RESEARCH ARTICLE

Joint Compression and Encryption of Virtual Machine Images Using LZMA and Chaotic Maps

Muhammad Usama[1,*] **and Jawad Ahmad**[2]

[1] Department of Cyber Security, Pakistan Navy Engineering College, National University of Sciences and Technology, Karachi 75350, Pakistan
[2] Cyber Security Center, Prince Mohammad Bin Fahd University, Al-Khobar, Saudi Arabia

Abstract

The exponential growth in security and storage requirements has led to increasing demand for high-performance solutions, particularly in managing large-scale data objects such as big binaries. Virtual Machines (VMs), a prime example of such binaries, are widely employed across desktops and dynamic server environments to deliver isolated execution contexts while minimizing hardware overhead. However, the proliferation of big binaries imposes significant challenges on both storage infrastructure and secure data transmission mechanisms. To address these issues, this paper proposes a novel "Combined Big Binaries Compression and Encryption" (CBBCE) scheme that jointly performs compression and encryption in a unified framework. The proposed approach integrates Lempel–Ziv–Markov chain algorithm (LZMA) for efficient compression with a stream cipher-like encryption mechanism. A chaotic logistic map is employed as a key-controlled pseudorandom bitstream generator due to its high sensitivity to initial conditions, ergodicity, and long-period randomness. The compressed data stream is subsequently encrypted using this pseudorandom sequence to ensure confidentiality. Experimental evaluations demonstrate that the CBBCE approach achieves competitive compression ratios and enhanced security properties when compared to conventional sequential compression-then-encryption schemes. The proposed method is particularly well-suited for secure storage and transmission of large binary files, offering an effective trade-off between performance, security, and storage efficiency.

Keywords: chaos, compression, encryption, secure compression, compression and encryption, big binaries, virtual machines.

1 Introduction

With the exponential growth of digital data, the demands for efficient security and storage solutions have intensified. To address these demands, increasingly larger data objects, commonly referred to as big binaries, are being generated. A prominent example of such big binaries is virtual machines (VMs), which are extensively deployed in modern desktop and server environments to deliver isolated, logically

Submitted: 02 July 2025
Accepted: 05 July 2025
Published: 31 July 2025

Vol. 1, **No.** 1, 2025.
10.62762/TISC.2025.148753

***Corresponding author:**
✉ Muhammad Usama
musama@pnec.nust.edu.pk

Citation
Usama, M., & Ahmad, J. (2025). Joint Compression and Encryption of Virtual Machine Images Using LZMA and Chaotic Maps. *ICCK Transactions on Information Security and Cryptography*, 1(1), 3–12.

separate execution environments without the need for dedicated physical hardware.

In large-scale server infrastructures, virtualization enables the consolidation of multiple virtual servers onto fewer physical machines. This not only reduces hardware acquisition costs but also lowers operational overhead, including energy consumption, cooling, and system management. Additionally, virtualization offers fine-grained control over individual virtual instances, allowing for diverse use cases and independent management. However, this flexibility comes at a cost: the need to store and manage massive multi-gigabyte VM images. As the prevalence of big binaries continues to rise, the dual challenges of storage optimization and secure transmission become critical. Compression and encryption have therefore become essential components in managing these digital assets. When employed effectively, they not only reduce storage requirements but also provide robust security, serving as foundational technologies for backup systems, archival solutions, and secure communications.

Compression and encryption are both widely used in data management systems and play a vital role in various domains, including multimedia processing, medical data transmission, shared networks, and military communications. Traditionally, compression and encryption have been applied sequentially, data is first compressed and then encrypted. While effective, this approach introduces latency and requires considerable computational resources, especially when dealing with high-volume data such as big binaries. An integrated approach that combines compression and encryption into a unified process presents a promising alternative. This strategy reduces data redundancy and transmission time through compression, while simultaneously enhancing resistance to cryptanalytic attacks by encrypting the compressed data. Previous attempts at such integration, however, have often struggled to balance compression efficiency with cryptographic strength, leading to trade-offs that compromise either storage compactness or security robustness.

In light of the growing need for secure and efficient handling of big binaries, this paper proposes a novel framework termed *Combined Big Binaries Compression and Encryption* (*CBBCE*). The proposed method utilizes the Lempel–Ziv–Markov chain algorithm (LZMA) for high-ratio compression and incorporates a pseudorandom bitstream generator based on the chaotic logistic map to achieve encryption. The chaotic nature of the logistic map offers key sensitivity, ergodicity, and long-period randomness, essential characteristics for secure cryptographic operations.

The remainder of this paper is organized as follows: Section 2 reviews the existing literature on compression and encryption integration. Section 3 introduces the proposed CBBCE algorithm. Section 4 presents experimental results and performance evaluation. Finally, Section 7 concludes the paper and outlines future directions.

2 Related Work

The intersection of chaotic systems and cryptography has been extensively explored due to the inherent characteristics of chaos, such as sensitivity to initial conditions, ergodicity, and randomness. However, the integration of chaos-based encryption with data compression remains a relatively nascent area of research. Existing approaches in this domain can be broadly categorized into two groups: compression-oriented and encryption-oriented algorithms [3, 4].

In encryption-oriented schemes, data compression is embedded within an encryption framework. These methods typically exhibit suboptimal compression performance, with compression ratios ranging between 10% and 17% [5], which are significantly lower than those achieved by standalone compression algorithms. This limitation stems from the fundamental contradiction between the goals of compression and encryption [17]. Compression exploits redundancy and patterns to reduce data size, while robust encryption aims to eliminate such patterns, producing output with high entropy, ideally approaching 8 bits per byte.

Compression-oriented schemes, where encryption is embedded into the compression process, are comparatively less explored. Notable early contributions include [6, 7], which introduced binary arithmetic coding with key-based interval splitting and integrated multimedia compression with encryption, respectively. However, these methods were later shown to be vulnerable to known-plaintext attacks [8].

Subsequent work sought to overcome these limitations. For instance, the authors in [9] improved upon [6] by removing interval constraints and introducing dual permutations for enhanced diffusion. In [10], a randomized arithmetic coding (RAC) approach was proposed for the JPEG 2000 standard, embedding encryption directly into the compression process by injecting randomness into the arithmetic coder. Nevertheless, the performance of these schemes remained inferior compared to standard compression-encryption pipelines [9].

A novel approach based on chaotic dynamical systems, specifically the Generalized Lurőth Series (GLS), was introduced in [20], combining non-linear dynamics with arithmetic coding for simultaneous compression and encryption. Other research efforts, such as [11, 21], proposed key-controlled compression methods using chaotically mutated Huffman trees. In [7], the Huffman tree branches were swapped using key-based rules, while [21] extended this by resolving multiple codeword issues and enlarging the key space to address weak security in earlier models.

Further enhancements were presented in [11], where two chaotic functions were employed to resist known-plaintext attacks. Although such schemes represent progress, most continue to suffer from critical security flaws [8, 12, 13], rendering them unsuitable for applications involving high-volume data such as big binaries. While a range of techniques have been developed to integrate compression and encryption, there remains a trade-off between security robustness and compression efficiency. Many existing approaches either compromise on performance or fail to offer strong cryptographic guarantees. This paper aims to address these limitations by proposing a novel joint compression and encryption framework specifically tailored for large binary data, combining LZMA compression with a stream cipher based on a chaotic logistic map to achieve both efficiency and security.

3 Proposed Work

This paper proposes a joint compression-encryption framework tailored for large binary objects, particularly in resource-constrained computational environments such as virtualized systems. The primary design objectives are efficiency and reliability in both compression and encryption processes. To this end, a stream cipher-like architecture is adopted, wherein compression is embedded within the encryption pipeline. The system first compresses the input using the Lempel–Ziv–Markov chain algorithm (LZMA), followed by encryption through a bitwise XOR operation with a pseudorandom bitstream generated by a chaotic logistic map.

3.1 Chaotic Logistic Map

For the encryption component, the chaotic logistic map is employed to generate a pseudorandom bitstream. The logistic map is particularly suitable for cryptographic applications due to its properties of ergodicity, high sensitivity to initial conditions, mixing behavior, and tunable control parameters [14, 22]. It is a one-dimensional nonlinear system that exhibits chaotic behavior under appropriate conditions and can be defined by the recursive equation:

$$x_{n+1} = \delta x_n(1 - x_n), \tag{1}$$

where $x_n \in [0, 1]$ denotes the state of the system at iteration n, x_0 is the initial condition, and $\delta \in (3.57, 4]$ is the control parameter ensuring chaotic dynamics. The sequence generated through iterations of this function forms the basis for the pseudorandom bitstream, which is then used to encrypt the compressed data via XOR operation. The high unpredictability and sensitivity of the logistic map enhance the security of the encryption process.

3.2 Compression Algorithm

Let $\texttt{Algo}$ represent a compression algorithm, ∂ be a security parameter, ϵ denote the empty string, and $\texttt{Algo}(\alpha)$ represent the set of possible outputs when $\texttt{Algo}$ is executed on input α. Let $x \overset{\$}{\leftarrow} S$ denote uniform random sampling from set S, $\circ$ be the string concatenation operator, and $l(s)$ denote the length of string s. Let $\mathcal{E}$ represent the space of encrypted messages, and $\mathcal{M}$ denote the space of original data. Also, let $\mathcal{P}(n)$ represent the set of all permutations over $\{0, 1\}^n$.

Data compression is formally defined as an encoding of a data sample such that the expected length of the encoded message is less than or equal to the original. Given a message space $\mathcal{M} \subseteq A^*$ over an alphabet A, and a probability distribution $\mathcal{D}$ over $\mathcal{M}$, an encoding function is a map $E : \mathcal{M} \to \{0, 1\}^*$. If the code is uniquely decodable, that is, no codeword is a prefix of another, the function E is considered valid.

The average code length under distribution $\mathcal{D}$ is defined as:

$$L(E) = \sum_{m \in \mathcal{M}} p(m) \cdot l(E(m)), \tag{2}$$

where $p(m)$ denotes the probability of message m under distribution $\mathcal{D}$, and $l(E(m))$ is the length of its encoding. A function E satisfying $L(E) \le \mathbb{E}[l(m)]$ is considered a valid compression function. The inverse function D, mapping codewords back to messages in $\mathcal{M}$, is the corresponding decompression function.

The Algorithm 1 describes a generic LZ-based compression process suitable for big binary data.

Algorithm 1: Dictionary-Based Compression (LZ-like)

```
Data: Character stream $I$
Result: Stream of dictionary indices
Initialize dictionary $T$ with all single-character strings;
$prefix \leftarrow \epsilon$;
$i \leftarrow$ number of single-character strings;
while more input exists in $I$ do
    Read next character $c$;
    if $prefix \circ c \in T$ then
        $prefix \leftarrow prefix \circ c$;
    else
        Output index of $prefix$ from $T$;
        $T[i] \leftarrow prefix \circ c$;
        $i \leftarrow i + 1$;
        $prefix \leftarrow c$;
    end
end
if $prefix \neq \epsilon$ then
    Output index of $prefix$ from $T$;
end
```

Algorithm 1 underpins the compression stage of the proposed framework. It ensures that the input stream is effectively encoded using dictionary-based substitutions, enabling significant reduction in data size prior to encryption.

3.3 Combined Compression and Encryption Algorithm

To achieve integrated data confidentiality and storage efficiency, we define a *keyed compression function*, a compression mechanism parameterized by a cryptographic key k, such that correct decompression is only possible using the corresponding key (see Algorithm 2). A trivial instantiation is a traditional compression scheme where the key is fixed (e.g., $k = \varepsilon$), providing no additional privacy. Conversely, a keyed cipher that is length-preserving also satisfies this definition, although it provides no inherent compression.

Definition: A keyed compression function over a message space $\mathcal{M} \subseteq A^*$ is a triple of (possibly probabilistic) polynomial-time algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, where:

- $\mathcal{K}(1^\lambda) \to k$ is the key generation algorithm taking a security parameter λ and producing a key k.
- $\mathcal{E}(k, m) \to e \in \{0,1\}^*$ is the keyed compression function which maps message $m \in \mathcal{M}$ to a compressed ciphertext e.
- $\mathcal{D}(k, e) \to m \in \mathcal{M} \cup \{\perp\}$ is the decompression function which returns the original message or failure symbol $\perp$.

Correctness requires that for all $m \in \mathcal{M}$, and all keys $k \leftarrow \mathcal{K}(1^\lambda)$, it holds that:

$$\mathcal{D}(k, \mathcal{E}(k, m)) = m. \quad (3)$$

As part of our construction, we utilize a pseudorandom permutation $f_k : \{0,1\}^n \to \{0,1\}^n$ in the key derivation function (KDF), incorporating an initialization vector (IV) to ensure semantic security across multiple encryptions.

Random Key Permutation: A keyed permutation $f : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ is said to be (t, q, ε)-indistinguishable from a random permutation if, for all probabilistic polynomial-time distinguishers $\mathcal{D}$ running in time t and making q oracle queries to $\mathcal{O}_f$, the advantage of distinguishing f from a truly random permutation is at most $\frac{1}{2} + \varepsilon$. If ε is negligible, then f is considered computationally secure.

Chaos-Based Pseudorandom Generator: A chaotic pseudorandom generator G is used to inject randomness into the compression process. We define G to be a (t, q, b, ε)-identical-random-code-generator if no distinguisher $\mathcal{D}$, running in time t, making q queries, and receiving at most b bits of output, can distinguish G's output from true randomness with probability greater than $\frac{1}{2} + \varepsilon$.

This chaotic generator is initialized using a seed derived from the keyed pseudorandom permutation f_k over the IV. The generator produces super-polynomial-length pseudorandom output, which is then used in both the compression table mutation and the encryption stream.

This construction results in a compression-encryption process where the mutation of the compression table and encryption output is inherently randomized and key-controlled. This enhances both compression diversity and cryptographic strength, particularly for large binary data such as virtual machine images.

Algorithm 2: Secure Compression Algorithm

```
Data: Key $k$, Character Stream $I$
Result: Encrypted Output Stream
InitializeCompression($k$, IV, $G$, prefix, $c$,
  $T$);
// (See Algorithm 2)
while more input exists in $I$ do
    Read next character $c$;
    if prefix $\circ c \in T$ then
        prefix $\leftarrow$ prefix $\circ c$;
    else
        Output index $i$ of prefix from $T$;
        RandomSwap(prefix, $T$);
        RandomInsert(prefix $\circ c$, $T$);
        // (See Algorithm 3)
        prefix $\leftarrow c$;
    end
end
if prefix $\neq \varepsilon$ then
    Output index of prefix from $T$;
end
```

Algorithm 3: Initialize Compression

```
Data: Key $k$, IV, Generator $G$, Variables: prefix, $c$,
      Table $T$
Result: Initialized compression dictionary
Generate fresh IV;
Compute seed $s = f_k(\text{IV})$;
Initialize $G$ with seed $s$;
prefix $\leftarrow \varepsilon$, $c \leftarrow \varepsilon$;
Initialize dictionary table $T$;
foreach single-character string $s$ do
    RandomInsert($s$, $T$);
end
```

Algorithm 4: Random Insert Function

```
Data: Symbol $s$, Dictionary Table $T$
Result: Symbol $s$ inserted into $T$ at random index
Choose random index $r \in T$;
if $T[r] = \varepsilon$ then
    $T[r] \leftarrow s$;
end
else
    Swap $s \leftrightarrow T[r]$;
    RandomInsert($s$, $T$);
end
```

4 Experimental Analysis

This section presents the empirical evaluation of the proposed Combined Big Binaries Compression and Encryption (CBBCE) framework in terms of compression efficiency, computational performance, and cryptographic strength. The experiments were conducted on a Pentium-IV 2.4 GHz machine running Windows 7 with 3 GB RAM. The algorithms were implemented in Java for consistency and cross-platform compatibility.

To provide a comprehensive performance comparison, the proposed CBBCE method was benchmarked against six existing techniques: Huffman Coding [15], LZMA (Lempel–Ziv–Markov chain Algorithm) [1, 2], Advanced Encryption Standard (AES) [16], Chaotic Mutated Adaptive Huffman Tree (CMAHT) [11], and Chaotic Huffman Tree (CHT) [21]. Both Windows and Linux-based virtual machine images were utilized as representative big binary datasets. A fixed 256-bit key "12345678910111121" (in ASCII) was used for encryption during simulation.

A series of standard evaluation criteria were employed to assess the algorithm's effectiveness in compression and its resilience against cryptographic attacks. The results demonstrate that CBBCE is capable of achieving a favorable balance between compression ratio and security, thereby validating its suitability for large-scale data protection.

4.1 Key Space Analysis

Key space size is a fundamental metric in evaluating an algorithm's resistance to brute-force attacks [18]. A cryptographically secure system must ensure a sufficiently large key space to make exhaustive key search computationally infeasible.

Table 1 presents the key sizes of the proposed algorithm in comparison with prior schemes. Notably, traditional compression algorithms such as Huffman and LZMA do not employ key-based encryption and hence do not possess a defined key space. The proposed CBBCE framework utilizes a 256-bit key, offering a key space of 2^{256}, which is considered secure against brute-force attacks under current computational capabilities. By comparison, CMAHT and CHT operate with smaller key sizes (144-bit and 112-bit, respectively), thus providing lower security margins.

Table 1. Key space comparison of compression-encryption algorithms.

Algorithm	Key Size
Huffman Coding	Not Applicable
CMAHT [11]	144 bits
CHT [21]	112 bits
LZMA [1, 2]	Not Applicable
AES [16]	128, 192, 256 bits
CBBCE (Proposed)	**256 bits**

4.2 Compression Ratio

Compression ratio (CR) is a critical metric that quantifies the efficiency of a compression algorithm [19]. It is defined as:

$$\text{CR} = \left(\frac{\text{Length of Encrypted Text}}{\text{Length of Plaintext}} \right) \times 100\% \qquad (4)$$

Table 2 presents the compression ratios obtained using different algorithms on Windows and Linux virtual machine (VM) images. The results demonstrate that the proposed CBBCE method achieves a compression ratio equivalent to LZMA, while also integrating strong cryptographic properties. Notably, CBBCE outperforms traditional compression-encryption pipelines, such as Huffman coding followed by AES, in terms of both compactness and security.

4.3 Algorithm Speed

Tables 3 and 4 report the encryption and decryption times, respectively, for the evaluated algorithms. The results indicate that although the proposed CBBCE approach incurs slightly more time than LZMA due to integrated encryption overhead, it remains significantly faster than hybrid combinations like LZMA+AES, and exhibits favorable performance compared to CMAHT and CHT.

CBBCE achieves encryption times that are competitive, especially given the additional security features. Moreover, its decompression-decryption process is notably efficient, approximately 15% faster than AES and 20–30% faster than CMAHT/CHT.

4.4 Entropy Analysis

Entropy analysis evaluates the randomness in the output stream and serves as an important measure of the strength of cryptographic algorithms. It reflects the uncertainty or unpredictability associated with the data source.

The entropy $H(m)$ of a source m is computed as:

$$H(m) = \sum_{i=0}^{2^N-1} P(m_i) \log_2 \left(\frac{1}{P(m_i)} \right) \qquad (5)$$

where $P(m_i)$ denotes the probability of symbol m_i occurring in the data. The maximum achievable entropy for 8-bit symbols is 8.0. In real-world scenarios, uncompressed data exhibits lower entropy due to redundancy, whereas compressed and encrypted data should ideally approach the maximum value.

Table 5 presents the entropy values observed for each algorithm. The CBBCE approach yields entropy values very close to 8.0, indicating high randomness and strong diffusion/confusion properties. The performance is comparable to that of AES and superior to CMAHT, CHT, and Huffman-based methods.

5 Results and Discussion

The experimental results validate the effectiveness of the proposed Combined Big Binaries Compression and Encryption (CBBCE) framework in addressing the dual challenges of storage efficiency and data confidentiality for large-scale binary objects. The performance has been benchmarked across three critical metrics: compression ratio, processing speed (encryption and decryption time), and entropy-based security analysis.

The CBBCE algorithm achieves a compression ratio comparable to the state-of-the-art LZMA compressor, while integrating cryptographic protection into the data stream. This dual-functionality is particularly advantageous in constrained environments such as cloud backups or virtual machine (VM) provisioning pipelines, where minimizing file size and securing content are equally crucial.

In terms of processing time, CBBCE demonstrates acceptable overhead for encryption and decryption operations. Although the encryption time is slightly higher than LZMA alone, it is justified by the added security layer. Furthermore, the decryption time of CBBCE is significantly lower than most encryption-centric schemes, including AES, CMAHT, and CHT, highlighting its suitability for real-time or latency-sensitive applications.

The entropy analysis reveals that CBBCE produces output streams with entropy values closely approaching the theoretical ideal of 8.0, indicating strong randomness and effective diffusion. This

Table 2. Comparison of compression ratios.

Algorithm	Windows VM (%)	Linux VM (%)
Huffman	71.38	70.22
CMAHT [11]	71.38	70.22
CHT [21]	71.38	70.22
LZMA [1, 2]	27.00	23.66
AES [16]	100.00	100.00
CBBCE (Proposed)	**27.00**	**23.66**

Table 3. Comparison of encryption time (in seconds).

Algorithm	Windows VM	Linux VM
Huffman	59.27	28.25
CMAHT [11]	64.04	29.03
CHT [21]	57.27	24.92
LZMA [1, 2]	192.34	98.52
AES [16]	42.31	30.95
CBBCE (Proposed)	**195.47**	**103.29**

is a critical property for resisting statistical and known-plaintext attacks, further confirming the robustness of the integrated chaotic pseudo-random generator.

Overall, the proposed CBBCE system successfully demonstrates its core design goals: to provide storage-efficient, computationally feasible, and cryptographically secure processing of large binary files. The fusion of LZMA-based compression with chaos-theory-driven encryption offers a balanced trade-off between data size reduction and data protection.

6 Limitations

While the results of this study are promising, the current implementation of CBBCE does present several limitations that warrant further investigation:

- **Processing Overhead:** The encryption phase introduces additional latency due to the initialization and execution of the chaotic key stream generator. While the overhead is moderate, it may not be optimal for ultra low-latency systems or time-critical data transmission scenarios.

- **Platform Dependency:** The implementation has been evaluated on Java-based platforms and Windows/Linux virtual machines. Performance may vary on other architectures or embedded systems with different resource constraints, such as ARM-based processors or mobile IoT devices.

- **Security Model Scope:** The current analysis focuses on brute-force resistance and entropy-based randomness. However, formal cryptographic proofs (e.g., IND-CPA/IND-CCA security under standard assumptions) and side-channel resistance are not addressed in this version. A deeper cryptographic security analysis remains an important direction for future work.

- **Key Management:** The scheme assumes a static 256-bit key and does not explore dynamic key exchange protocols or integration with public key infrastructures (PKI). Secure and scalable key management remains a prerequisite for real-world deployment.

- **Lack of Compression Adaptivity:** Although LZMA performs well on the test datasets, the framework currently lacks adaptivity to the underlying data distribution. Incorporating a learning-based model to select or tune the compression algorithm could potentially improve performance across diverse binary types.

These limitations do not undermine the contributions of the proposed approach but rather provide motivation for future improvements and extensions of the CBBCE framework.

7 Conclusion

In this paper, we proposed a novel framework, Combined Big Binaries Compression and Encryption (CBBCE), to simultaneously address the twin

Table 4. Comparison of decryption time (in seconds).

Algorithm	Windows VM	Linux VM
Huffman	50.78	21.08
CMAHT [11]	47.99	21.21
CHT [21]	50.51	20.59
LZMA [1, 2]	23.98	8.66
AES [16]	55.28	24.31
CBBCE (Proposed)	**28.49**	**11.27**

Table 5. Information entropy comparison.

Algorithm	Windows VM	Linux VM
Huffman	7.79	7.92
CMAHT [11]	7.93	7.96
CHT [21]	7.93	7.96
LZMA [1, 2]	7.96	7.96
AES [16]	7.98	7.98
CBBCE (Proposed)	**7.98**	**7.99**

challenges of data storage optimization and information security, particularly in the context of large-scale binary data such as virtual machine images. The core idea is to embed a stream cipher-like encryption mechanism into a high-performance compression algorithm (LZMA), resulting in a unified process where compression inherently contributes to cryptographic obfuscation.

The proposed system leverages the chaotic logistic map to generate pseudorandom bit streams, which are then XORed with the compressed data, thereby eliminating the need for separate encryption layers and reducing overall processing overhead. Comprehensive experimental evaluations were conducted on metocean and Calgary corpus binary files, demonstrating that CBBCE achieves compression ratios comparable to or better than conventional schemes, while offering robust security properties validated through entropy and key space analyses.

Compared to standalone or sequential compression-encryption methods, CBBCE not only ensures data confidentiality but also achieves improved execution time during decompression-decryption, making it suitable for applications in cloud storage, backup systems, and virtual infrastructure management.

8 Future Directions

While the CBBCE framework shows significant promise, several avenues remain open for further enhancement and research:

- **Formal Cryptographic Proofs:** Future work will focus on providing formal security guarantees such as indistinguishability under chosen-plaintext or chosen-ciphertext attacks (IND-CPA/IND-CCA), as well as integration with modern cryptographic standards.
- **Key Management and Distribution:** Extending the framework to support dynamic key exchange protocols (e.g., Diffie–Hellman) or integration with public key infrastructure (PKI) would facilitate secure deployment in distributed systems.
- **Hardware Acceleration:** Implementing CBBCE using hardware accelerators (e.g., FPGAs or GPUs) may significantly improve runtime performance and energy efficiency, especially for high-throughput applications in cloud or edge environments.
- **Adaptive Compression Selection:** Incorporating machine learning or heuristic models to dynamically choose the most efficient compression strategy based on file type and distribution characteristics could enhance compression efficacy.
- **Cross-Platform Evaluation:** Broadening the performance evaluation to include a wider range of architectures, including mobile, embedded, and IoT platforms, would assess CBBCE's

applicability across diverse use cases.

- **Side-Channel Attack Resistance:** A rigorous analysis of resistance against side-channel attacks (e.g., timing, power, or electromagnetic analysis) is essential to validate its resilience in adversarial environments.

Overall, the integration of compression and encryption as demonstrated in CBBCE opens new opportunities for efficient and secure data processing, and the proposed extensions will further solidify its practicality in real-world applications.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] LZMA SDK (Software development kit). (n.d.). 7-Zip. Retrieved from http://www.7-zip.org/sdk.html

[2] Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory, 23*(3), 337–343. [CrossRef]

[3] Zhang, Y., Xiao, D., Liu, H., & Nan, H. (2014). GLS coding based security solution to JPEG with the structure of aggregated compression and encryption. *Communications in Nonlinear Science and Numerical Simulation, 19*(5), 1366-1374. [CrossRef]

[4] Usama, M., Aziz, A., Alsuhibany, S. A., Hassan, I., & Yuldashev, F. (2025). IDCE: Integrated Data Compression and Encryption for Enhanced Security and Efficiency. *CMES-Computer Modeling in Engineering and Sciences, 143*(1), 1029-1048. [CrossRef]

[5] Wong, K. W., & Yuen, C. H. (2008). Embedding compression in chaos-based cryptography. *IEEE Transactions on Circuits and Systems II: Express Briefs, 55*(11), 1193-1197. [CrossRef]

[6] Wen, J., Kim, H., & Villasenor, J. D. (2006). Binary arithmetic coding with key-based interval splitting. *IEEE Signal Processing Letters, 13*(2), 69-72. [CrossRef]

[7] Wu, C. P., & Kuo, C. C. (2005). Design of integrated multimedia compression and encryption systems. *IEEE Transactions on Multimedia, 7*(5), 828-839. [CrossRef]

[8] Jakimoski, G., & Subbalakshmi, K. P. (2008). Cryptanalysis of some multimedia encryption schemes. *IEEE transactions on multimedia, 10*(3), 330-338. [CrossRef]

[9] Kim, H., Wen, J., & Villasenor, J. D. (2007). Secure arithmetic coding. *IEEE Transactions on Signal processing, 55*(5), 2263-2272. [CrossRef]

[10] Grangetto, M., Magli, E., & Olmo, G. (2006). Multimedia selective encryption by means of randomized arithmetic coding. *IEEE Transactions on Multimedia, 8*(5), 905-917. [CrossRef]

[11] Zhu, Z. L., Tang, Y., Liu, Q., Zhang, W., & Yu, H. (2012, October). A chaos-based joint compression and encryption scheme using mutated adaptive Huffman tree. In *2012 Fifth International Workshop on Chaos-fractals Theories and Applications* (pp. 212-216). IEEE. [CrossRef]

[12] Zhou, J., Au, O. C., & Wong, P. H. W. (2009). Adaptive chosen-ciphertext attack on secure arithmetic coding. *IEEE Transactions on Signal Processing, 57*(5), 1825-1838. [CrossRef]

[13] Zhou, J., & Au, O. C. (2008). Comments on 'A Novel Compression and Encryption Scheme Using Variable Model Arithmetic Coding and Coupled Chaotic System'. *IEEE Transactions on Circuits and Systems I: Regular Papers, 55*(10), 3368–3369. [CrossRef]

[14] Xiang, T., Liao, X., Tang, G., Chen, Y., & Wong, K. W. (2006). A novel block cryptosystem based on iterating a chaotic map. *Physics Letters A, 349*(1-4), 109-115. [CrossRef]

[15] Huffman, D. A. (2007). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE, 40*(9), 1098-1101. [CrossRef]

[16] Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES—the advanced encryption standard*. Springer. [CrossRef]

[17] Lin, Y., Yang, Y., & Li, P. (2025). Development and future of compression-combined digital image encryption: A literature review. *Digital Signal Processing, 158*, 104908. [CrossRef]

[18] Kumar, S., & Sharma, D. (2024). A chaotic based image encryption scheme using elliptic curve cryptography and genetic algorithm. *Artificial Intelligence Review, 57*(4), 87. [CrossRef]

[19] Salahdine, F., Ghribi, E., & Kaabouch, N. (2020, January). Metrics for evaluating the efficiency of compressing sensing techniques. In *2020 International Conference on Information Networking (ICOIN)* (pp. 562-567). IEEE. [CrossRef]

[20] Nagaraj, N., Vaidya, P. G., & Bhat, K. G. (2009). Arithmetic coding as a non-linear dynamical system. *Communications in Nonlinear Science and Numerical Simulation, 14*(4), 1013-1020. [CrossRef]

[21] Hermassi, H., Rhouma, R., & Belghith, S. (2010). Joint compression and encryption using chaotically mutated Huffman trees. *Communications in Nonlinear Science and Numerical Simulation, 15*(10), 2987-2999. [CrossRef]

[22] Baptista, M.S. (1998). Cryptography with chaos. *Physics Letters A, 240*(1), 50-54. [CrossRef]

Muhammad Usama received BS and MS degrees in Software Engineering from Bahria University, Karachi in 2005 and 2009, respectively, and a PhD degree from the University of Technology Petronas, Malaysia in 2018. Currently, he is working as an Assistant Professor at the Department of Cyber Security, Pakistan Navy Engineering College, NUST, Karachi. He has completed his post-doctoral research studies at Qatar University, Doha, Qatar. He also held a research position at the King Saud University, Riyad, Saudi Arabia. His research interests include Big data security, Cloud computing security, Information security and cryptography, and Applied secure computation for machine learning applications. He has around 17 years of experience that spans industry, research, and teaching, and has published articles in various highly reputed journals and conferences. (Email: musama@pnec.nust.edu.pk)

Jawad Ahmad is a highly experienced teacher with more than 13 years of teaching and research experience in prestigious institutes. He has conducted teaching and research at renowned institutions such as Prince Mohammad Bin Fahd University (KSA), Edinburgh Napier University (UK), Glasgow Caledonian University (UK), and Hongik University (South Korea), among others. He has also served as a supervisor for several PhD, MSc, and undergraduate students, providing guidance and support for their dissertations. He has published in renowned journals including IEEE Transactions, ACM Transactions, Elsevier, and Springer with over 200 research papers and 7000 citations (H-Index 50). For the past four years, his name has appeared on the list of the world's top 2% scientists in Cybersecurity, as published by Clarivate (a list endorsed by Stanford University, USA). Furthermore, in 2020, he received the endorsement of a UK exceptional talent candidate ('Emerging Leader') for pioneering work in the field of Cybersecurity and AI. To date, he has secured research and funding grants of more than £250K in the UK and Norway, etc., as a Principal Investigator (PI) and a Co-Investigator (Co-I). In terms of academic achievements, he has earned a Gold medal for his outstanding performance in MS and a Bronze medal for his achievements in BS. (Email: jahmad@pmu.edu.sa)