arr,low,mid)+binary\_sort(arr,mid+1,high)*

***Output:***

1. *result=binary\_search(arr)*
2. *No\_of\_ ones= result*
3. *a=l-result*
4. *No\_of\_zeros=a*
5. *print(file is compressed)*

*end*

1. **Interpolation Search**

When the values in a sorted array are evenly distributed, a new technique called as the interpolation search may be applied. In addition to being quicker and more accurate than binary search, interpolation search also avoids becoming caught in local minima. The algorithm for interpolation search is given below:

***Algorithm 3:*** *Interpolation\_Search()*

***Input:***

1. *Select file of format txt, pdf, doc, jpeg*

*Begin*

1. *file \_format=.pdf||.doc||.txt||.jpg*
2. *data = content*
3. *(For Compression)*
4. *def interpolation\_Search(A, result):*
5. *while A[high] != A[low] and A[left] <= target <= A[high]:*
6. *cen = low + (result - A[low]) \* (high -low) // (A[high] - A[low])*
7. *if result == A[cen]*
8. *return cen*
9. *elif result < A[cen]:*
10. *right = cen – 1*
11. *else:*
12. *low = cen + 1*
13. *if result == A[low]:*
14. *return low*
15. *index = interpolationSearch(A, key)*

***Output:***

1. *if index != -1:*
2. *no\_of\_one = [l for l,value in enumerate (lst) if value==1]*
3. *no\_of\_zero = [i for l,value in enumerate (lst) if value==0]*
4. *print(file is compressed)*

*end*

Method used to put back the compressed data in its original form is decompression. There are several compression methods stated and each one required decompression for data. The following algorithm is used to decompress any file using linear, binary, and interpolation search for compression.

***Algorithm 5:*** *Decompression()*

***Input:***

1. *To decompress, choose the input*
2. *file.path = filedialog.askopenfilename()*
3. *name=Path(path).stem*
4. *file\_extension = pathlib.Path(path).suffix*

***Ouput:***

1. *new\_path=shutil.copy (path1,'C:/Users/Hp/Desktop/Download')*
2. *print("File is downloaded")*

*end*

Audio video in digital form, have high quality and large file size. This makes necessary to compress audio and video file to make more space. Below is the ZAV-compression algorithm helps in compression and decompression of audio video file without damaging the quality of file.

***Algorithm 6:*** *Compression\_Audio\_Video\_File()*

***Input:***

1. *Select file format of mp3, mp4*
2. *Begin*
3. *fileout = "compressed\_file"*

*(For Compression)*

1. *with open(filein, mode="rb") as fin,*
2. *open(fileout, mode="wb") as fout:*
3. *content = fin.read()*
4. *compress\_file=zlib.compress(content, zlib.Z\_COMPRESSION)*
5. *print (f"Compressed size: {sys.getsizeof(compressed\_file)} bytes")*

*(For Decompression)*

1. *with open(fileout, mode="rb") as fin:*
2. *content = fin.read()*
3. *Decompressed\_file = zlib.decompress (content)*
4. *print(f"Decompressed size: {sys.getsizeof(compressed\_data)} bytes")*

*end*

3. results and discussion

In the present work, the performance of proposed searching methods is evaluated. There are three searching techniques i.e. linear, binary and interpolation used on different file formats such as text,.doc., pdf, .jpg, audio and video. The dataset is used which consist of file types including text (.txt, .pdf, .doc), images (.jpg), audio (.mp3), and video (.mp4), ranging 5,10,20,25 MBs, representing real-world applications. Dataset consists of both structured and unstructured data such as reports, academic papers, text and PDFs. Images in the dataset include high-resolution photographs and compressed web images. Audio files are of different bit-rates and formats and video files include standard-definition (SD), high-definition (HD), and ultra-high-definition (UHD) formats. The datasets are designed to cover practical cloud storage environments.

The study evaluates the efficiency using following different key parameters.

1. **Compression Ratio (CR):** Data is compressed effectively and may be measured by finding compression ratio. The percentage file size is reduced after compression and higher percent gives better efficiency. The ratio is computed by following formula:

CR = $\frac{Uncompressed file bits}{Compressed file bits}$ (1)

1. **Compression Time (CT):** It is time required for compressing the file using proposed method. It is usually measured in millisecond or seconds.
2. **Decompression Time (DT)**: It is time required for decompressing the file using proposed method. It is usually measured in millisecond or seconds.

In the Table 2., the results of linear, binary and interpolation search compression are summarized, showing the effectiveness of compression and the time is required for compression and decompression in terms of different file sizes and likewise in Table 3., the audio and video file's compression ratio, compression and decompression time are shown. Table 2., shows that the interpolation based search method gives high compression ratio with lower compression and decompression time in comparison to linear and binary based search techniques. Text, Pdf’s and document files compression ratios are reduced to 79%, image files achieves upto 83% without losing data integrity. From Table 3., it is shown that audio and video files are reduced up to 85% and 83% after compression method. From both the tables, it is clear that suggested strategy offers a superior compression rate for cloud storage while keeping quality. Low compression and decompression time helps to access files faster, which improves the overall performance of cloud server. Key strength of the said techniques is adaptability over multiple file formats, which makes suitable for real world application. Some techniques compromise the quality of data while proposed method ensures lossless compression. Also, it is reduced the file size transfer quicker with minimum latency in storage and retrieval process. The advantage of proposed method shows the practical and scalable quality for handling large files over cloud storage.

**Table 2. Computation of compression and decompression for document and image files**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type of File** | **File Size (MB)** | **Compression ratio (%)** | **Compression time (sec)** | **Decompression time (sec)** |
| **Linear** | **Binary** | **Inter-pola-tion** | **Linear** | **Binary** | **Inter-pola-tion** | **Linear** | **Binary** | **Inter-pola-tion** |
| **.txt** | 5 | 45 | 47 | 49 | 35 | 22 | 18 | 33 | 21 | 16 |
| 10 | 46 | 48 | 51 | 38 | 23 | 19 | 35 | 23 | 17 |
| 15 | 48 | 50 | 53 | 43 | 25 | 23 | 41 | 23 | 20 |
| 20 | 50 | 56 | 54 | 47 | 29 | 25 | 43 | 26 | 22 |
| 25 | 53 | 58 | 57 | 49 | 32 | 27 | 46 | 30 | 24 |
| **.doc** | 5 | 54 | 56 | 59 | 30 | 21 | 15 | 28 | 19 | 14 |
| 10 | 57 | 58 | 62 | 32 | 23 | 18 | 30 | 20 | 17 |
| 15 | 61 | 66 | 68 | 36 | 26 | 19 | 35 | 24 | 19 |
| 20 | 63 | 69 | 73 | 39 | 28 | 22 | 37 | 26 | 20 |
| 25 | 66 | 71 | 76 | 41 | 30 | 25 | 39 | 30 | 23 |
| **.pdf** | 5 | 51 | 53 | 56 | 5 | 4 | 2 | 5 | 4 | 2 |
| 10 | 62 | 64 | 65 | 8 | 6 | 4 | 7 | 5 | 3 |
| 15 | 65 | 68 | 72 | 10 | 8 | 5 | 8 | 7 | 4 |
| 20 | 67 | 70 | 76 | 13 | 9 | 7 | 12 | 8 | 6 |
| 25 | 69 | 75 | 79 | 18 | 15 | 10 | 15 | 11 | 8 |
| **.jpg** | 5 | 65 | 67 | 69 | 5 | 4 | 2 | 5 | 3 | 2 |
| 10 | 68 | 72 | 75 | 7 | 5 | 3 | 6 | 5 | 2 |
| 15 | 70 | 73 | 78 | 9 | 8 | 7 | 7 | 7 | 6 |
| 20 | 75 | 77 | 80 | 11 | 10 | 8 | 9 | 9 | 6 |
| 25 | 79 | 81 | 83 | 13 | 12 | 10 | 12 | 11 | 9 |

**Table 3 Computation of compression and decompression time for audio-visual files using ZAV compression method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type of File**  | **File size (MB)** | **Compression ratio (%)** | **Compression time (sec)** | **Decompression time (sec)** |
| **audio** | 5 | 75 | 8 | 6 |
| 10 | 78 | 12 | 11 |
| 15 | 81 | 15 | 14 |
| 20 | 83 | 20 | 18 |
| 25 | 85 | 24 | 21 |
| **video** | 5 | 70 | 15 | 17 |
| 10 | 74 | 19 | 23 |
| 15 | 76 | 22 | 27 |
| 20 | 81 | 26 | 29 |
| 25 | 83 | 31 | 34 |

**Table 4.Comparison between different metrics of different techniques**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of File** | **File size (MB)** | **Existing methods** | **Proposed method** **(Interpolation search)** |
| **Compression Ratio (%)** | **Compression Time(sec)** | **Decompression Time(sec)** | **Compression Ratio (%)** | **Compression Time(sec)** | **Decompression Time(sec)** |
| **text [19]** | 1.879(SF) | 38.10 | 0.062 | - | 46.0 | 0.055 | 0.052 |
| 3.132(SF) | 38.02 | 0.101 | - | 42.0 | 0.089 | 0.082 |
| 3.871(SF) | 33.67 | 0.127 | - | 41.0 | 0.119 | 0.111 |
| **average** |  | **36.59** | **0.096** | - | **43.0** | **0.087** | **0.081** |
| **doc [20]** | 0.031(SF) | 44.5 | 0.000 | - | 46.3 | 0.010 | 0.013 |
| 0.048(SF) | 67.0 | 0.015 | - | 73.0 | 0.014 | 0.017 |
| 0.210(SF) | 77.7 | 0.031 | - | 92.9 | 0.029 | 0.052 |
| **average** |  | **63.0** | **0.015** |  | **70.73** | **0.017** | **0.027** |
| **pdf [21]** | 0.039(SF) | 59.2 | 12.600 | 12.876 | 63.3 | 10.738 | 11.580 |
| 0.071(SF) | 62.6 | 42.940 | 32.845 | 68.6 | 36.858 | 31.345 |
| 0.118(SF) | 63.7 | 153.810 | 114.180 | 64.5 | 95.755 | 93.328 |
| **average** |  | **61.8** | **69.7** | **53.3** | **65.4** | **37.783** | **45.417** |
| **image [22]** | 2.7(FC) | 85.62 | 0.078 | - | 89.0 | 0.051 | 0.047 |
| 4.8(FC) | 84.50 | 0.156 | - | 90.0 | 0.137 | 0.127 |
| **[23]** | 8.47(Deflate) | 1.404 | - | - | 63.0 | 0.474 | 0.432 |
| 9.97(Deflate) | 2.714 | - | - | 57.0 | 0.587 | 0.524 |
| **average** |  | **43.55** | **0.058** |  | **74.75** | **0.312** | **0.282** |
|  |  | **ZAV Compression Method** |
| **audio[20]** | 1.738(SF) | 94.0 | 0.109 | - | 95.0 | 0.890 | 0.170 |
|  | 4.790(SF) | 76.0 | 0.250 | - | 87.0 | 0.200 | 0.240 |
|  | 9.580(SF) | 94.1 | 0.515 | - | 97.0 | 0.450 | 0.430 |
| **average** |  | **88** | **0.29** |  | **93.0** | **0.513** | **0.280** |
| **video[24]** | 1.468(IFC) | 27.68 | - | - | 67.12 | 5.810 | 5.180 |
| 2.600(IFC) | 25.30 | - | - | 78.63 | 6.890 | 6.590 |
| 4.418(IFC) | 26.04 | - | - | 81.02 | 8.458 | 8.106 |
| **average** |  | **26.34** | **-** | **-** | **75.59** | **7.052** | **6.625** |

Table 4., is a comparison table between existing algorithms available in the literature and present approaches. For text, pdf, doc and audio file the existing algorithm i.e., Shannon-fano algorithm is used, while for image file, two different methods like Deflate and Fibonacci code (FC) based methods are used and for video files, inter-frame-based compression (IFC) is used against the proposed method. For text, doc, pdf, existing method shows average compression ratio of 36.59%, 63%, 61.8%, respectively whereas proposed method improve it to 43%,70.73%, 65.4%, respectively for providing faster processing speed. For image files of size 2.7MB and 4.8MB files, Fibonacci method is used and for 8.4MB and 9.9MB Deflate method is used but interpolation method outperforms by 74.75% compression ratio speed. While in audio and video files, it achieved 93.0% and 75.59% compression ratio, giving excellent performance for securing playback quality. Major advantage of the proposed method is its versatility which makes it suitable for cloud storage diversity. Thus, the proposed methods are ideal methods for service provider to minimize storage and keeping excellent quality of and accessibility. Comparative analysis shows the continuous outperformance of proposed method in terms of speed, latency and storage efficiency. Reduced file size is not only improve cloud management but also enhance data transfer in lesser time in cloud based environment.

4. ConclusionS

The presented work examined the compression algorithms for text, pdf, doc, images, audio and video files. A file must first be chosen before being compressed and sent over the cloud server. Similarly, for decompression, the uploaded file is first converted to its original state before being downloaded. Linear, binary and interpolation search methods are used for compressing the data. Based on the experimental findings, every search is conducted using files of various sizes and file types. In the proposed search strategies, compression ratio, compression and decompression time are compared to one another. The outcomes demonstrate that the suggested plan provides the optimum search for files in minimum computation time. However, a study of compression ratio revealed that interpolation search is the most effective in compression while the other two binary and linear search methods are equally effective. Further, the interpolation is compared with existing technique mentioned in the literature. In result, interpolation again gives the most effective results, when considering the time from techniques used during compression and decompression and compression ratio. The strength of present research is adaptation on different file types and quality to reduce storage problem without data loss. Unless earlier studies that work on format-specific compression, this work presents a compression model that enhances cloud storage efficiency. Moreover, the direct comparison with existing techniques further substantiates its effectiveness in real-world applications. Future work is to optimization strategy to reduce the computational complexity of the proposed technique, making it more scalable for large-scale cloud environments. It may be possibility to Integrate with machine learning models which may create most efficient compression technique and may lead to further enhancement in speed. Another improvement is the extension of presented approaches for ensuring both compression efficiency and data security. The work delivers storage optimization by giving lossless and efficiency compression methods that balances space savings, speed, and scalability. In result, it motivates future research into smart compression method that may further refine data storage and retrieval processes in cloud server.

**COMPETING INTERESTS DISCLAIMER:**

Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

Option 2:

Author(s) hereby declare that generative AI technologies such as Large Language Models, etc. have been used during the writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

Details of the AI usage are given below:

1.

2.

3.

References

1. Widodo, R.N., Abe, H., Kato, K (2020). HDRF. In Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems 2020 Aug 24. 122-129 ACM. <https://doi.org/10.1145/3409963.3410500>.
2. Abdo, A., Karamany, T.S. &Yakoub, A (2024). A hybrid approach to secure and compress data streams in cloud computing environment. Journal of King Saud University-Computer and Information Sciences, 36(3), 1-14. <https://doi.org/10.1016/j.jksuci.2024.101999>.
3. Yang, Y., Chen, Y., Chen, F. & Chen, J (2022). An efficient identity-based provable data possession protocol with compressed cloud storage. IEEE Transactions on Information Forensics and Security. 17, 1359-71. <https://doi.org/10.1109/tifs.2022.3159152>.
4. Jalil, B.A., Hasan, T.M., Mahmood, G.S. and Abed, H.N (2022). A secure and efficient public auditing system of cloud storage based on BLS signature and automatic blocker protocol. Journal of King Saud University-Computer and Information Sciences, 34(7), 4008-21. https://doi.org/10.1016/J.JKSUCI.2021.04.001.
5. Ahmad, S., Arif, M., Ahmad, J., Nazim, M. and Mehfuz, S (2024). Convergent encryption enabled secure data deduplication algorithm for cloud environment. Concurrency and Computation: Practice and Experience. 36(21), 1-28. <https://doi.org/10.1002/cpe.8205>.
6. Diop, B.I., Gueye, A.D. & Diop, A (2022). Comparative study between different algorithms of data compression and decompression techniques. In Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences, 737-744. <https://doi.org/10.1007/978-981-19-8742-7_59>
7. Addepalli, P.S., Lakshmi, P.V (2024). Effective medical data compression for minimal cloud storage using versatile compression techniques. In International Conference on Algorithms and Computational Theory for Engineering Applications. 43-51. <https://doi.org/10.1007/978-3-031-72747-4_7>.
8. Gupta, M., Agrawal, P (2022). Compression of deep learning models for text: A survey. ACM Transactions on Knowledge Discovery from Data. 16(4), 1-55. <https://doi.org/10.1145/3487045>.
9. Vishwanath, B., Nanjundaswamy, T. & Rose, K (2021). Effective prediction modes design for adaptive compression with application in video coding. IEEE Transactions on Image Processing. 31, 636-47. <https://doi.org/10.1109/TIP.2021.3134454>.
10. Gu, S., Hou, J., Zeng, H., &Yuan, H(2020). 3D point cloud attribute compression via graph prediction. IEEE Signal Processing Letters. 27, 176-80. <https://doi.org/10.1109/LSP.2019.2963793>.
11. Wahab, O.F., Khalaf, A.A., Hussein, A.I. and Hamed, H.F (2021). Hiding data using efficient combination of RSA cryptography and compression steganography techniques. IEEE access. 9, 31805-31815. <https://doi.org/10.1109/ACCESS.2021.3060317>.
12. Pinnapati, S. and Shivanna, P (2024). An efficient data compression and storage technique with key management authentication in cloud space. Indonesian Journal of Electrical Engineering and Computer Science. 35(3), 1-17. doi:10.11591/ijeecs.v35.i3.pp1680-1687.
13. Mukherjee, K., Shah, R., Saini, S., Singh, K., Kesarwani, H., Barnwal, K. and Chauhan, A (2023). Towards optimizing storage costs on the cloud. In2023 IEEE 39th International Conference on Data Engineering (ICDE), 2919-2932, 1-14. IEEE. doi:10.1109/ICDE55515.2023.00223.
14. Mondal, A., Singh, S (2024). Power-efficient image storage: leveraging super resolution generative adversarial network for sustainable compression and reduced carbon footprint. arXiv preprint arXiv:2404.04642, 1-15. doi:<https://doi.org/10.48550/arXiv.2404.04642>
15. Jayasankar, U., Thirumal, V., Ponnurangam, D (2021). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. Journal of King Saud University-Computer and Information Sciences. 33(2), 119-40. doi:<https://doi.org/10.1016/j.jksuci.2018.05.006>.
16. Gautam, D., Saxena, V (2023). Secure exchange of IMSI number between sender and receiver. Kepes. 21(3), 750-62. doi:<https://doi.org/10.6084/m9.figshare.24182547#128>.
17. Gautam, D., Saxena, V (2023). Optimization of storage of cloud servers through binary search algorithm. In2023 IEEE 7th Conference on Information and Communication Technology (CICT) 1-6. doi:[10.1109/CICT59886.2023.10455361](https://doi.org/10.1109/CICT59886.2023.10455361).
18. Gautam, D., Rimer, S. & Saxena, V (2024). Secure access of folders and files after removal of duplicacy over the cloud. Int. J. of Computer Network and Information Security. 16(1), 48-60. 10.5815/ijcnis.2024.01.04. doi: 10.5815/ijcnis.2024.01.04
19. Mantoro, T., Ayu, M.A. &Anggraini, Y (2017). The performances of text file compression using Shannon-Fano and Huffman on small mobile devices. In 2017 International Conference on Computing, Engineering, and Design (ICCED) 1-5. doi:10.1109/CED.2017.8308127.
20. Sharma, K., Gupta, K (2017). Lossless data compression techniques and their performance. In 2017 International Conference on Computing, Communication and Automation (ICCCA) 256-261. doi:[10.1109/CCAA.2017.8229810](https://doi.org/10.1109/CCAA.2017.8229810).
21. Kodituwakku, S.R., Amarasinghe, U.S (2010). Comparison of lossless data compression algorithms for text data. Indian journal of computer science and engineering, 1(4), 416-25.
22. Hardi, S.M., Angga, B., Lydia, M.S., Jaya, I. &Tarigan, J.T (2019). Comparative analysis run-length encoding algorithm and fibonacci code algorithm on image compression. In Journal of Physics: Conference Series, 1235(1). doi:10.1088/1742-6596/1235/1/012107
23. Gupta, A., Bansal, A. & Khanduja, V(2017). Modern lossless compression techniques: review, comparison and analysis. In 2017 second international conference on electrical, computer and communication technologies (ICECCT), 1-8. doi:10.1109/ICECCT.2017.8117850.
24. Putra, A.B., Gaffar, A.F. &Wajiansyah, A. & Qasim, I.H (2018). Feature-based video frame compression using adaptive fuzzy inference system. In 2018 International symposium on advanced intelligent informatics (SAIN), 49-55. doi:[10.1109/SAIN.2018.8673386](https://doi.org/10.1109/SAIN.2018.8673386).
25. Kumar, J., Kumar, H., Singh, K.V. and Saxena, V. (2024) Secure Data Storage and Retrieval over the Encrypted Cloud Computing, International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.4, pp.52-64, 2024. doi:10.5815/ijcnis.2024.04.04
26. Kumar, J., and Saxena, V. (2021) Asymmetric Encryption Scheme to Protect Cloud Data Using Paillier-Cryptosystem, *IJAEC* vol.12, no.2 2021: pp.50-58. doi:https://doi.org/10.4018/IJAEC.2021040104