Exploring Light-weight Cryptography for Efficient and Secure Lossy Data Compression

Ruiwen Shan*, Sheng Di†, Jon C. Calhoun*, Franck Cappello†‡
*Clemson University, Clemson, SC, USA
†Argonne National Laboratory, Lemont, IL, USA
‡ University of Illinois at Urbana-Champaign, Urbana, IL, USA
rshan@clemson.edu, sdi@anl.gov, jonccal@clemson.edu, cappello@mcs.anl.gov

***Abstract*—The enormous volume of data generated by large-scale instruments and simulations poses significant challenges in archiving, transferring, sharing and analyzing data for various scientific groups. Lossy reduction techniques are vital to reducing data size to acceptable levels. However, putting more information content per bit, increases the severity of loss if perturbed by malicious users or hardware failures. In the worst case, the entire dataset is compromised. Malevolent alteration or destruction of datasets containing crucial discoveries can completely invalidate research outcomes in scientific studies. Therefore, it is critical to integrate compression and encryption to handle data securely and efficiently. The current state-of-the-art combination technique Cmpr-Encr handles compression and encryption as two distinct processes. This reduces the compression ratio and bandwidth, especially for hard-to-compress datasets.**

In this paper, we propose two data protection strategies that work in conjunction with the lossy compressor SZ: Encr-Quant and Encr-Huffman, and carefully evaluate the overhead they introduce on compression bandwidth and ratio. Based on the results of testing with real-world scientific datasets, we find that the cost of Encr-Quant varies with the dataset's properties and requires cautious selection. Encr-Huffman is able to maintain more than 99% of the original compression ratio while saving 6.5% in compression time compared to SZ. Applying Cmpr-Encr leads to a reduction in compression bandwidth, whereas Encr-Huffman increases bandwidth by 3.1% over the SZ, on average.

***Index Terms*—Lossy Data Compression, Encryption, Data Security, High-performance Computing**

I. Introduction

Over the past decade, the utilization of large-scale computing has spread to more diverse domains. In order to conduct their science and make advances in their fields, scientists generate, share, and analyze large amounts of data. For example, approximately 170 TB of Community Earth System Model (CESM) data were produced for the Coupled Model Intercomparison Project 5 (CMIP5) [1]. The overall amount of output data for the CMIP6 is estimated to be between 20 and 40 PB [2]. Another typical example is the cosmological simulation code HACC, which can generate over 20 PB of down-sampled snapshots of particles in a single run [3]. Although large amounts of data can be generated, I/O bottlenecks result in a higher overhead to store computational results [4], [5]. Storage constraints are already limiting the length and size of some large-scale simulations [6], which presents difficulties for subsequent data analysis. Thus, the cost of storing/transmitting the output data from calculations can no longer be overlooked.

Lossless and lossy data compression are approaches for handling the large amount of floating-point data generated by large-scale scientific projects [7]–[10]. Error-bounded lossy compression (EBLC) is receiving considerable attention because of its ability to significantly reduce data size while maintaining user-defined accuracy constraints. While EBLC brings a lot of benefits, it neglects the problems that arise in terms of security. Prior work shows that lossy compression cannot withstand the consequences of bits being corrupted. Even a single bit-corruption can result in the complete failure of decompression [11]. Besides, information leakage may occur in compression techniques due to observable features such as the compression ratio [12]. Furthermore, sensitive data on HPC still faces a serious threat of being compromised, intercepted, or eavesdropped [13]. The loss of vital data or the availability of instruments can hinder or even halt the scientific process. To ensure the security of data, cryptographic techniques, such as encryption, are attractive due to their ability to provide strong privacy guarantees [14]–[16].

The question of how EBLC and data security can be properly and effectively integrated in practical situations is still an open question [17], [18]. The reasons this problem presents difficulties are as follows: 1) Lossy compressors have undergone rapid development in recent years. Consequently, most of the development is predominantly focused on improving compression efficiency (e.g., compression ratio) rather than ensuring the security of data. 2) Data compression and security are generally considered separately. Incorporating security methods into the compression process may adversely affect the compression ratio as well as the level of security provided by original encryption algorithms. It is important to investigate how to establish a link between security and lossy compression. 3) HPC systems are designed to process large datasets quickly and efficiently, and the emergence of security methods could influence the computational process, resulting in a reduction in overall performance. Therefore, the security solution should be light-weight, so as not to impose substantial overhead on the systems.

The cutting-edge technique in this domain, Cmpr-Encr (treats compression and encryption as independent processes), which introduces the problem of excessive overhead [18].

In particular, encryption dominates performance for hard-to-compress datasets and the sizable overhead imposed by encryption has a substantial impact on system performance. Our goal is to analyze and develop techniques for the unification of lossy data reduction and data security to enable reliable and dependable across scientific workflows. An efficient and secure compression-encryption method for all scientific datasets must be developed, and the biggest challenge is striking a balance between time/space overheads and data security.

Our contribution is summarized in two main points:

- We analyze the current-state-of-art method Cmpr-Encr and identify key weaknesses in the approach. To alleviate those weaknesses, we present two techniques for combining error-bounded lossy compression with data encryption in HPC systems: Encr-Quant and Encr-Huffman, which seek to limit the volume of data encrypted while maintaining a high-level of data security.
- We evaluate our proposed approaches on real-world scientific datasets to quantify the overhead caused by each on the compression ratio and bandwidth. The experimental results demonstrate that the light-weight Encr-Huffman method is able to retain over 99% of the original compression ratio while achieving higher bandwidth in the majority of circumstances. In the best case, Encr-Huffman achieves a bandwidth improvement of 4.8% and 7.8% over Cmpr-Encr and the original SZ, respectively, thus, enabling fast and secure data transmission and storage in HPC systems.

The rest of the paper is organized as follows. In Section II, we introduce background. We discuss the importance of integrating data compression and security and application scenarios in Section III. In Section IV, we propose our method and present experimental results in Section V. Finally, we discuss related work in Section VI and conclude in Section VII.

II. Background

A. Data compression

Compression reduces the burden produced by enormous volumes of data on transmission and storage by scientific datasets. Lossless compression is not an appropriate option in this situation since scientific datasets are primarily made up of floating-point values, and the highly random nature of the mantissa bits results in low compression ratios, typically around 2–4× [19], [20]. Some lossless compressors, e.g., FPZIP [9], are built exclusively for floating-point data; they can achieve a maximum compression ratio of roughly 10×, which is still far from satisfactory. On the other hand, error-bounded lossy compression (EBLC) allows for user-controlled data loss and has higher compression ratios on scientific datasets. Although lossy methods introduce errors into data, prior work has shown that error-bounding can preserve the workflow correctness very well [21]–[23]. Furthermore, in scientific simulation, some errors are inevitable due to inaccurate scientific sensors [24]. Current state-of-the-art lossy compressors, such as SZ [25] and ZFP [8], achieve compression ratios of 10–1000× while strictly respecting the error bound specified by the user.

SZ is an error-bounded lossy compression scheme for scientific data which is based on data prediction. Numerous studies [25], [26] have demonstrated that SZ behaves remarkably well for compressing scientific data.

As shown in the solid black chain in Figure 1, SZ's compression is composed of four steps: data prediction, linear-scale quantization, variable-length encoding and lossless compression [7], [27], [28]. In data prediction stage, SZ first splits the whole dataset into blocks of equal size and then uses a sampling approach to pick the best predictor among classical Lorenzo [29], mean-integrated Lorenzo and linear regression [27]. The classical Lorenzo predictor predicts the value of the current data point based on the value of the adjacent data point, but the uniformly skewed issue will occur since it uses the decompressed data to reconstruct the data point. Mean-integrated Lorenzo predictor is used to eliminate this deviation by approximating data points by a fixed value if the majority of data values are clustered to a small range with high density. A linear-regression model provides relatively accurate predictions for uniformly distributed datasets. SZ then continues to quantify the difference between the predicted and actual values using linear-scale quantification based on the user-identified error bound. In quantization, integers are used to represent the predictable data, whereas the unpredictable data is stored separately. Huffman encoding is subsequently performed in the third step to encode the predictable data. The final step is a pass of a lossless compressor such as GZIP [30] over reduced data to further boost the compression ratio.

B. Data security

The goal of data security is to ensure the security and compliance of data throughout its entire life cycle. A variety of techniques are available for securing data, including intrusion detection, role-based access control, multifactor authentication, firewalls, etc. [31]–[34]. However, due to the constraints in the HPC environment, these typical security measures may not operate as effectively in HPC as they do in traditional IT systems. For example, the size of the monitoring object in an HPC system is several orders of magnitude larger, which makes traditional intrusion detection approaches, such as inspecting every byte transmitted over the network impractical [35].

Security issues in HPC have received insufficient attention within the past few years as most HPC systems were mostly closed environments [36]. The advent of open-science expects researchers to transcend geographical boundaries and share their data among different organizations and systems [37]. However, open-science faces confidentiality challenges, as it may include information that is embargoed for regulatory purposes or that requires special handling before publication, such as HIPAA-regulated medical data. In HPC environments, security protocols must adhere to the PICA model, which emphasizes ensuring high performance while maintaining data confidentiality, integrity, and availability [38].

Cryptography is widely used for data protection. For example, the National Center for Computational Sciences

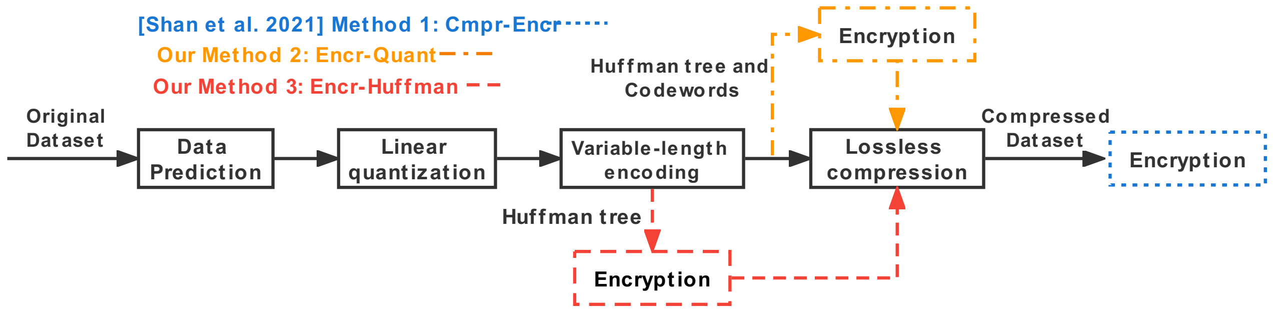


Fig. 1: Overview of the SZ compression process (Solid black chain) and modifications to enable encryption of the data.

(NCCS) at Oak Ridge National Laboratory (ORNL) released a secure framework, CITADEL, that allows researchers to access NCCS's supercomputers for open-science projects utilizing protected data [39]. This framework uses an encrypted parallel file system that enhances reliability and functionality.

Among the various algorithms available in the field of cryptography, the symmetric encryption algorithm Advanced encryption standard (AES) stands out for its efficiency and security [14]. According to the National Institute of Standards and Technology (NIST), AES is robust enough to guarantee security at extended key lengths and has a high probability of remaining secure well into the coming decades. Other widely known symmetric algorithms, such as Data Encryption Standard (DES) [40] and 3DES [41], suffer from a low level of confidentiality or slow processing speed. DES is extremely vulnerable due to its short 56-bit key. 3DES is a more secure variant of DES which applies DES encryption three times to each data block to improve data security. Because of the multiple calculations required for this process, the encryption speed of 3DES is not promising [42]. Meanwhile, asymmetric encryption methods such as Rivest–Shamir–Adleman (RSA) [43] are inappropriate for high-performance situations, as they are particularly slow in encrypting large data files.

III. Motivation

In this section, we discuss the importance of applying error-bounded lossy compression and cryptography in practice and analyze the use cases and the possible application scenarios.

A. Importance

Security in HPC is essential in order to maintain the legitimacy and confidentiality of data. Stakeholders in the HPC ecosystem can include academic computing, industry, national laboratories and even defense R&D. Moreover, some HPC systems are utilized for real-time operational challenges such as air traffic control and transportation, weather forecasting, etc. Under this premise, there is a legal obligation to provide enhanced security for sensitive information. Besides, malicious alteration of data can affect critical studies with policy implications, such as climate studies. Prior work investigates the resilience of lossy compressed data to bit-flip corruptions. Their results show that even a single bit-corruption is sufficient to violate the compressor's error bound or make the compressor fail during decompression [11], [44]. As for academia HPC systems, the loss of integrity of important research datasets, as well as the loss of confidentiality in terms of intellectual property, are serious issues. Furthermore, the theft of certain types of personal information, such as biometric data, can have catastrophic ramifications. There are numerous examples of science projects that were affected by cyberattacks. The cost of the damage ranges from lost working time to the usage of a substantial amount of financial resources to repair the system and restore scientific data. For instance, when COVID-19 research at UC San Francisco was hacked by ransomware, they were forced to pay approximately $1.14 million to obtain a tool to decrypt the sensitive data [45].

B. Use-cases

Applying compression methods significantly relieves the pressure of communication and storage. Although modern computers are capable of calculating billions to trillions of floating-point operations per second and generating datasets of terabytes and even petabytes in size, the bandwidth of the current internal interconnects is approximately 200 Gb/s [5]. This disparity places a significant strain on I/O devices. Data compression alleviates this by shrinking the size of data files. Some studies [4] observed that the overall I/O performance can increase by 1.86–4.12$\times$ over the original when using different types of compressors.

Meanwhile, cryptographic algorithms protect the transmitted content and prevent attacks from external sources. Jobs and resources stored or executed in HPC systems may include sensitive and highly profitable data, making them valuable targets for cybercriminals. Encryption protects these data in transit and during storage, preventing them from being easily interpreted even if they are intercepted or accessed by unauthorized users [13]. When a suitable encryption algorithm is selected and only used on the most sensitive portions of the system, the purpose of securing data can be achieved with little impact on system performance.

Furthermore, the redundancy reduced by data compression can potentially hinder certain cryptanalysis attacks such as frequency analysis and decrease their effectiveness [46]. The less ciphertext there is, the fewer clues it contains. Thus, making it harder for attackers to decipher [47].

C. Application Scenarios

The combination of compression and cryptography algorithms manifests in several scenarios:

HPC is the most typical scenario. Data from scientific projects, such as weather forecasts, inform national policy decisions and can have a direct impact on a country's destiny. Therefore, datasets residing on HPC systems must be of high integrity in order to assure reliable analysis. Applying compression methods at the proper phase could mitigate the I/O burden by shrinking the size of data files, while cryptography methods offer an effective solution to protect sensitive data.

In a cloud computing scenario, compression methods could help avoid the problem of storing massive data in the cloud. From the perspective of security, encryption schemes are needed to prevent the content of the data from being interpreted or by unauthorized access. Some cryptography methods, such as Homomorphic, enable users to leverage the power of cloud-based servers to process encrypted data without the original content being compromised during the computation, but with added costs [48].

The beauty of this combination also presents a possibility for its application in federated learning [49] [50]. Federated learning is the process of constructing efficient distributed machine learning systems by integrating models trained by various participants or computing nodes that do not have direct access to the training data. The ideas adhered to by this model reduce some costs brought by traditional centralized models, but also pose bandwidth and privacy issues in the transmission of gradients updates. In this circumstance, the combination of compression and encryption can be used to accelerate model transmission while also preventing unauthorized alternations.

IV. Combining Compression and Encryption

In this section, we describe the current state-of-the-art and our two proposed combination strategies for reducing inefficiency and insecurity in data transmission and storage. Cmpr-Encr is the most advanced technique in the field of integrated compression and encryption. It treats compression as a black-box, thus providing broad adaptability, but poses overhead problems [18]. In contrast, we present two white-box approaches that are tightly integrated within SZ and only focus on the most critical part of the datasets. Each scheme is designed from a unique perspective and incorporates certain modifications to SZ at different stages to satisfy users' expectations in terms of efficient and secure data transit and storage. Our methods significantly reduce the overhead of combining EBLC and encryption. We concentrate on SZ since it is an open-source compressor that provides the most robust and stable compression performance when compared to other existing lossy compressors [51], [52]. We note that our ideas can be translated into developing white-box integrations of compression and encryption for any compressor that leverages Huffman encoding (e.g., MGARD [53] and JPEG [54]).

A. Method 1: Cmpr-Encr

Cmpr-Encr considers data compression and encryption as two separate processes, where the cryptographic algorithm consumes the direct output of the compressor [18]. The blue dashed lines in Figure 1 indicate adjustments to SZ needed for Cmpr-Encr.

The contradiction between the ideas of compression and encryption algorithms is one justification for placing encryption at the end of the compression process. The ultimate goal of an encryption algorithm is to eliminate redundancies in the plaintext whereas EBLC needs these repetitions to identify relationships between each data point to compress them. In other words, the entropy of a dataset hits a theoretical maximum when optimally encrypted, making it extremely difficult to compress [55].

Even though some encryption schemes may require the injection of metadata (e.g., data from padding), which adversely affects compression performance, previous studies [18] show Cmpr-Encr encrypts data without significantly impacting the compression ratio. Thus, the encryption overhead is completely acceptable as long as fast and lightweight encryption methods such as AES are applied. Furthermore, the encryption bandwidth is data-type agnostic and only sensitive to data size.

The benefit of this methodology is that no changes need to be made to the compressor itself, and no in-depth understanding of how the compressor works is required. Developers simply regard it as a black-box with an encryption algorithm attached at the end. Moreover, it is a generic solution that is applicable to any lossless or lossy compressor, not just SZ. This strategy, however, does have certain limitations. For instance, there are occasions when it is necessary to encrypt a very large dataset because of a non-optimal compression ratio. At this point, the addition of encryption presents a large bandwidth overhead, which is inappropriate for efficiency-conscious systems such as HPC.

In general, Cmpr-Encr is suitable for cases that have high compression ratios and strict confidentiality requirements. Meanwhile, it is imperative to ensure that the system is adequately resourced.

B. Method 2. Encr-Quant

As SZ compresses the data, it attempts to predict data values based on spatial similarities. SZ stores the predictable and the unpredictable data separately and performs lossless compression on both together at the fourth stage. Consequently, we decided to encrypt the quantization array, which includes the Huffman tree, Huffman codewords and other metadata before lossless compression. The process of SZ after modification with Encr-Quant is depicted in Figure 1 by the orange dashed lines, and the pseudocode of Encr-Quant is presented in Algorithm 1 with orange and green text.

The Encr-Quant design is influenced by the following factors. First, in contrast to establishing security policies on traditional IT systems, performance is an additional core mission in data-intensive science. Thus, instead of assuming that all compressed bits are equally important and encrypting them all,

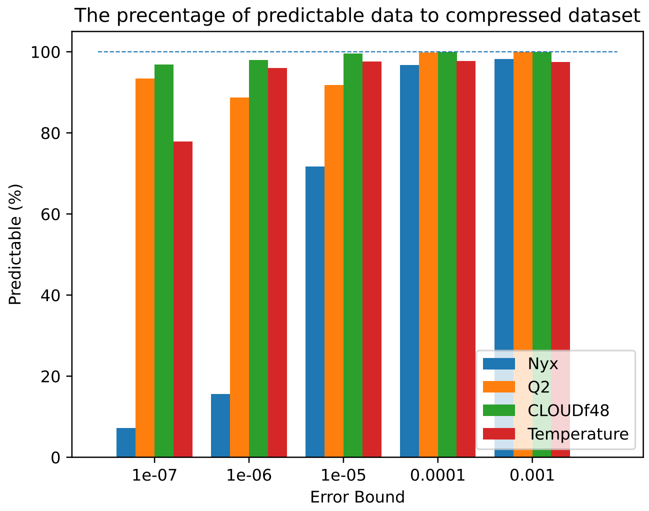
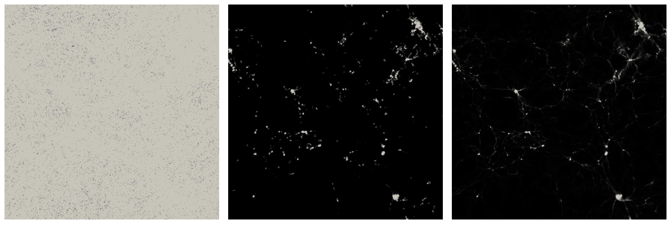


Fig. 2: Percentage of predictable data size compared to the size of compressed dataset.



(a) EB=1e-7. (b) EB=1e-3. (c) Original.

Fig. 3: Images of *dark_matter_density* from Nyx with different error bounds (EBs) where gray is the unpredictable data and black is the predictable data.

it is more effective to encrypt only a subset of the compressed bit stream. Figure 2 displays the quantization array size as a percentage of the overall compressed bit stream for four datasets we use in our evaluations (see Section V). In some cases, encrypting the quantization array is a relatively light approach to achieve the goal of securing data without incurring excessive costs. Thus, it should increase the efficiency of the encryption process for datasets with a relatively small percentage of predictable data (see Table IV). Second, even if the exact value of a data point is determined, it is irrelevant unless it can be precisely positioned into the raw dataset. Although unpredictable values in SZ are stored independently, it is notable that the location information of unpredictable data is stored in the quantization array as part of the codewords to ensure that the decompression results are accurate. In other words, the value of predictable and the location of unpredictable data are encrypted in this scheme. Figure 3 illustrates the binary images of the *dark_matter_density.bin* from dataset *Nyx*, with 1e-7 and 1e-3 error bounds along with the original data, respectively. Data points that are unpredictable are shown in gray, whereas data points that are predictable are shown in black. As shown in the Figure 3(b), relying simply on unpredictable data points does not help much in reconstructing the original image when predictable data dominates.

The Encr-Quant method is more suitable for HPC environments than Cmpr-Encr in certain cases. The reason is that this scheme is relatively lightweight; it decreases the amount of data that needs to be encrypted as well as the resources consumed during the process, thus reducing the overhead involved. However, the low overhead here is a relative concept. Due to the fact that each dataset has its own characteristics, when predictable data constitutes a particularly large fraction (e.g., more than 99%) of the entire data volume, the encryption effort of Encr-Quant is comparable to that of Cmpr-Encr. Besides, since we encrypt the quantization array before the lossless stage, the randomness introduced by encryption affects the performance of this subsequent process by some degree. Thus, this method may have a relatively large impact on the compression ratio. Although Encr-Quant is designed SZ, it is compatible with other compressors that use quantization and Huffman coding (e.g., JPEG [54]).

C. *Method 3: Encr-Huffman*

In the Encr-Huffman method, only the small-size Huffman tree is encrypted after the quantization array is compressed by Huffman encoding. Red dashed lines in Figure 1 represent the process after applying Encr-Huffman. According to our previous assumption, the efficiency of Encr-Quant could be comparable to that of Cmpr-Encr at a higher compression ratio. Thus, instead of processing the entire quantization array, we decide to extract the Huffman tree and encrypt it. The pseudocode of Encr-Huffman is presented in Algorithm 1 with red and green text.

Two key factors are considered in this design. On one hand, in HPC security, efforts should be concentrated on those elements that are most crucial, such as sensitive data and the limited hardware and stack [35]. The composition of the Huffman tree is the most important step in the entire compression process. On the other hand, according to previous studies, guessing the plaintext of encoded data without Huffman trees is NP-hard [56], [57]. In our case, this means that in the absence of the Huffman tree, it is almost impossible to restore the quantization array and reconstruct the dataset. Besides, the effective key space of AES-128 is large enough to withstand brute force attacks(see Section V-G).

Although the size of the Huffman tree varies when compressing different datasets, it remains a relatively small and stable proportion of the compressed bit stream. We compress and compare the percentage of the Huffman tree within several datasets. Results in Figure 4 demonstrate our assumptions: the Huffman tree comprises no more than 4.5% of the quantization array during the compression process.

The Encr-Huffman scheme is not generic, but can be applied to other compressors that leverage Huffman encoding, such as JPEG [54], MGARD [53], and MP3. However, as it only encrypts a small fraction of the compressed data, it offers high bandwidth while guarantying the CR. Section V shows this strategy has a lower overhead and superior performance in the HPC environment than the previous two approaches.

Algorithm 1: Combining compression and encryption. Black indicates the flow of original SZ, orange represents Encr-Quant, red stands for Encr-Huffman, and green shows the process of encryption

```
Input: Dataset $D$, user-specified error-bound $eb$, Key $k$
Output: Compressed-Encrypted Dataset $D'$
while Next data point in dataset exists do
    Calculate the predicted value $v_1$ based on the previous
      predicted value $v_0$ and the best-fit predictor;
    if data point is predictable then
        Quantified the difference between predicted value and its
          original value;
    else
        Data point is unpredictable;
    end
end
Construct the Huffman tree $H$ according to the quantization array;
Encode quantization array using $H$;
Divide $H$ into n blocks $B_n$;
Get predictable array $PA$ includes $H$ and $codewords$;
Divide $H$ and $codewords$ into n blocks $B_n$;
Generate random Initial Vector $IV$;
while $i<n$ do
    if $i = 0$ then
        $M_i$=IV$\bigoplus B_i$;
    else
        $M_i$= $Cipher_{i-1} \bigoplus B_i$;
    end
    $Cipher_i = Encrypt(k, M_i)$;
    i++;
end
Compress the unpredictable array using IEEE 754 binary
  representation analysis;
Compress regression coefficients;
Apply Zlib compression;
```

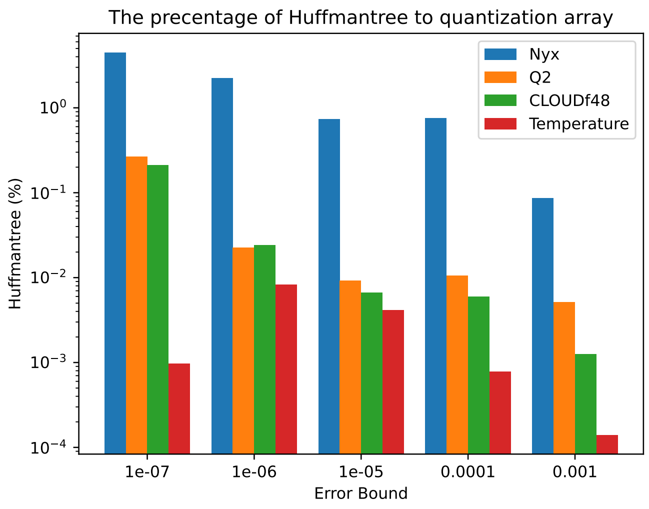


Fig. 4: The percentage of compressed array size of Huffman tree compared to quantization array.

V. Evaluation Results

A. Experimental Setup

To evaluate the efficiency and applicability of each of the proposed strategies, we run various experiments on Clemson University's Palmetto, which has two 2.4 GHz Intel Xeon Gold 6148 processors and 376 GB of RAM per node. We test with data sets from SDRBench [58]: *CLOUDf48* and *Wf48* from the Hurricane Isabel simulation, *dark_matter_density* from Nyx, as well as four fields *Q2*, *Height*, *QI*, and *T* from atmospheric modeling code SCALE-LetKF [59] for our experiments. We use *Nyx* to denote *dark_matter_density* in the rest of our paper. Table I details the attributes of each dataset.

TABLE I: Attributes of the datasets used in experiments.

dataset	Dimensions	Size	Description
CLOUDf48	100×500×500	95.37MB	Cloud moisture mixing ratio
Wf48	100×500×500	95.37MB	Hurricane wind speed
Nyx	512×512×512	527MB	Dark matter density
Q2	11×1200×1200	61MB	2m Specific humidity
Height	98×1200×1200	1.1GB	Height above ground
QI	11×98×1200×1200	5.8GB	Cloud Ice mixing ratio
T	11×98×1200×1200	5.8GB	Temperature

We compile our code using GCC-8.3.1 and run all of our experiments with a single thread. We employ the error-bounded lossy compressor SZ [26] with absolute error bound mode because several studies have demonstrated the benefits of SZ in terms of its ability to compress scientific datasets among current up-to-date lossy compressors [51]. We choose SZ-1.4 because it is easier to integrate encryption into than later versions. Newer versions of SZ leverage all the features of SZ-1.4, but add newer prediction algorithms. Therefore, our approach is also applicable to subsequent versions. On the basis of SZ-1.4, we implement three strategies proposed in Section IV. For encryption, we adopt the lightweight cryptography algorithm AES-128 Cipher Block Chaining (CBC) mode, as it maintains adequate security levels with faster encryption [60], [61].

B. Evaluation metrics

We evaluate the performance of different combined compression and encryption methods based on several criteria.

Compression Ratio (CR) quantifies the degree of compression. CR is defined as the ratio of the original total data size to the compressed data size. A higher CR indicates that data has been compressed to a greater extent.

$$CompressionRatio(CR) = \frac{Size_{original}}{Size_{compressed}} \tag{1}$$

Bandwidth refers to the total amount of data that is compressed/decompressed per unit of time. A larger bandwidth is attained when a dataset takes less time to compress/decompress.

$$Bandwidth(BW) = \frac{Size_{original}}{t_{compression/decompression}} \tag{2}$$

Time Overhead represents the percentage of the total time required for compression after adding the encryption process compared with the time required to compress the file using the original compressor. When this value is less than 100%, it signifies that the new approach takes less time to process the dataset than the original method, and vice versa. A lower time overhead indicates that encryption places less strain on the critical path in the reduction pipeline.

$$Overhead(\%) = \frac{t_{new}}{t_{original}} \times 100\% \quad (3)$$

Randomness analysis. Encryption algorithms should generate random output from any input data to prevent an attacker from identifying patterns between plaintext and ciphertext. The National Institute of Standards and Technology (NIST) has a test suite, NIST SP800-22 [62], for randomness analysis. This test suite includes 15 statistical tests, each of which is evaluated using a p-value between 0 and 1. When the p-value exceeds 0.01, the test results meet the criteria.

C. Compression Ratio

Table II summarizes the CR of each dataset under different error bounds while using the original compressor. It is evident that there is a significant difference in CR amongst datasets. These variations are attributed to the characteristics of each dataset. *CLOUDf48* and *QI* have the highest CRs of all datasets, indicating that these two datasets are easier to compress. In contrast, *Nyx* is a hard-to-compress dataset with CRs in the range of 1–3×. One of the major reasons for this discrepancy is the fact that SZ relies on data correlations to make predictions. In general, datasets with highly correlated data points are more compressible. When a high proportion of predictable data is present in the dataset (see Figure 2), such as *CLOUDf48* and *Q2* when the error bound is set to 1e-3, the CR is relatively higher. However, data predictability is not the only criterion used to assess compressibility. For example, *T* and *Nyx* have more than 96% predictability when the error bound is 1e-3, but the CR is still in the single digits. One of the reasons is that the size of the Huffman tree is still relatively large for *Nyx* (see Figure 4). Besides, the predictable data is unique, and a vast number of unique codewords are generated when encoded with a Huffman tree, leading to a lower CR.

TABLE II: Baseline compression ratio with no encryption.

	Absolute Error Bound				
Dataset	**1e-7**	**1e-6**	**1e-5**	**1e-4**	**1e-3**
CLOUDf48	17.959	27.216	51.731	311.799	2380.782
Nyx	1.145	1.184	1.704	2.320	3.082
Q2	4.288	7.387	13.352	24.470	89.384
Height	2.802	4.343	5.718	7.847	12.687
QI	67.931	182.291	446.896	1709.021	3654.457
T	3.076	3.301	3.414	5.197	9.997

The addition of data encryption has an adverse effect on CRs by introducing extra data. We use the original CR as a baseline and normalization point to compare the three compression and encryption methods. Figure 5 presents the impact on CR. Both Cmpr-Encr and Encr-Huffman are able to retain more than 99% of the original CR in each experiment. In most cases, the CR of Cmpr-Encr is closer to the original than Encr-Huffman. The largest discrepancy is roughly 0.26% in *Nyx* where error bound is 1e-7. *Nyx* is dominated by unpredictable data (see Figure 2), with the Huffman tree accounting for approximately 4.4% of its quantization array (see Figure 4). Encrypting the relatively large-size Huffman tree reduces the CR by lowering the quality of the subsequent

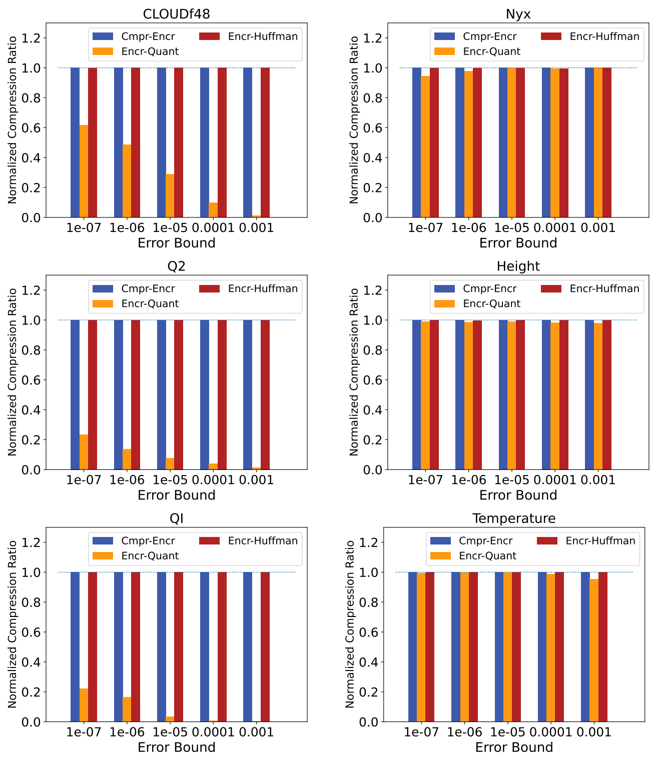


Fig. 5: Normalized compression ratio for different datasets.

lossless compression. Conversely, Encr-Quant has a significant impact on the CR in some datasets. For example, the ratio for *QI* and *Q2* is only 5%∼20% of the original. In general, the Encr-Quant approach has a greater impact on easy-to-compress datasets (*CLOUDf48*, *Q2*, *QI*) than on those which are hard-to-compress (*Nyx*, *Height*, *Temperature*). These easy-to-compress datasets contain a higher proportion of predictable data that is encoded with high repeatability, hence obtaining greater CRs when lossless compression is subsequently applied. The encryption of the quantization array prior to lossless compression increases the entropy and randomness of the array and limits its ability to benefit from the lossless process, resulting in substantial reductions in the CR. The worst case occurs when the error bound for *Q2* is set to 1e-2. At this point, 99.99% of the data is predictable and the Encr-Quant scheme is roughly 0.01% of the original CR. By contrast, the Encr-Huffman method introduces randomness into the Huffman tree storage, which has little effect on the compression capabilities of the lossless compression step. Thus, we conclude that Cmpr-Encr and Encr-Huffman algorithms are preferable for applications with data security requirements and strict size restrictions.

D. Compression Bandwidth

To make the experimental results more generalizable, we use three datasets from Table I to evaluate the throughput of our methods. We select *Temperature* from SCALE-LetKF (lowest CR), *CLOUDf48* from Hurricane (high CR) and *dark_matter_density* from *Nyx* (low CR). All data points in Figure 6 are an average of five runs.

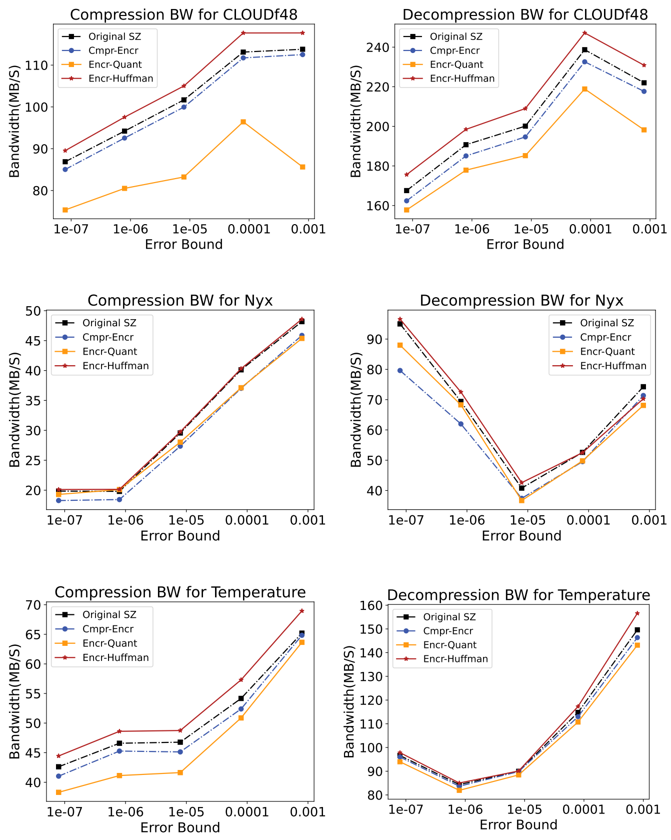


Fig. 6: Bandwidth for different datasets.

Generally, the bandwidth increases with the easing of the error bound. A special case is when the error bound is 1e-3 in *CLOUDf48* for Encr-Quant, the bandwidth actually drops. This is due to the encryption process generating data that requires more time to compress losslessly. Thus, decreasing the bandwidth. Compression bandwidth is usually lower than decompression. In the case of *CLOUDf48*, the highest compression bandwidth is 117 MB/s while the lowest decompression bandwidth is 157 MB/s per CPU core. This is due to the mathematical computations required in the compression process, such as prediction and quantification, that are not present in decompression.

The three methods perform similarly on the hard-to-compress dataset *Nyx*, but differ significantly on the compressible dataset. It is noteworthy that Encr-Huffman (red trend) remains dominant over the other methods. The largest performance gap among Encr-Huffman, Cmpr-Encr and Original SZ occurs while processing *Temperature*. In this case, the bandwidth of Encr-Huffman is 7.8% and 4.8% higher than Cmpr-Encr and SZ, respectively, on average. Besides, Encr-Huffman achieves a bandwidth that is 25% higher on average than Encr-Quant when working with *CLOUDf48*. Encryption is a time-consuming and resource-intensive operation, the Encr-Huffman approach encrypts only small-sized Huffman trees (see Figure 4) rather than the entire dataset. Consequently, a high level of security is provided while saving considerable computation time and energy (see Section V-G).

TABLE III: Time overhead for Cmpr-Encr when compressing (%)

	Absolute Error Bound				
Dataset	**1e-7**	**1e-6**	**1e-5**	**1e-4**	**1e-3**
CLOUDf48	102.172	101.789	101.729	101.244	101.138
Nyx	105.870	105.252	105.026	105.332	103.602
Height	103.849	103.877	103.465	103.020	102.234
Q2	105.550	102.480	101.963	101.507	101.059
QI	100.885	100.124	100.055	100.022	100.016
T	103.869	102.920	103.641	103.389	100.555

When compared to Encr-Quant, Cmpr-Encr exhibits impressive bandwidth for *CLOUDf48* and *Temperature*; however, since it must impose time overhead from encryption on the original compressor, it cannot attain more bandwidth than the original SZ. Besides, Cmpr-Encr never achieves a higher bandwidth than Encr-Huffman since the size of Huffman tree is always smaller than the compressed bit stream with the same error bound. The overhead of Cmpr-Encr varies with the size of the compressed dataset. As the data size decreases, the cost of encryption also reduces. The advantage of Encr-Huffman is apparent in this evaluation. However, when dealing with datasets like *Nyx*, the bandwidth of the three methods is similar. In this case, the data in *Nyx* is evenly distributed and the majority of data points are unpredictable. At this point, the encryption process takes a relatively short time and does not have a significant impact on the compression time. The data size to be encrypted by Encr-Quant may be larger than that of Cmpr-Encr for a dataset containing primarily predictable data. For example, CLOUDf48 has 96.8% predictable data when the error bound is 1e-7, and the size of codewords to be encrypted by Encr-Quant is around 8.8 MB, whereas Cmpr-Encr only needs to encrypt 5.3 MB compressed bit streams. Besides, encrypting large-size codewords introduces a lot of randomnesses, which significantly increases the time required for lossless compression (see Figure 7).

E. Time Overhead

Tables III–V summarize the results of testing the time overhead of three combination algorithms with *CLOUDf48*, *Nyx*, *Q2*, *Height*, *QI*, *T* under different error bounds. The values in each table represent the average of five runs.

As shown in Table IV, the overhead of Cmpr-Encr is greater than 100% in all cases, meaning that executing the Cmpr-Encr algorithm takes longer than running the original algorithm. Since this approach makes no changes to the compression algorithm, all overhead is derived from the subsequent encryption process. The overhead grows as the error bounds get tighter because larger data takes longer to encrypt. The overhead for most of the datasets ranges between 0.1%–3%. *QI* has the lowest overhead of 0.016% with an error bound of 1e-3, while *Nyx* has the highest overhead of approximately 6% when the error bound is set to 1e-7.

The time overhead of Encr-Quant is not proportional to the change in the error bound in most cases. The overhead of this scheme is affected by two factors: the size of the quantization array and the time spend on lossless compression. The huge amount of codewords places considerable loads on

TABLE IV: Time overhead for Encr-Quant when compressing (%)

Dataset	Absolute Error Bound 1e-7	1e-6	1e-5	1e-4	1e-3
CLOUDf48	115.301	117.019	118.154	119.303	122.882
Nyx	104.787	104.376	104.950	103.957	102.853
Height	103.670	109.089	107.983	104.066	108.504
Q2	100.601	100.345	100.138	104.093	100.217
QI	127.163	119.645	131.480	133.524	133.299
T	111.272	113.277	114.337	106.470	102.432

TABLE V: Time overhead for Encr-Huffman when compressing (%)

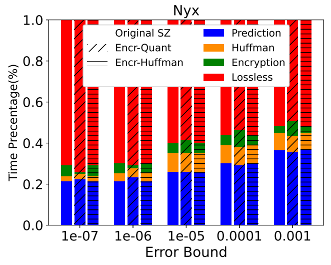
Dataset	Absolute Error Bound 1e-7	1e-6	1e-5	1e-4	1e-3
CLOUDf48	97.032	96.576	96.828	96.114	96.679
Nyx	98.691	98.328	99.364	99.524	99.157
Height	95.739	95.953	94.993	94.469	93.577
Q2	95.204	91.512	89.598	91.295	90.175
QI	98.541	97.188	97.193	97.818	97.623
T	99.084	99.344	97.949	97.747	97.567

encryption when the proportion of predictable data in the dataset is relatively large and the data is uniquely distributed. Meanwhile, the increase in entropy of the encoding array after encryption reduces the dataset's compressibility and slows down lossless compression. This unpredictability is determined by the dataset's properties. It is clear from Table IV that the overhead of this technique is not promising in some circumstances, it sometimes increases processing time by more than 30%. For datasets like *Nyx* where unpredictable data dominates, Encr-Quant requires a smaller encryption size than Cmpr-Encr, thus lowering the overhead by approximately 1%. However, for compressible datasets as *QI* and *CLOUDf48*, fewer unpredictable data cause a significant temporal overhead, which in the worst case introduces 33% overhead more than that introduced by Cmpr-Encr and 46% more than the time overhead of the Encr-Huffman.

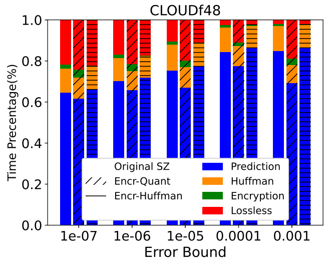
With respect to time overhead, Encr-Huffman performs well and relatively stable in comparison to the first two schemes. Encr-Huffman takes less time to execute in most circumstances than utilizing SZ only, as it only encrypts the small-size Huffman tree. In the best situation, Encr-Huffman saves 6.5% in time compared to the original SZ. The randomness introduced by the encryption algorithm in this case affects a tiny portion of the data. This small modification to the buffer results in a transformation that enables the lossless compressor to compress faster, but not give as good of a compression ratio. The entropy value of the dataset after applying Encr-Quant is extremely high, approaching the theoretical maximum value of 8 [55], making lossless compression time-consuming. In comparison to the original SZ, Encr-Huffman reduces entropy by 0.01 on average. Thus, not only preserving more than 99% of the original CR, but also speeding up the lossless procedure. Therefore, when considering all the factors, the Encr-Huffman scheme is the best choice among the three.

F. Randomness Analysis

Randomness is critical to data security because it allows intruders to see all the coded data but cannot extract any



(a) Nyx.



(b) Hurricane.

Fig. 7: Time breakdown for different datasets.

information or patterns from it. The NIST SP800-22 [62] test suite is an excellent tool to investigate different aspects of randomness in the crypto bit streams. We conduct NIST SP800-22 tests on all approaches, but only discuss results from *Nyx* and *Q2* for space reasons. The test suite splits the compressed data file is separated into several bit streams, each of which is evaluated independently to obtain a complete picture of the data distribution. The *Pass Rate* column in Table VI, show the number of bit streams passing the test.

The second column in Table VI lists results when we set the error-bound to 1e-7 and apply the Encr-Quant method to the dataset *Nyx*. *Nyx* in this setting is an extreme example, with only 7.2% of predictable data that is encrypted (see Figure 2). Unlike Cmpr-Encr, which easily passes all the randomness tests from the test suite, Encr-Quant failed the vast majority of the tests, indicating the resulting data has non-randomness.

We also investigate the randomness of Q2 when the error bound is 1e-6, where the data is 85% predictable. The third column in Table VI shows that the Encr-Quant scheme successfully passes all NIST tests in this case, thus proving its randomness. Compared with the results from *Nyx*, we conclude that when there is a high amount of predictable data, the Encr-Quant technique can encrypt most of the data and successfully produce completely random compressed bit streams. Encr-Huffman, predictably, fails all randomness tests since it only encrypts the small-size Huffman tree and has no effect on the randomness of the rest of the compressed data. The studies above reveal that the proportion of the encryption scale determines the randomness of output bit streams. The randomness of the compressed data improves as the proportion of encrypted data grows. As a result, Cmpr-Encr produces completely random output, whereas the randomness of Encr-Quant is dictated by the fraction of predictable data, and Encr-Huffman does not provide sufficiently random results because it only encrypts a very small part of the data.

G. Security Analysis

From a randomness standpoint, Encr-Huffman and Encr-Quant may appear insufficient to guarantee appropriate security, but randomness is only one of the factors that determine data security. These two techniques still provide an adequate security level because the attacker is unable to rebuild the Huffman tree and decode the compressed data without the

TABLE VI: NIST test for *Nyx* and *Q2* with Encr-Quant

Statistical test	Pass Rate for Nyx	Pass Rate for Q2
Frequency	58.33%	100%
Block frequency	50.00%	100%
Runs	58.33%	100%
Long runs of one's	58.33%	100%
Binary Matrix Rank	91.66%	100%
Spectral DFT	91.66%	100%
No overlapping templates	50.00%	100%
Overlapping templates	58.33%	100%
Universal	58.33%	100%
Linear complexity	100%	100%
Serial	66.67%	100%
Approximate entropy	58.33%	100%
Cumulative sums	58.33%	100%
Random excursions	100%	100%
Random excursions variant	100%	100%

key. Several studies have demonstrated that it is an NP-Hard problem to completely decompress the compressed data without the Huffman tree [56], [57], and the Encr-Huffman method complicates decoding by injecting randomness into the Huffman tree. Moreover, the encryption algorithm we use, AES-128, has an effective key space of 2^{64} [63], which is large enough to resist brute force attacks. Even with a supercomputer capable of testing 22×10^{19} encryptions/sec, brute-forcing the encrypted data still takes roughly 3.7×10^{10} years. Among the currently known attack strategies against AES, the biclique attack is computationally faster than the brute-force attack, which can recover the key for AES-128 with a computational complexity of $2^{126.1}$ and is not feasible [64].

VI. Related work

HPC security is currently gaining popularity and inspiring a growing quantity of research in this field. Science DM Z (Demilitarized Zone) is an effective solution for optimizing security strategy for high-performance scientific applications [38]. With this security model, a site's scientific computing is isolated in its own network enclave while all data is transferred through a single network that can be monitored with intrusion detection systems (IDS) controlled with access control lists (ACLs). However, resources in the Science DMZ are specifically designed to interact with external systems and are segregated from internal systems, making them ineffective when confronting insider threats. Other than this isolation-based security strategy, some studies [65] have explored the potential for hardware-based Trusted Execution Environment (TSS) to achieve safe scientific computing. Their findings reveal that AMD's Secure Encrypted Virtualization (SEV) can protect HPC systems against threats with lower performance overhead if configured correctly. However, SEV still slows down the performance by 1x-4x compared to native execution for graph applications.

A verity of additional fields are investigating combining compression with encryption. Some researchers [60] have investigated the possibility of combining encryption and compression with edge computing. After data is collected, it is first encrypted at the sensor node with the AES algorithm prior to transmission, then decrypted and compressed at the edge-assisted gateway before finally sent to the LoRa-based gateway. This design has proven to be effective at saving bandwidth while providing more robust data validation with only a minor increase in latency. A Fast and Secure (FS) data transmission module is proposed for vertical federated learning structure in paper [50]. The Host first extracts and compresses half of the feature value of the data in the host side, then use a homomorphic-based encryption algorithm to transmit the data to the Guest. This module can boost transmission efficiency by 50% and significantly reduce overall system delay. The aforementioned studies demonstrate the numerous practical applications of lossless compression and encryption. However, these methods are not suitable for scientific data due to its massive size and need for lossy algorithms with fast reduction bandwidths and higher compression ratios to handle data generated from instruments and applications.

There is also some exploratory work on combining data compression and encryption on HPC systems, such as integrating these two techniques into persistent key-value stores (KVS) for HPC [66]. The author proposes only compressing and encrypting the value in the key-value pair and sensitive data in the Sorted Strings Table (SSTable) by exploiting the memory hierarchy in HPC systems. Their findings demonstrate that this strategy can achieve the goal of decreasing data size and securely sharing at a cost that is practically undetectable. However, this scheme is limited to lossless compression and encryption of key-value pairs and is not suitable for scientific datasets with large amounts of floating-point data.

VII. Conclusion

Lossy compression techniques significantly alleviate the problem of managing, transferring, and storing large volumes of data. However, the data is still vulnerable and valuable to malicious agents. Encryption is a method to preserve the confidentiality and integrity of data throughout its lifecycle, but is currently not well integrated into lossy compressors. This paper presents two techniques for integrating data compression and encryption in HPC systems. Our experiments indicate that the Encr-Quant method is more sensitive to dataset features and has a restricted range of applicability. When there is a considerable amount of predictable data, this strategy reduces the effectiveness of the subsequent lossless process, which leads to a significant fall in compression ratio and bandwidth. The Encr-Huffman approach has a wider range of applications on which it achieves higher compression bandwidth than the original SZ while still retaining a great compression ratio even for hard-to-compress datasets. Overall, the Encr-Huffman scheme is light-weight and has a great practical utility. This approach compresses datasets in high quality while maintaining data security, enabling fast and confidential data transmission and storage in HPC environments.

Acknowledgments

This research was also supported by the U.S. National Science Foundation under Grants SHF-1910197, SHF-1943114, OAC-2003709, and OAC-2104023. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357.

References

[1] R. J. Small, J. Bacmeister, D. Bailey, A. Baker, S. Bishop, F. Bryan, J. Caron, J. Dennis, P. Gent, H.-m. Hsu *et al.*, "A new synoptic scale resolving global climate simulation using the community earth system model," *Journal of Advances in Modeling Earth Systems*, vol. 6, no. 4, pp. 1065–1094, 2014.

[2] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor, "Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization," *Geoscientific Model Development*, vol. 9, no. 5, pp. 1937–1958, 2016.

[3] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, V. Vishwanath, T. Peterka, J. Insley *et al.*, "HACC: Extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

[4] 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.

[5] J. Zhong, L. Zhonghua, L. Huiyuan, C. Xuebin, and S. Jiachang, "Origin of high performance computing——current status and developments of scientific computing applications," *Bulletin of Chinese Academy of Sciences (Chinese Version)*, vol. 34, no. 6, pp. 625–639, 2019.

[6] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, "A methodology for evaluating the impact of data compression on climate simulation data," in *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, 2014, pp. 203–214.

[7] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with sz," in *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE, 2016, pp. 730–739.

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

[9] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[10] M. Burtscher and P. Ratanaworabhan, "High throughput compression of double-precision floating-point data," in *2007 Data Compression Conference (DCC'07)*. IEEE, 2007, pp. 293–302.

[11] D. Fulp, A. Poulos, R. Underwood, and J. C. Calhoun, "Arc: An automated approach to resiliency for lossy compressed data via error correcting codes," in *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, 2021, pp. 57–68.

[12] J. Kelsey, "Compression and information leakage of plaintext," in *International Workshop on Fast Software Encryption*. Springer, 2002, pp. 263–276.

[13] S. Peisert, V. Welch, A. Adams, R. Bevier, M. Dopheide, R. LeDuc, P. Meunier, S. Schwab, and K. Stocks, "Open science cyber risk profile (oscrp)," 2017.

[14] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[15] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[16] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[17] N. Weinberger and N. Merhav, "A large deviations approach to secure lossy compression," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 2533–2559, 2017.

[18] R. Shan, S. Di, J. C. Calhoun, and F. Cappello, "Towards combining error-bounded lossy compression and cryptography for scientific data," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–7.

[19] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W. keng Liao, and A. Choudhary, "Data compression for the exascale computing era - survey," *Supercomputing frontiers and innovations*, vol. 1, no. 2, 2014. [Online]. Available: http://superfri.org/superfri/article/view/13

[20] S. Mittal and J. S. Vetter, "A survey of architectural approaches for data compression in cache and main memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1524–1536, 2015.

[21] Z. Miao, J. C. Calhoun, and R. Ge, "Relaxed replication for energy efficient and resilient gpu computing," in *2021 IEEE/ACM 11th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS)*, 2021, pp. 41–50.

[22] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, "Toward a multi-method approach: Lossy data compression for climate simulation data," in *High Performance Computing*, ser. Lecture Notes in Computer Science, J. M. Kunkel, R. Yokota, M. Taufer, and J. Shalf, Eds. Springer International Publishing, 2017, vol. 10524, pp. 30–42.

[23] J. Calhoun, F. Cappello, L. N. Olson, M. Snir, and W. D. Gropp, "Exploring the feasibility of lossy compression for pde simulations," *The International Journal of High Performance Computing Applications*, vol. 33, no. 2, pp. 397–410, 2019. [Online]. Available: https://doi.org/10.1177/1094342018762036

[24] G. Han, X. Wu, S. Zhang, Z. Liu, and W. Li, "Error covariance estimation for coupled data assimilation using a lorenz atmosphere and a simple pycnocline ocean model," *Journal of Climate*, vol. 26, no. 24, pp. 10 218–10 231, 2013.

[25] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello, "Significantly improving lossy compression quality based on an optimized hybrid prediction model," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–26.

[26] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 1129–1139.

[27] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, "Error-controlled lossy compression optimized for high compression ratios of scientific datasets," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 438–447.

[28] X. Liang, H. Guo, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, and T. Peterka, "Toward feature-preserving 2d and 3d vector field compression." in *PacificVis*, 2020, pp. 81–90.

[29] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.

[30] [Online]. Available: https://www.gzip.org.

[31] S. Beg, U. Naru, M. Ashraf, and S. Mohsin, "Feasibility of intrusion detection system with high performance computing: A survey," *Int. J. Advances in Computer Science*, vol. 1, pp. 26–35, 2010.

[32] D. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-based access control*. Artech house, 2003.

[33] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: A survey," *Cryptography*, vol. 2, no. 1, p. 1, 2018.

[34] A. Prout, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, M. Houle, M. Jones, P. Michaleas *et al.*, "Enhancing hpc security with a user-based firewall," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2016, pp. 1–4.

[35] S. Peisert, "Security in high-performance computing environments," *Commun. ACM*, vol. 60, no. 9, p. 72–80, Aug. 2017. [Online]. Available: https://doi.org/10.1145/3096742

[36] K. Connelly and A. A. Chien, "Breaking the barriers: high performance security for high performance computing," in *Proceedings of the 2002 workshop on New security paradigms*, 2002, pp. 36–42.

[37] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein *et al.*, "The open science grid," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007, p. 012057.

[38] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science dmz: A network design pattern for data-intensive science," *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[39] C. TURCZYN. (2021) Nccs introduces citadel security framework. [Online]. Available: https://www.olcf.ornl.gov/2021/05/05/nccs-introduces-citadel-security-framework/

[40] N. I. of Standards and Technology. (1979) Fips-46: Data encryption standard. Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999, Available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[41] A. B. Association *et al.*, "Tripple data encryption algorithm modes of operation," *ANSI X9*, pp. 52–1998.

[42] P. Patil, P. Narayankar, D. Narayan, and S. M. Meena, "A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish," *Procedia Computer Science*, vol. 78, pp. 617–624, 2016.

[43] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public key cryptosystems," in *Secure communications and asymmetric cryptosystems*. Routledge, 2019, pp. 217–239.

[44] S. Li, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Resilient error-bounded lossy compressor for data transfer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476195

[45] UCSF. (2020) Update on it security incident at ucsf. [Online]. Available: https://www.ucsf.edu/news/2020/06/417911/update-it-security-incident-ucsf

[46] M. Mohan, M. K. Devi, and V. J. Prakash, "Security analysis and modification of classical encryption scheme," *Indian journal of science and technology*, vol. 8, no. 8, pp. 542–548, 2015.

[47] C. Lv and Q. Zhao, "Integration of data compression and cryptography: Another way to increase the information security," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, vol. 2. IEEE, 2007, pp. 543–547.

[48] M. M. P. Mr, C. A. Dhote, and D. H. S. Mr, "Homomorphic encryption for security of cloud data," *Procedia Computer Science*, vol. 79, pp. 175–181, 2016.

[49] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[50] J. Ji, D. Yan, and Z. Mu, "Personnel status detection model suitable for vertical federated learning structure," in *2022 The 6th International Conference on Machine Learning and Soft Computing*, ser. ICMLSC 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 98–104. [Online]. Available: https://doi.org/10.1145/3523150.3523166

[51] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Optimizing lossy compression rate-distortion from automatic online selection between sz and zfp," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1857–1871, 2019.

[52] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello, "Significantly improving lossy compression for hpc datasets with second-order prediction and parameter optimization," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 89–100.

[53] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," vol. 41, pp. A1278–A1303, 2019.

[54] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[55] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local shannon entropy measure with statistical tests for image randomness," *Information Sciences*, vol. 222, pp. 323–342, 2013, including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002002551200521X

[56] A. S. Fraenkel and S. T. Klein, "Complexity aspects of guessing prefix codes," *Algorithmica*, vol. 12, no. 4, pp. 409–419, 1994.

[57] D. W. Gillman, M. Mohtashemi, and R. L. Rivest, "On breaking a huffman code," *IEEE Transactions on Information theory*, vol. 42, no. 3, pp. 972–976, 1996.

[58] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, B. Julie, and F. Cappello, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *International Workshop on Big Data Reduction (IEEE IWBDR20) in conjunction with IEEE International Conference on Big Data (IEEE BigData20),*, 01 2021. [Online]. Available: https://sdrbench.github.io/

[59] "Scalable computing for advanced library and environment-regional model," https://scale.riken.jp/scale-rm.

[60] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "A survey on lora for iot: Integrating edge computing," in *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2019, pp. 295–300.

[61] A. Massoudi, F. Lefebvre, C. De Vleeschouwer, B. Macq, and J.-J. Quisquater, "Overview on selective encryption of image and video: challenges and perspectives," *Eurasip Journal on information security*, vol. 2008, no. 1, p. 179290, 2008.

[62] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Booz-allen and hamilton inc mclean va, Tech. Rep., 2001.

[63] B. Subramanyan, V. M. Chhabria, and T. S. Babu, "Image encryption based on aes key expansion," in *2011 Second International Conference on Emerging Applications of Information Technology*. IEEE, 2011, pp. 217–220.

[64] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full aes-192 and aes-256," in *International conference on the theory and application of cryptology and information security*. Springer, 2009, pp. 1–18.

[65] A. Akram, A. Giannakou, V. Akella, J. Lowe-Power, and S. Peisert, "Performance analysis of scientific computing workloads on trusted execution environments," *CoRR*, vol. abs/2010.13216, 2020. [Online]. Available: https://arxiv.org/abs/2010.13216

[66] J. Kim and J. S. Vetter, "Implementing efficient data compression and encryption in a persistent key-value store for hpc," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1098–1112, 2019. [Online]. Available: https://doi.org/10.1177/1094342019847264