2024 Data Compression Conference (DCC) | 979-8-3503-8587-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/DCC58796.2024.00052

Light-weight Steganography for HPC Lossy Compression

Ruiwen Shan and Jon C. Calhoun

Holcombe Department of Electrical and Computer Engineering - Clemson University
433 Calhoun Dr
Clemson, SC, 29634, USA
rshan@clemson.edu and jonccal@clemson.edu

Abstract

The explosive data growth in high-performance computing (HPC) puts pressure on systems to process huge amounts of data and poses threats to the security of important data during transmission. Traditional data protection methods such as encryption inevitably attracts intermediate intercepting entities' attention. As a means of hiding information within other irrelevant data (carrier), steganography is used to transmit critical data without arousing the attention of regulators. Error-bounded lossy compression is a data reduction technique that effectively alleviates system pressures due to large data volumes. In this paper, we propose a steganography scheme based on the lossy compressor SZ named StegaZ. StegaZ performs steganography while compressing data by selecting random bits for insertion based on the password entered by the user. StegaZ does not affect unaware users' normal usage of the decompressed dataset. The experimental results show StegaZ preserves more than 99.6% of the original compression ratio and achieves a PSNR of 100% when selecting an appropriate dataset. Additionally, it imposes minimal compression bandwidth overhead, sometimes even able to obtain a higher compression bandwidth than the original.

1 Introduction

In the past few decades, the academic communities have relied on scientific computing techniques to study problems involving large-scale data computation and simulation. HPC has become an integral part of scientific research due to its robust data processing abilities. Such systems provide abundant computation capabilities while reducing operation and maintenance costs. However, the use of HPC systems also poses challenges. One of the most prominent issues is the increasing volume of scientific data generated in experiments. Cosmological simulations such as HACC [1] can produce over 20 PB of data per snapshot when run at the trillion-particle scale. The broad range of problem domains often requires vast amounts of data to ensure the simulation process runs smoothly. The enormous amount of data consumes a significant amount of storage space and also leads to I/O bottlenecks during data transmission. For example, the Linear Coherent Light Source (LCLS) at SLAC National Accelerator Laboratory [2] generates X-ray imaging data at a rate of 250 GB/s, which is then transmitted to data centers for further analysis. But the current standard bandwidth for mainstream exascale HPC interconnection networks is around 200 Gbps [3].

Data compression is a promising approach to alleviate the problem of overwhelming system resources caused by excessive data volume. However, due to the multidimensional and highly random nature of scientific datasets, conventional deduplication or lossless compression methods are of limited use when shrinking HPC floating-point values. In fact, most lossless compressors can only achieve a compression ratio (CRs) of 2–4× when working with scientific datasets [4]. In contrast, error-bounded lossy compression (EBLC) is capable of achieving higher CRs with accurate error control. For state-of-the-art EBLC such as SZ [5] and ZFP [6], CRs of 10–1000× is achievable in real-world applications while tightly adhering to user-specified error bounds (eb).

Despite the fact that EBLC performs admirably in compression of HPC datasets, the adoption of compression algorithms also poses security threats such as information leakage. CR is a side channel that leaks vital information by providing a reference to the content and internal structure of the compressed data file, making it susceptible to side-channel attacks [7]. In addition, the compression algorithm is computationally vulnerable and has limited resistance to a single bit-flip or data corruption [8, 9]. Besides, sensitive data are also processed and transferred in HPC systems, including personal information, scientific research results, and intellectual property protected data. The loss of these data hinders the research process and even affects political decisions. Thus, it is imperative to protect the security and integrity of critical data.

Data protection strategies such as encryption are commonly used to handle issues related to confidentiality as well as integrity authentication. Although encryption is an effective method, there are some limitations. Encryption is not stealthy. The use of this mechanism is perceivable as it visually affects the content, thus making the communication behavior easily detectable and enabling attackers to follow the trail. Besides, encryption cannot resolve the problem of content authentication. Data encryption cannot protect the content and property rights of transmitted data. The presence of an encryption key does not guarantee that the message has not been altered during transmission. Instead of encrypting transmitted data , it is preferable to utilize steganography methods to hide the secret information directly and covertly in other carriers to avoid attracting the attention or suspicion of eavesdroppers.

The popularity of HPC systems and collaborative research across multiple parties has led to the need to ensure the secure transmission of confidential data across these systems. Previous studies have focused on using encryption techniques to protect critical data [10]. However, in some cases, an appropriate solution is to transmit sensitive data by covert means to avoid attracting the attention of untrusted third parties. As a commonly used method to improve the efficiency of data transmission and storage in HPC, compression can serve as an effective mask for steganographic techniques. Our contributions are summarized as follows:

- We analyze the state-of-the-art lossy compressors for HPC data and develop a new steganographic approach, StegaZ. To the best of our knowledge, this is the first steganography scheme specialized for lossy compression in HPC environments. StegaZ randomly embeds secret data without significantly increasing the burden on the HPC system, while ensuring the retention of more than 99.6% of the original CR and 100% PSNR when selecting an appropriate dataset.
- StegaZ enables informed users to quickly extract secret data, while normal users still correctly decompress and use the datasets. Using a password provides dual security, preventing intruders who are aware of hidden content from extracting and restoring the inserted content, thereby ensuring the security of critical data.

2 Background

Lossy compression: Obtaining a theoretical upper limit for the CR of lossy compression is challenging because the amount of data to be compressed in lossy compression is uncertain [11]. A completely different result can be achieved depending on the way the problem is abstracted, or the error bounds are set. The main goal of EBLC is to minimize the size of datasets within the error bound eb that the user stipulates, with SZ being one of the most advanced lossy compressors [5]. The overall process of SZ compression is divided into four steps: data prediction, linear quantization, variable-length encoding, and lossless compression. SZ processes the input dataset D element-by-element. For a given element $x_i \in D$, SZ selects the most appropriate prediction function based on the previously processed data elements to obtain the predicted value $x_i^{'}$. Depending on the choice of prediction function, there might be additional data to save. In the case of linear regression, it is necessary to save the regression coefficients. The additional data is compressed separately later. Then, SZ quantizes the difference between the predicted values $x^{'}$ and the original data x into a series of integers and encodes them using Huffman encoding. In the last step, SZ leverages lossless compression methods to further shrink the file size.

Steganography: A technique for hiding information. The purpose of this technology is to protect confidential communications by embedding messages imperceptibly in a carrier whose contents are accessible to all. A wide variety of carrier types are available. The most common at this time are images, text, and video. The difference between data encryption and steganography is that in the former case, third parties are aware that the data is encrypted but do not know the contents before encryption, while the latter case focuses on making the existence of the secret data undetectable.Unlike encryption, steganography can withstand a certain degree of bit-flipping.Bit-level substitution is one of the popular choices for inserting secret data. However, such algorithms may lead to a significant drop in compression results, especially in CR. Section 4 discusses and analyzes this issue.

Motivation: HPC systems need to handle data multiple orders of magnitude larger than ordinary IT systems, traditional security measures are difficult to implement [12]. Data in HPC systems also face the threat of being intercepted and eavesdropped. Specifically, some data processed in the system may involve individual privacy, and sometimes may even interfere with political decision-making. Thus, it is imperative to pay attention to data security and confidentiality during its transmission and storage. Previous research combined HPC compression and encryption to improve data security while protecting only the most sensitive data [10]. Encryption only masks the content of the data, not the fact that the ciphertext is being transmitted. Data encrypted in this manner can easily be observed. Moreover, the highly identifiable nature of the encrypted message is also likely to attract the attention of a regulatory body, thereby blocking any communication of a similar type in the future. Steganography can prevent anyone other than the intended recipient from knowing that a message is being delivered. As the hidden transmission is embedded in a carrier that appears to be unrelated, it is more disorienting. Research combining lossy compression and steganography has mainly focused on image compression, whereas the data processed in HPC environments are massive floating-point values. Typical image steganography algorithms embed the secret message by modifying the discrete cosine transform(DCT) functions, e.g. compression resistant principle-based adaptive steganography(CPRAS) [13]. These schemes only consider the visual invisibility of steganography, however, it is not sufficient in the HPC environment.

Challenge: The use of popular substitution steganography methods can pose several difficulties when applied to the EBLC algorithm. 1) Finding the proper position. In the EBLC algorithm, the information represented by each value is lost to varying degrees depending on the *eb* set by the user. There is no guarantee that the hidden information will remain intact after compression. Furthermore, since most of the values in scientific datasets are floating point numbers, SZ is a lossy compression algorithm developed to handle this situation. The bit representation of floating-point values differs from other forms in that the mantissa bit determines the fractional part, making it difficult to locate the lowest bit that fits within the user-specified eb. Even if the lowest bit can be identified in some cases, a higher bit may need to be replaced when the *eb* is too loose, leading to a large variation in the value itself. Therefore, it is crucial to find the appropriate location to apply steganography. 2) Steganography may negatively impact compression results. In HPC systems, Data security needs to be guaranteed while high performance is ensured. Even if only the lowest bits are replaced, the original CRs or PSNR might be greatly affected (See Section 4).

3 Method

3.1 Proper position

To minimize steganography's impact on compression, the ideal is one that takes up a relatively small amount of memory and is not affected by user-specified *eb*. During quantization, SZ quantifies the difference between the predicted and original values into a series of integers (quantization index). Fully utilizing this feature makes it possible to convert the steganography problem of lossy compression of floating-point values into the steganography problem of lossless compression of integers. A viable option is to perform steganographic methods after quantification.

To ensure that compression results are not excessively affected, it is important to avoid adding steganographic operations directly to the dataset. We notice that SZ stores some extra data when it selects certain predictors for data blocks. For example, when a linear regression predictor is selected, SZ stores four regression coefficients for each data block, which are compressed in a separate pipeline. The coefficient array occupies a very small portion of the memory (See Table 1). Thus, manipulating the coefficients array does not place as much pressure on the compression as directly operating compressed data.

3.2 Least-significant bit (LSB)

Due to its simplicity and ease of implementation, we chose LSB. The embedding process is divided into two steps.

Table 1: Coefficient Percentage

	CLOUDf48	**Wf48**	**Nyx**	**einspline**	**T**	**density**
coeffient(%)	0.2399	0.0084	0.1327	0.0918	0.0126	0.0926

First, convert the message into binary format. If there is enough space available, the LSB of each coefficient is replaced by a bit from the secret message in consecutive order. Since the bits of some coefficients are flipped during embedding, it is necessary to recalculate the decompressed value using the new quantization index to verify it still correctly respects the user-set *eb*. If the new decompressed value falls outside the predictable range, this coefficient is grouped into the unpredictable array.

This method has the advantage that it processes only the lowest bit of the value in a series of integers that occupy a very small fraction of the overall memory. Thus, minimizing the effects on the CR. Furthermore, LSB has a low computational effort and does not have a material impact on compression speed. The downside is that this method is easily cracked as it flips the fixed bit position. Furthermore, regression coefficients are sensitive to bit flips [9]. For highly compressible datasets, a substantial negative impact on the results may occur if the percentage of unpredictable coefficients increases after the verification step.

3.3 StegaZ

Algorithm 1: StegaZ

```
Input: Quantization index for regression
       coefficients $quant\_index$, secret
       message $M$, Hash function$H$
Output: Quantization index array for
        regression coefficients with inserted
        secret message
for $i$ in $quant\_index$ do
    $num \leftarrow$ 4 bit of $H$;
    $quant\_bit \leftarrow quant\_index >> num$;
    $secret\_message\_bit \leftarrow M >> i$;
    $result \leftarrow quant\_bit \oplus secret\_message\_bit$;
    if $(result! = quant\_bit)\&(quant\_bit = 1)$
      then
        Save $-quant\_index;$
    else if
      $(result == quant\_bit)\&(quant\_bit = 0)$
      then
        Save $-quant\_index;$
    else
        Save $quant\_index$;
    end
end
```

Our proposed method consists of two main components: position selection and bit hiding.We use a hash function to select random bits instead of inserting the message at a fixed location as in LSB. Our method has 2 steps: 1) The user enters a password (key) along with compression parameters. This key is then mixed with a random string, which serves as salt, and hashed using SHA-256. 2) Take the first four bits of the hashed string as the first target bit position for inserting the secret message. Next, shift one bit to the right and use the next four bits as the second target position, until there are sufficient places to embed the entire message. In the event that the data still needs to be steganographically written after one round of looping, return to the beginning of the hash string and start a new round. In this way, third parties without the key are unable to obtain the location of the inserted information.The addition of salt is equivalent to randomizing the hash value, which makes analysis algorithms like lookup tables and rainbow tables not be effective [14].

We utilize bitwise xor to conceal secret data bit by bit and use the symbol of the index as an identifier to distinguish embedded content. Algorithm 1 shows how to xor secret data and coefficient indices. In the case that the bit from the index is 1 and the result of the xor is still 1, keep the original index; otherwise, store the index as negative. Alternatively, if the bit from the index is 0 and the xor result is still 0, store the index as negative; otherwise, keep the original index unchanged. Only the sign of the index may change throughout this process. These four cases are identified without modifying the absolute value of the original coefficients. Our method is symmetric, and the extracting process only requires reversing the operation. Figure 1 shows the maximum size of inserted data for a 500MB dataset. We see compressing with blocks of size 6×6×6, a maximum of 289.35KB of secret data is hidden.

StegaZ compensates for the shortcomings of LSB while retaining its advantages. This approach still uses coefficients for steganography. However, it does not change the absolute values of the index, and the compression results are minimally affected. StegaZ does not significantly impact the overall compression speed, as most operations are bitwise. From a security perspective, even if a third party detects an anomaly, it is hard to extract the data from randomly flipped bits. A unique advantage of StegaZ is that only those who know the password can get the steganographic data, while uninformed users use the compressor and the decompressed data without any impact.

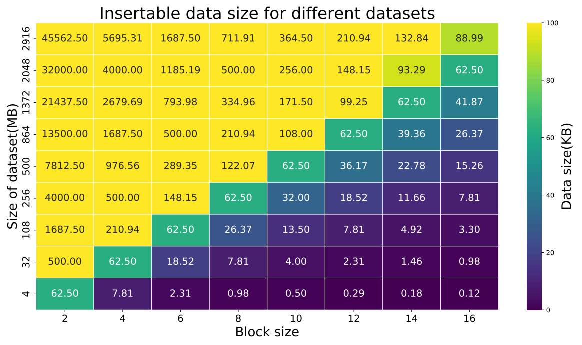

Figure 1: Maximum inserted data size.

4 Evaluation results

4.1 Experimental setup

Table 2: Dataset attributes

Dataset	Type	Dimensions	Size	Description
CLOUDf48	float	100×500×500	95.37MB	Cloud moisture mixing ratio
Wf48	float	100×500×500	95.37MB	Z wind speed
Nyx	float	512×512×512	527MB	Dark matter density
einspline	float	69×69×33120	602MB	ab initio electronic structure
T	float	98×1200×1200	61MB	Temperature
density	double	256×384×384	302MB	Rayleigh-Taylor simulation

We run experiments using real-world datasets from different applications to demonstrate the impact of our method on lossy compression. We select several datasets from SDRBench [15]. We choose *CLOUDf48* and *Wf48* from Hurricane Isabel simulation, *dark_matter_density* (to be called *Nyx*) from Nyx, *einspline* from QMCPACK, *temperature* (to be called *T*) from SCALE-LETKF, and *density* from Miranda. Table 2 shows detailed descriptions. Our experiments are conducted on Clemson University's Palmetto Cluster, where each node contains two 2.6 GHz Intel Xeon 2650 processors and 263GB of RAM. GCC-8.5.0 to compile our code, and we SZ3 in absolute error bound mode, enabling the Lorenzo and Linear Regression predictors for lossy compression. SZ is recognized as one of the best techniques available for compressing scientific datasets [16]. Every data point is the average of 500 rounds of hidden random data.

4.2 Compression ratio (CR)

CR is a standard to measure the effectiveness of a compressor and is defined as the ratio of the data size before compression to the data size after compression (higher is better). Our goal is to maintain the original CR as much as possible when combining steganography with SZ. CRs with minor variations are less likely to attract regulator attention. Table 3 presents the original CR for the datasets with different *eb*. From Table 3, *CLOUDf48* has the best compressibility, with a CR of about 38–900× greater than the hard-to-compress dataset *Nyx* with the same *eb*. This indicates that a significant portion of the data points in *CLOUDf48* falls within the same quantization interval, and the codewords are highly repeatable after Huffman encoding.

Incorporating steganography into SZ reduces CR. To compare two steganography methods, we use the original CRs as baselines and normalize other results. Figure 2 depicts the variance of CRs under different *eb*. As most datasets suffer little impact on CRs, modifying coefficients rather than the dataset itself is a proper option. In particular, *CLOUDf48* suffers from the greatest reduction in CR. When *eb* is set to 1e-3, the LSB and StegaZ methods retain 6.54% and 85.52% of the original CR, respectively. The reason is *CLOUDf48* highly compressible. Thus, the prediction coefficient compressibility has a large influence on the overall CR. Therefore, datasets whose CR is overly influenced by the coefficients are not amenable for our approach. However, our method is most applicable to datasets with very high CRs, such as *CLOUDf48*, when the *eb* is set to a more stringent range. For example, when the *eb* is 1e-6, the StegaZ approach retains 99.2% of the original CR. For datasets other than *CLOUDf48*, the StegaZ steganography approach can preserve more than 99.6% of the original CR, whereas LSB can only retain 97.1% on average. StegaZ always performs better than LSB since StegaZ eliminates the possibility of creating too many new and unique codewords.

Table 3: Baseline compression ratio

	Absolute Error Bound			
Dataset	**1e-6**	**1e-5**	**1e-4**	**1e-3**
CLOUDf48	38.745	90.916	353.336	2871.995
Wf48	2.286	3.056	4.531	8.293
Nyx	1.163	1.721	2.311	3.106
einspline	2.994	4.369	7.511	13.123
T	3.103	3.641	4.751	8.414
density	20.594	29.847	51.500	122.386

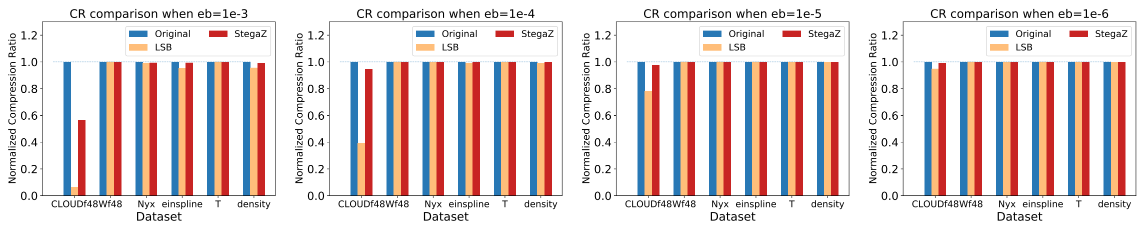

Figure 2: CR comparison with different eb

4.3 Peak signal-to-noise ratio (PSNR)

The output image is somewhat different from the original image after lossy compression. To ensure that the image quality of the decompressed dataset is maintained after steganography is engaged, we compare the PSNR values of the decompressed dataset images with and without steganographic methods. We observe that the LSB method results in a higher PSNR than the original, with an average increase of about 0.102% (See Figure 3). StegaZ, however, guarantees 100% of the original PSNR, indicating that this steganography scheme has no effect whatsoever on the image quality of the decompressed dataset. This is because SZ must verify and recalculate the correct bin number every time the least significant bit changes. This procedure causes some parameters that were previously predictable to become unpredictable. Unpredictable parameters are saved individually and precisely, which allows the decompressed data value to be more accurate and produce better images.

4.4 Compression Bandwidth (BW)

BW refers to the amount of data transferred per unit of time. One of the goals of HPC systems is to gain high BW so as to transfer more data in a given amount of time. In this section, *CLOUDf48*, *Nyx*, and *density* are selected as representative datasets for presenting the results. *CLOUDf48* exhibits a dataset with high compressibility and its CR varies greatly with the change of *eb*. *Nyx* represents a single-precision floating-point dataset that is hard to compress. And *density* is the only dataset with double-precision data points.

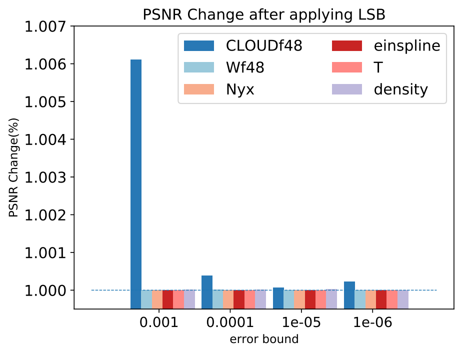

Figure 3: PSNR change after LSB

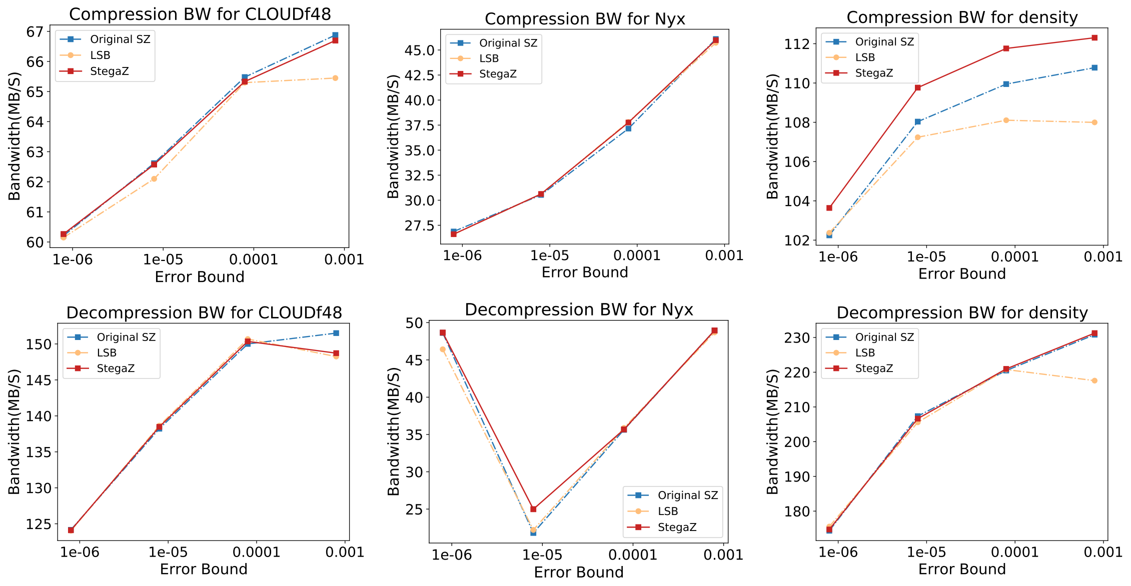

Figure 4: Bandwidth for different datasets

Overall, StegaZ can obtain very similar or even higher BW than the original SZ, with an overall fluctuation range of -0.2%–1.6%. This indicates the addition of the steganographic method does not greatly burden the data processing speed. This is due to both the fast bitwise operation and the fact that the StegaZ algorithm only produces a value with the opposite sign of the original quantization index instead of a completely new one. Therefore, it will not substantially affect the time required for subsequent compression sessions. The BW of a highly repetitive dataset suffers more, such as *CLOUDf48* when *eb* is 1e-3. At this point, the overall compression BW is dependent on the efficiency of coefficient processing. The way StegaZ handles the parameters still generates a small portion of new codewords, which slows down the overall compression process. However, datasets with high CRs are not an ideal steganographic mask and should be avoided as much as possible. LSB has a negative impact on BW. In particular, there is a significant BW pull-down problem when compressing *CLOUDf48* and *density*. The LSB algorithm changes the last bit of the quantization index indiscriminately, resulting in two possible scenarios: 1) The new index is still within the predictable range, and a new codeword is created after Huffman encoding. 2) The altered index is unpredictable and the coefficient value needs to be saved accurately. Both of these cases imply that it takes longer to compress the dataset. The most undesirable result appears when compressing *density*(*eb*=1e-3), the BW decreases by 2.5% compared to the original.

4.5 Security analysis

From a security perspective, the high randomness of the compressed dataset makes it difficult to directly distinguish whether it has been modified using steganography methods. The security of LSB depends entirely on the vigilance of intermediate entities. But StegaZ introduces an additional step of using the password to generate a hash value and perform operations on random bits. This step ensures only the recipient who knows the key can extract the secret data. We utilize a one-way hash algorithm SHA-256 as the hash function. Its security has been widely certified by various sources [14]. SHA-256 has a collision threshold value of 2^{128}. It takes millions of years to find a collision using brute-force attacks. Existing attacks have not yet been able to find a full collision for SHA-256. Besides, the addition of salt helps to randomize the hash value. Table 4 shows the entropy of the coefficients with and without StegaZ. The entropy of the coefficients increases prominently after StegaZ. A higher entropy indicates a greater level of randomness and a lower amount of carried information. This helps to better resist attacks such as lookup tables and rainbow tables [14].

Table 4: Coefficients entropy comparison

	CLOUDf48	**Wf48**	**Nyx**	**einspline**	**T**	**density**
Before	0.7189	11.3740	15.1833	10.7786	15.0017	0.7777
After	1.7161	12.0540	15.7015	11.3837	15.6522	1.7494

4.6 Select the appropriate dataset

Based on the above experimental results, selecting the appropriate dataset is crucial for achieving fast and inconspicuous steganography. For instance, in the case of *CLOUDf48*, the data itself exhibits high redundancy and can be highly compressed when the *eb* is loose. Thus, any variation in the parameters causes large fluctuations in the overall compression results. It is advisable to avoid selecting datasets with excessively high CRs for steganography. However, if no other datasets are available, the user can attempt to tighten the *eb*. For example, the compression results of *CLOUDf48* become closer to the original as the *eb* shrinks. Sacrificing some processing time by narrowing the *eb* is justifiable when considering the secrecy of the confidential data. Users can assess their own requirements and make trade-offs accordingly.

5 Conclusion

HPC systems are increasingly used in various fields for their exceptional data processing capability. Apart from big data analysis and simulations, HPC systems are also utilized for handling sensitive data. While cryptography is an excellent method of protection, the obvious encryption format is vulnerable to intruders. Sometimes it is preferable to hide sensitive confidential information. Compressed datasets are well-suited as carriers for hiding secret data in HPC systems. We propose a light-weight steganographic scheme based on HPC lossy compressor SZ which maintains the original compression results for most datasets. Only informed users with the password can extract the hidden content, while unaware users can use the compressor as usual.

Acknowledgments

This paper is supported by the U.S. NSF under Grants SHF-1910197/1943114.

References

[1] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, and K. Heitmann, "Hacc: Extreme scaling and performance across diverse architectures," in *Proceedings of International Conference on HPC, Networking, Storage and Analysis*, 2013, pp. 1–10.

[2] G. Marcus, Y. Ding, P. Emma, Z. Huang, J. Qiang, T. Raubenheimer, M. Venturini, and L. Wang, "High fidelity start-to-end numerical particle simulations and performance studies for lcls-ii," in *Proceedings, 37th International FEL Conference*, 2015.

[3] P. Lu, M. Lai, and J. Chang, "A survey of high-performance interconnection networks in high-performance computer systems," *Electronics*, vol. 11, no. 9, 2022.

[4] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W. Liao, and A. Choudhary, "Data compression for the exascale computing era-survey," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 76–88, 2014.

[5] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao *et al.*, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, 2022.

[6] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[7] G. Pellegrino, D. Balzarotti, S. Winter, and N. Suri, "In the compression hornet's nest: A security study of data compression in network services," in *24th {USENIX} Security Symposium*, 2015, pp. 801–816.

[8] D. Fulp, A. Poulos, R. Underwood, and J. C. Calhoun, "Arc: automated approach to resiliency for lossy compressed data via error correcting codes," in *Proceedings of 30th International Symposium on HPDC*, 2021, pp. 57–68.

[9] R. Shan and J. C. Calhoun, "Exploring data corruption inside sz," in *2022 IEEE International Conference on Big Data*. IEEE, 2022, pp. 3172–3178.

[10] R. Shan, S. Di, J. C. Calhoun, and F. Cappello, "Exploring light-weight cryptography for efficient and secure lossy data compression," in *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2022, pp. 23–34.

[11] R. Underwood, J. Bessac, D. Krasowska, J. C. Calhoun, S. Di, and F. Cappello, "Black-Box Statistical Prediction of Lossy Compression Ratios for Scientific Data," *ARXIV*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.08801

[12] S. Peisert, "Security in high-performance computing environments," *Communications of the ACM*, vol. 60, no. 9, pp. 72–80, 2017.

[13] Y. Zhang, X. Luo, J. Wang, Y. Guo, and F. Liu, "Image robust adaptive steganography adapted to lossy channels in open social networks," *Information Sciences*, vol. 564, pp. 306–326, 2021.

[14] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," in *Selected Areas in Cryptography: 10th Annual International Workshop*, 2004, pp. 175–193.

[15] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "Sdrbench:https://sdrbench.github.io."

[16] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong, "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.