Towards Combining Error-bounded Lossy Compression and Cryptography for Scientific Data

Ruiwen Shan,* Sheng Di,† Jon C. Calhoun,* Franck Cappello,†‡
*Clemson University, Clemson, 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

2021 IEEE High Performance Extreme Computing Conference (HPEC) | 978-1-6654-2369-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/HPEC49654.2021.9622874

***Abstract*—In the scientific domain, extremely large amounts of data are generated by large-scale high performance computing (HPC) simulations. Storing and sending such vast volumes of data poses serious scalability and performance issues, which can be considerably mitigated by data compression techniques which significantly reduced storage size and data movement burdens. Since scientific data are being shared by scientists more and more frequently, data security methods that ensure the confidentiality, integrity, and availability of data itself are becoming increasingly important. As such, combing compression and encryption is critical to storing large-scale datasets securely. In this work, we explore how to integrate data compression and cryptography techniques as efficiently as possible for big scientific datasets in the HPC field. We perform thorough experiments using different scientific datasets with the state-of-the-art error-bounded lossy compressor - SZ - on a real-world supercomputing environment. Experiments verify that performing encryption before lossy compression (a.k.a., encr-cmpr method) may invalidate the advantage of compression algorithms. By contrast, executing encryption after lossy compression (a.k.a., cmpr-encr method) keeps not only high compression ratios but high overall execution speed. Experiments also verify that the encryption overhead under the cmpr-encr method decreases with increasing compression ratios, which means very good scalability.**

***Index Terms*—data compression, cryptography, data security**

I. Introduction

High-performance computing (HPC) is one of the fastest-growing technical fields, which may produce extremely large volumes of data to process or analyze. Researchers conduct large-scale simulations in HPC and cloud environments to resolve challenging physical problems or exploit new strategies/products. Large-scale simulations such as the cosmological simulation code - HACC [1] are capable of generating more than 20 PB of down-sampled particle snapshots during one single full system run. Limited storage space is the first serious concern in practice. Many scientific projects or companies require a large number of devices to cope with the ever-increasing storage demand but larger storage capacity takes up the energy and resource expenditure of systems [2]. On the other hand, the I/O performance is also a big concern for the large-scale applications. One study by Cappello et al. [3] highlights that for supercomputers like IBM's Summit at the Oak Ridge National Laboratory, storing the full memory content to the parallel file system (PFS) may take more than 1 hour. Future systems will produce larger amounts of data, which means much higher cost to save the computation results.

Data compression has become one of the most promising solutions to the big-scientific-data problem. Lossless compression is generally not suitable for scientific data compression because most of lossless compressors depend on the repeated patterns in the data stream while scientific dataset is mainly composed of floating-point values, whose representations contain rather random ending mantissa bits. Error-bounded lossy compressors have been proposed for years to resolve the big scientific data compression issue, in that not only can it obtain fairly high compression ratios but it can strictly control the data distortion based on user-specified error bounds [4], [5].

In addition to data compression, data security in HPC systems is another critical topic, which has been paid increasing attentions in recent years [6], [7]. There are numerous scientific research projects funded by government agencies and they increasingly rely on large-scale computing systems. The related data managed by the scientific projects provide a basis for national security and policy decisions and sometimes can directly affect the future economy and security of a country. Crucially, these datasets must be trustworthy and have high integrity. Prior work shows that even a single bit-corruption in lossy compressed data can cause violations of the compressor's error bound or make the decompression process fail [8]. Moreover, the data could be intercepted/altered by malicious attacks inside the system or during the data transfer on wide area network (WAN). Cryptography algorithms are methods that protect the critical data that has a practical level of security to users [9].

How to combine error-bounded lossy compression and data security algorithms efficiently is a critical yet non-trivial issue in practice because of the following two key challenges. On the one hand, applying data security algorithms on either raw datasets or compressed datasets would affect the overall performance, so how to minimize the encryption overhead needs to be investigated carefully. In fact, since the main goal of HPC simulations is to accelerate the large and complex calculations in parallel, the security mechanisms must be fairly efficient and also effective such that it does not significantly degrade the overall computing performance. Prior work [10] studied the applicability of the hardware-based Trusted Execution Environments (TEEs) to achieve secure scientific computing. The results, however, showed that AMD Secure Encrypted Virtualization (SEV) may induce 1$\sim$4$\times$ performance slow-

down compared to native execution for graph applications. As such, the software-based encryption mechanism is the main focus of our work. On the other hand, the security/encryption algorithms always need to inject extra information, which, in turn, may degrade the overall compression ratios. How to minimize the overhead of encryption from the perspective of error-bounded lossy compression quality is non-trivial to answer, in that a comprehensive investigation needs to involve different datasets each with multiple fields as well as diverse settings related to encryption algorithms.

In this paper, we carefully investigate the interplay between error-bounded lossy compression and encryption algorithms, which is the first attempt to the best of our knowledge. We make the following contributions:

We analyze the interplay between the use of compression algorithms along with cryptography methods in the HPC field for scientific datasets based on two combination methods regarding encryption: compression-after-encryption (or encr-cmpr) and encrytion-after-compression (or cmpr-encr).

We evaluate the two methods on two HPC systems with real-world scientific datasets. Our takeaway is two fold:

- The cmpr-encr method does not significantly impact the overall execution performance when compression bandwidth is low and/or the compression ratio is high. By contrast, the encr-cmpr model may cause very significantly degraded compression ratios and overall throughput, because of considerably increased entropy after encryption.
- We model the overall time overhead that considers advances in compressor technology and show that the encryption/decryption overhead is low except for hard-to-compress data and fast compressors.

The rest of the paper is organized as follows. In Section II, we carefully analyze the use-cases of integrating the error-bounded lossy compression with cryptography. In Section III, we discuss two combination methods we investigate. In Section IV, we present and discuss the experimental results. We discuss the related work in Section V and conclude the paper in Section VI.

II. Use-cases of combining error-bounded lossy compression and cryptography

In this section, we analyze the use-cases that require both error-bounded lossy compression and cryptography in practice.

A. Fast and Secure Data Transmission

Data transmission to/from HPC systems is an important portion of many scientific workflows allowing users to share data with collaborators. The volume of data produced during the HPC application execution could be extremely large (such as 20TB for one single HACC simulation run, as mentioned previously). The bandwidth of the current internal interconnect network, however, is only around reached 200 Gb/s [11]. Transferring the vast volumes of data on Internet is even more challenging because of limited network bandwidth on WAN. Applying error-bounded lossy compression is indispensable to today's HPC applications and data sharing.

Data transmission faces a serious threat of eavesdropping or intercepting [12] in practice, and cryptography schemes offer effective solutions. Today's scientific projects often have security concerns for their sensitive data. Some data must be confidential at all times, and loss of valuable data can slow down or even stop the scientific process, leading to loss of scientific opportunities. Besides, the output of scientific instruments and applications is the basis for future analysis, so they must be protected against being compromised. Cryptography schemes ensure that even if the data is intercepted, the transmitted data is still unreadable to attackers. For the schemes that are adopted by the federal government, such as Advanced Encryption Standard (AES), a sufficient level of security is afforded to satisfy user's needs [13].

B. Effective and Safe Long-term Data Storage

Storing the large amount of data generated by simulations or instruments [3] in PFS requires both error-bounded lossy compression and cryptography technology. The compression technology is used to significantly reduce the size of scientific datasets so as to reduce the storage space, and the cryptography scheme is applied to guarantee the safety of the data stored in the persistent storage device. As mentioned previously, lossless compression cannot work effectively in compressing the scientific datasets that are composed of mainly floating-point values. Some studies [14] showed that lossless compression algorithms can only get compression ratios of $\sim$2 in most cases, while error-bounded lossy compressors [4], [5] can achieve compression ratios of 10$\sim$1000 also with satisfied reconstructed data quality. On the other hand, from the perspective of security, HPC systems are accessible by many people from multiple entities and locations, and the jobs could be executed with high permissions to a certain extent after they connect. Open science is particularly vulnerable because the resource is often fully exposed and tends to share large amounts of data or experimental results with several entities. All these characteristics make them a notable target for attackers. The data saved in storage systems have high post hoc analysis value and are vulnerable to attackers. In this case, cryptography schemes are needed to prevent the content of the data from being intercepted by unauthorized access. Encryption algorithms transform the plain data files to ciphertext such that only the users with the matched keys can decrypt the content. Such encryption methods have a fairly high security level. For example, AES-128 has a 128-bit long key, which means there are 2^{128} possible keys [15]. If a brute-force attack is launched to decrypt the AES-128-encryped data on a supercomputer even with 10^5 key searches per nanosecond, it still requires nearly 5.3×10^{17} years to try all possible keys.

C. Hindering Cryptanalysis Attacks

Data compression could benefit the cryptographic process from the security point of view. Cryptanalysis is the subject of obtaining the plaintext from the ciphertext without knowing the key and cryptographic algorithm. A common cryptanalysis

978-1-6654-2369-4/21/$31.00 ©2021 IEEE

method is frequency analysis, and the ciphertext with higher redundancies is easier to crack [16]. By contrast, data compression shrinks the volume of data by removing redundancy. In other words, the redundancy that is reduced by data compression can potentially hinder certain cryptanalysis attacks and decrease their effectiveness. Besides, the less ciphertext there is, the fewer clues it includes and it is harder for attackers to break cryptography [17]. Even if attackers get the key through other means or crack the cryptographic algorithm, they can only get the compressed data, the unification of compression and cryptography techniques will further increase the difficulty for opponents to reconstruct the original data sets thus prevent them from obtaining the content.

III. Analysis of Two Combination Methods

In this section, we provide an in-depth analysis of the two candidate methods: compression after encryption (or *encr-cmpr*) vs. encryption after compression (or *cmpr-encr*).

A. Compression after Encryption (Encr-Cmpr)

In the encr-cmpr method, encryption occurs before compression. This kind of cryptosystem is for people who want to encrypt sensitive data as early as possible to prevent untrusted entities from accessing the data or transmitting data over an insecure channel. In this case, the untrusted transmitter or compressor has no privilege to access the private content.

Generally, the encr-cmpr method is inherently format compliant but it may adversely impact compression efficiency [13]. Cryptography methods change scientific data sets from floating-point to random bytes which is not compatible with lossy compression. However, this reversed system may be preferable because compression schemes may suffer from information leakage caused by observable properties such as the compression ratio in some cases. For example, if the attacker enumerates a set of possible inputs and knows the compression algorithm, he/she can use the length of the input plus the compression ratio as a checksum to derive the input [18]. This is considered a side-channel attack. Assuming this attack occurs in an HPC system that processes with biomedical data and if the attacker somehow gets the information, then it is possible for him/her to obtain the input data. Once this private information is leaked, the consequences can be disastrous. The encr-cmpr method avoids this kind of information leakage. Even though the attacker gets knowledge of the compression algorithm and obtains the compressor's input, it has already been encrypted, so the attacker is unable to obtain the content without the secret key. A reverse cryptosystem can still achieve perfect security and the same compression efficiency as performing lossy compression before encryption for any fixed distortion if the original source is Gaussian [19].

B. Encryption after Compression (Cmpr-Encr)

The cmpr-encr method performs the compression before encryption. Compared to the encr-cmpr method, it has no impact on the compression quality, except for a very small overhead (i.e., encryption key) to be introduced. Moreover, there is no need to modify the encoder and decoder.

Our analysis shows that the cmpr-encr method is more suitable for scientific data compression in most of cases, from the perspective of both compression quality and security concern. (1) Not only does error-bounded lossy compression eliminate the correlation between each data point but it also reduces the predictability of the cryptography process [13]. (2) The cmpr-encr method is an excellent way to take advantage of both techniques (lossy compression and encryption). On the one hand, cryptographic schemes resolve security problems such as interception and confidentially that are neglected during compression. On the other hand, the error-bounded lossy compression can reduce the data size significantly, thereby making the encoding process much faster and more efficient than without the compression technique applied in advance.

All in all, this combination method has a minor influence on the performance of the compressor itself, retaining the compression ratio very well. Furthermore, as a relatively fast cryptographic scheme is adopted, the overall performance overhead can be very limited. We show the advantage of this approach in Section IV.

C. Selection between Encr-Cmpr vs. Cmpr-Encr

In principle, the cmpr-encr is superior to the encr-cmpr method, considering the following reason. An ideal encryption algorithm should be able to remove all the redundancies in the plaintext and produce a ciphertext that has an uniform distribution of characters. Error-bounded lossy compression is a method that significantly relies on the redundancy and correlation of the raw data. If the encryption scheme is performed before lossy compression (i.e., encr-cmpr method), the compression ratio is significantly degraded because of considerably lower correlation in the encrypted data. By comparison, the cmpr-encr method performs the lossy compression first, which takes full advantage of the correlations in the dataset to get high compression ratios. The encryption algorithms performed after lossy compression inject tiny additional information, to further improve the security level.

In order to verify how much the compressibility of data changes after the encryption, we compare the Shannon entropy for the original raw data versus the encrypted data based on the real-world scientific datasets. The scientific datasets we use here are described in details in next section. Shannon entropy is a measure of the unpredictability or randomness of information content, which is defined in Formula (1).

$$H = -\sum p(x) \log p(x) \tag{1}$$

where H is value of entropy, $p(x)$ is the probability of occurrence of the symbol x in the dataset, and $\log p(x)$ is the information quantity carried by observing x. If the entropy H is high, then the distribution of each symbol is uniform, which means the probability of occurrence for each symbol is similar, so it is hard to compress the data. In contrast, if the entropy is low, this means the occurrence of some symbol are more frequent than others, and the dataset is easier to compress.

978-1-6654-2369-4/21/$31.00 ©2021 IEEE

TABLE I: Entropy calculated before and after encryption.

Datasets	Entropy(Original)	Entropy(Encrypted)
Height	7.468389	8.0000
Q2	6.671749	7.999997
QI	5.894355	8.0000
T	6.618928	8.0000

Table I presents the entropy value of the original raw data versus the encrypted data. It is observed that the entropy value is always increased prominently after data encryption. In fact, when the dataset is ideally encrypted, the entropy reachs the theoretical maximum [20] which makes it extremely difficult to compress. Floating-point scientific data has high diversity and needs strict accuracy for post hoc analysis [9]. The entropy of the original data is already high ($\sim$6 as shown in the table), and the encryption algorithm further increases the entropy, meaning that the encrypted data in turn is much harder to compress than before.

IV. Evaluation Results

A. Experimental Setup

To study the effectiveness of applying error-bounded lossy compression and encryption schemes together on various real-world scientific datasets, we use two HPC systems, Palmetto at Clemson University and Bebop at Argonne National Laboratory. Table II shows the specification of each system.

TABLE II: Detailed hardware used for experiments

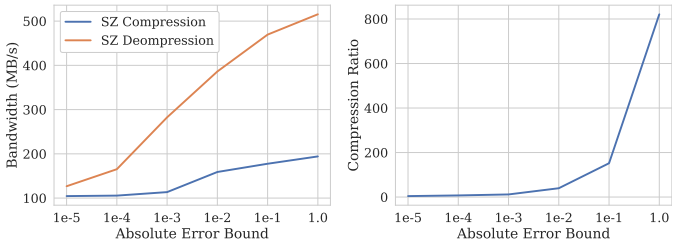
	Palmetto	Bebop
CPU	Intel Xeon Gold 6148	Intel Xeon E5-2695v
RAM	376 GB	125 GB
Cache	27.5 MB	48 MB
Cores	20	18

To investigate the combination of lossy compression and encryption of HPC data, we use *Wf04* from the Hurricane Isabel simulation in SDRBench [21] and four typical fields *Height*, *Q2*, *QI*, and *T* from SCALE-LetKF [22], a atmospheric modeling code. The datasets we use in experiments are summarized in Table III.

TABLE III: Attributes of the datasets used in experiments.

dataset	Dimensions	Size	Description
Wf04	100×500×500	95.37MB	Hurricane wind speed
Height	98×1200×1200	1.1GB	Height above ground
Q2	11×1200×1200	61MB	2m Specific humidity
QI	11×98×1200×1200	5.8GB	Cloud Ice mixing ratio
T	11×98×1200×1200	5.8GB	Temperature

We execute the error-bounded lossy compressor SZ-2.1 [23] with absolute error bound mode by LibPressio [24], a generic library binding to abstract between different compressors and their configurations. We focus on SZ as it has been proven to have the best overall compression performance and quality from among existing state-of-the-art lossy compressors [4], [25]. As for the encryption, we explore several methods via OpenSSL [26]: Data Encryption Standard (DES), 3DES, Advanced Encryption Standard (AES), Blowfish. GCC 8.3.1 compiles all of our code, and we run all codes with a single thread of execution. Each data point is an average of five runs.



(a) SZ Throughput. (b) SZ Compression Ratio.

Fig. 1: Compression performance of SZ on Hurricane data.

B. Separate Cost of Compression and Encryption

To compare the execution throughput (or bandwidth) between compression/decompression and encryption/decryption on scientific data, we use the combination methods using the Hurricane dataset on Clemson's Palmetto Cluster. Figure 1a shows the throughput (MB/s) of compression and decompression under SZ. In Figure 1b, as the error bound increases, the throughput increases, and more distortion is observed in the data with higher compression ratios. We see a maximal compression bandwidth of 194 MB/s and a decompression bandwidth of 514 MB/s per CPU core. Mathematical calculations such as prediction and quantization in the compression process makes its bandwidth smaller than decompression.

For encryption to have minimal impact on compression we require an encryption bandwidth that is larger than the compression bandwidth. To determine the throughput of data encryption and decryption, we use random datasets of various sizes to pinpoint an exact file size. The encryption algorithms are data agnostic. Figure 2 shows the encryption and decryption bandwidth for our selected algorithms. Looking at the performance, we see that there is a large discrepancy in performance between the algorithms. For DES, 3DES, and Blowfish we see small bandwidth improvements as the input data size increases. However, for AES, we see that the bandwidth dramatically increases for larger data files. We use the Cipher Block Chaining(CBC) mode of AES because CBC mode is able to introduce a limited amount of metadata to the compressor while ensuring a sufficient security level. For AES-128, the throughput exceeds 400 MB/s for encryption and 490 MB/s for decryption. Therefore, for AES the time overhead between compression and encryption does not significantly impact total reduction time.

This experiment demonstrates that encryption would not significantly impact the time to compress data for lower error bounds when compression bandwidth is low; however, as the error bound and compression ratio increases, the size of the data to encrypt is less and encryption performance suffers. For other algorithms like DES, 3DES, and Blowfish, the bandwidths are low. Most of them are lower than 100MB/s, and if we integrate these methods with SZ, the overhead would cause dramatic decreases in the compression and decompression bandwidth. For the remaining experiments, we utilize AES-128 because it has the best performance.

978-1-6654-2369-4/21/$31.00 ©2021 IEEE

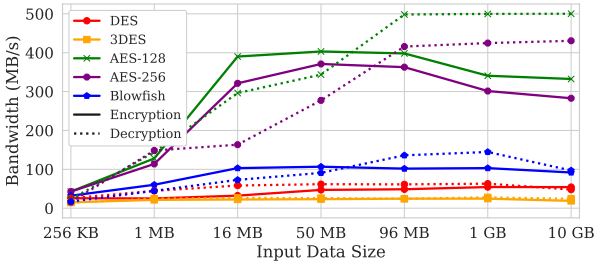


Fig. 2: Througphput of Encryption and Decryption.

C. Combining Compression and Encryption

From the results in Section IV-B, we see that the cost of adding encryption into the data reduction varies, depending on the throughput of the compressor and the encryption algorithm. To further explore this trade-off, we use the four SCALE-LetKF datasets and test them on the Bebop cluster at Argonne.

We now quantify the impact on the compression ratio. From Table IV (our compression ratio baseline), we notice that, even though the compression ratio of each datasets depends on its own characteristics, it increases with larger error bounds. Compared to other datasets, QI is the most compressible with compression ratios in the range of 70–1.41E+08×, while T has a low compression ratio 2–16×. The reason is that SZ depends on prediction accuracy, and QI has a stronger data correlation then T which has more random/disordered data values, which is validated by their entropy values (see Table I).

TABLE IV: Baseline compression ratio with no encryption.

	Absolute Error Bound				
Dataset	**1e-7**	**1e-5**	**1e-4**	**1e-3**	**1e-2**
Height	1.927854	5.732317	7.522575	12.08979	23.98324
Q2	4.246728	12.57578	31.41299	209.5349	1662.381
QI	70.94958	591.3201	4235.384	18413.63	1.41E+08
T	2.854135	2.737459	5.065659	9.485292	16.98267

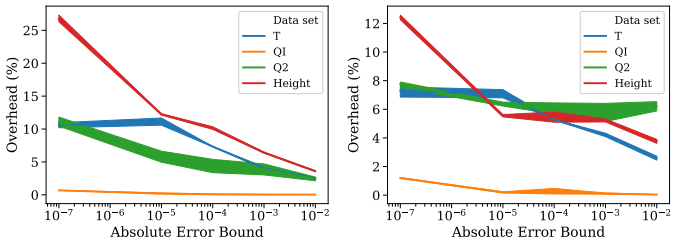
Encrypting the data allows opportunities to diminish the compression ratio through increasing the entropy and/or extra data. Here we explore two methods of integrating encryption.

1) Encr-Cmpr: Table V shows the performance of each stage in the encr-cmpr pipeline. All error bounds show similar performance. Notably, the compression ratio for all four datasets is low; most lower than 1. Compression ratios being lower than 1 indicates inflation of the file size. We attribute this to AES's obfuscation operations, which eliminate data correlations; making the ciphertext appear random and increasing the data size. Moreover, the lossy compressor also needs to store some metadata — e.g., Huffman tree, lookup table — , which also increases the size in turn. Compared to the compression ratios in Table IV, we see a sharp decline in the compression ratio. Thus, we conclude that naïve encr-cmpr as two disjoint passes is not suited for HPC use-cases because of poor compression ratios.

2) Cmpr-Encr: Cmpr-encr applies encryption after compression. Thus, the lossy compressor is able to take full advantage of data correlations. Figure 3 shows the overhead of encryption/decryption when compressing/decompressing the SCALE-RM datasets, where shading represents the standard deviation. We see similar overhead for compression and decompression among the datasets, but we note that the decompression overhead is low due to the fast decompression bandwidth of SZ (see Figure 1a). In general, as the compression ratio drops, the overhead increases as the size of the data turns larger and encryption takes longer. The overhead for QI is quite low (<1.5%) because its large compression ratio leads to a small amount of data to encrypt. Compressing T with an error bound of 1e-5 has slightly higher overhead than the overhead with the error bound of 1e-7 because the compression ratio drops slightly and thus increasing encryption time. The variable $Height$ exhibits the largest overhead for compression and decompression since it has the lowest compression ratio and therefore the largest encryption time. Observing the *effective* compression ratios of cmpr-encr in Table VI shows that most situations yield less than a 0.01% deterioration in compression ratio compared to the those in Table IV. However, for datasets that has extremely high compression ratios before encrypting, the overhead of metadata becomes far more visible because the input data for encryption is very small. In summary, we conclude that cmpr-encr retains 99.9% of the compression ratio in most cases with decreasing overhead as the compression ratio increases.

TABLE V: Performance of different datasets with encr-cmpr.

Data Set	**Encrypt BW(MB/s)**	**Compress BW(MB/s)**	**Decompress BW(MB/s)**	**Decrypt BW(MB/s)**	**Compr. Ratio**
Height	372.25	327.45	394.23	631.47	1.99
Q2	290.47	151.68	375.35	670.34	0.99
QI	402.13	160.81	361.52	693.63	0.99
T	343.84	146.49	335.46	730.77	0.99



(a) Compression. (b) Decompression.

Fig. 3: Overhead in time when adding encryption after compression and decrypting before decompression.

D. Modeling the Combination of Compression and Encryption

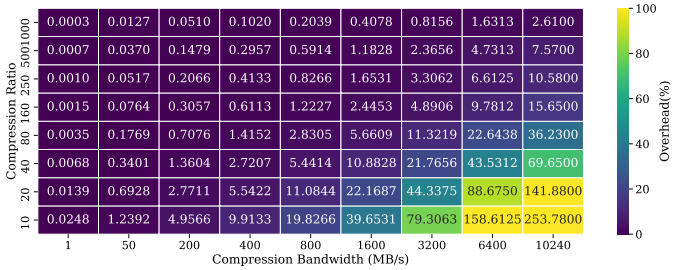
The size of the dataset may significantly grow in future HPC systems. To explore the compression/encryption overheads for larger data files, we tested a 100 GB data file. We model the expected time overhead for cmpr-encr where we parameterize the compressor to account for improvements in performance and compression ratio. As for the encryption/decryption's performance, we use the data from Figure 2.

Using our model, we plot Figure 4, where the overhead is the percentage encryption/decryption time compared to compression/decompression. Darker blocks indicate low overhead, while lighter blocks indicate high overhead. We cap the color bar at 100% to clearly highlight regions. Actual percentages

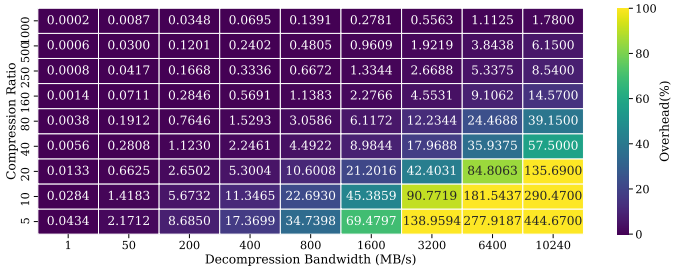
978-1-6654-2369-4/21/$31.00 ©2021 IEEE

TABLE VI: Compression ratio after encryption. Value in parentheses is the percentage that we obtain from Table IV.

	Absolute Error Bound				
Dataset	**1e-7**	**1e-5**	**1e-4**	**1e-3**	**1e-2**
Height	1.93 (99.9%)	5.73 (99.9%)	7.52 (99.9%)	12.09 (99.9%)	23.98 (99.9%)
Q2	4.25 (99.9%)	12.58(99.9%)	31.41(99.9%)	209.51(99.9%)	1661.07 (99.9%)
QI	70.95 (99.9%)	591.32(99.9%)	4235.29(99.9%)	18412.49(99.9%)	9.7E+07 (68.8%)
T	2.85 (99.9%)	2.74(99.9%)	5.06(99.9%)	9.49(99.9%)	16.98 (99.9%)



(a) Compression + Encryption



(b) Decrypting + Decompression

Fig. 4: Performance Overhead of The Cmpr-Encr Method.

are displayed on each grid cell. For example, for compression ratio 40× and 200MB/s compression throughput, the overhead of AES is 1.4%, which means it takes a small fraction of the total time to encrypt the data. Mapping to the real-world use case, such as using SZ to compress $Wf04$, the compression ratio ranges from 4× to 821×, while the compression throughput is between 100M–200MB/s. These points map to darker blocks (low overhead). This holds true for decompression and decryption too. However, for the situations with high compression/decompression throughput, higher compression ratios are required to avoid sizable overheads. Thus, for hard-to-compress data, encryption dominates performance by 253% (compression) and 444% (decompression).

V. Related work

Combining compression and encryption has been studied in the multimedia field. Most research focuses on integrating selective encryption within compression, where only the essential part of the compressed bitstream is encrypted to achieve several levels of security. For example, Lei [27] encrypts only the I-frames of the MPEG stream. DC and AC coefficients are encrypted in [28]. A encryption-after-compression method is proposed for JPEG2000 in [29], where the algorithm identifies a start and stop of a data packet to encrypt using AES-Cipher Feedback mode. The highest visual degradation is achieved by encrypting 20% of the data and there is no impact on compression performance. Compressive sensing is a signal-acquisition framework that performs compression and encryption simultaneously [30]–[32], but cannot control compression errors as error-bounded lossy compressor (such as SZ and ZFP) does.

The algorithms proposed in the multimedia field generally use static definitions of encrypted parts and encryption parameters. This limits the usability of the algorithm to a restricted set of compressors and applications that is not feasible for scientific data. When it comes to HPC systems, the serious challenge we have to face is how to use the characteristics of lossy compressors themselves to better integrate compression and security without changing the traditional encryption algorithm that already has a sufficient level of security.

An encryption-after-compression method that only compresses and encrypts the partial value in the key-value pair and Sorted Strings Table(SSTable) is implemented in the National Energy Research Scientific Computing Center [9]. They found that the overhead of compression and encryption for data is much smaller than the I/O overhead in NVM devices. Some researchers are dedicated to improving security by transforming the compressor. Mazharul et. al proposed a Huffman coding-based encryption scheme HELiOS for data transmission [33], where a dynamic order statistic tree is built to compress the plain text. HELiOS encrypts the classical Huffman tree and chaotic random seed using the receiver's public key. Attackers are not able to reconstruct the statistic tree and without the possession of the receiver's private key.

Compared with all the related studies, we explore how to combine error-bounded lossy compression and encryption technology and also investigate the pros and cons of different solutions using real-world scientific datasets on HPC systems, which is the first attempt, to our best knowledge.

VI. Conclusion

This paper analyzes and quantifies the pros and cons two solutions by combining error-bounded lossy compressor and different encryption schemes based on real-world scientific datasets. Our experimental results demonstrate that the encr-cmpr method is not an ideal combination method since the latter compression method is unable to take advantage of data redundency and correlation, thus leading to very low compression ratios. By contrast, the cmpr-encr method performs well in preserving a good compression ratio and suffers little encryption overhead, which is acceptable in most cases. As the compression ratio increases, the overhead of encryption decreases. In the future, we plan to investigate the cmpr-encr method using more datasets and more security levels.

978-1-6654-2369-4/21/$31.00 ©2021 IEEE

ACKNOWLEDGMENT

The material was supported by U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357, also by National Science Foundation under Grant No. SHF-1910197, SHF-1943114, and OAC-2003709.

REFERENCES

[1] 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.

[2] Y. Dong, J. Chen, and T. Tang, "Power measurements and analyses of massive object storage system," in *2010 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 1317–1322.

[3] 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. [Online]. Available: https://doi.org/10.1177/1094342019853336

[4] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, 2016, pp. 730–739. [Online]. Available: https://doi.org/10.1109/IPDPS.2016.11

[5] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, Dec 2014.

[6] 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

[7] 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*, ser. NSPW '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 36–42. [Online]. Available: https://doi.org/10.1145/844102.844109

[8] 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 24th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '21. New York, NY, USA: ACM, 2021.

[9] 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

[10] 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

[11] Z. Jin, Z. Lu, H. Li, X. Chi, and J. Sun, "Origin of high performance computing——current status and developments of scientific computing applications." *Bulletin of Chinese Academy of Sciences*, vol. 34, no. 6, pp. 625–639, 2019.

[12] 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.

[13] 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 J. Inf. Secur.*, vol. 2008, no. 1, Dec. 2008.

[14] P. Lindstrom, "Error distributions of lossy floating-point compressors," 10 2017. [Online]. Available: https://www.osti.gov/biblio/1526183

[15] L. Lapworth, "Parallel encryption of input and output data for hpc applications," *The International Journal of High Performance Computing Applications*, vol. 0, no. 0, p. 10943420211016516, 0. [Online]. Available: https://doi.org/10.1177/10943420211016516

[16] M. Mohan, M. Devi, and V. Prakash, "Security analysis and modification of classical encryption scheme," *Indian journal of science and technology*, vol. 8, pp. 542–548, 2015.

[17] 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, 2007, pp. 543–547.

[18] J. Kelsey, "Compression and information leakage of plaintext," in *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2365. Springer, 2002, pp. 263–276. [Online]. Available: https://iacr.org/archive/fse2002/23650264/23650264.pdf

[19] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2992–3006, 2004.

[20] 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

[21] 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/

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

[23] 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 *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, 2018, pp. 438–447. [Online]. Available: https://doi.org/10.1109/BigData.2018.8622520

[24] libpressio, https://github.com/CODARcode/libpressio, 2019.

[25] 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,, Aug 2019, 10.1109/TPDS.2019.2894404.

[26] "Openssl," https://www.openssl.org.

[27] L. Tang, "Methods for encrypting and decrypting mpeg video data efficiently," in *Proceedings of the Fourth ACM International Conference on Multimedia*, ser. MULTIMEDIA '96. New York, NY, USA: Association for Computing Machinery, 1997, p. 219–229. [Online]. Available: https://doi.org/10.1145/244130.244209

[28] J. Meyer, "Security mechanisms for multimedia-data with the example mpeg-1 video," *http: www. gadegast. de/frank/doc/secmeng. pdf*, 1995.

[29] R. Norcen and A. Uhl, "Selective encryption of the jpeg2000 bitstream," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2003, pp. 194–204.

[30] H. Gan, S. Xiao, and Y. Zhao, "A novel secure data transmission scheme using chaotic compressed sensing," *IEEE Access*, vol. 6, pp. 4587–4598, 2018.

[31] W. Xue, C. Luo, G. Lan, R. Rana, W. Hu, and A. Seneviratne, "Kryptein: A compressive-sensing-based encryption scheme for the internet of things," in *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2017, pp. 169–180.

[32] H. Peng, Y. Tian, J. Kurths, L. Li, Y. Yang, and D. Wang, "Secure and energy-efficient data transmission system based on chaotic compressive sensing in body-to-body networks," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 3, pp. 558–573, 2017.

[33] M. Islam, N. Nurain, M. Kaykobad, S. Chellappan, and A. B. M. A. A. Islam, "Helios: Huffman coding based lightweight encryption scheme for data transmission," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MobiQuitous '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 70–79.

978-1-6654-2369-4/21/$31.00 ©2021 IEEE