JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AN INTRODUCTION TO ADVANCED ARITHMETIC CODING BASED ENCRYPTION

[1]Debjit Banerjee, [2]Soham Maity, [3]Aniket Kar, [4]Pritam Adhikari, [5]Sabyasachi Samanta

[1]Student, [2] Student [3] Student [4]Student [5] Professor
[1] Department of Computer Science Engineering (Cyber Security),
[1]Haldia Institute of Technology, Haldia, India, 721657

Abstract : Arithmetic coding is a pivotal technique in data compression, providing efficiency and versatility superior to traditional Huffman coding. This paper delves into advanced arithmetic coding exploring its enhanced algorithms, optimizations, and applications in modern computing. We begin by examining the fundamentals of arithmetic coding, highlighting its core mechanism which encodes entire messages into a single number within the interval [0,1]. This approach allows for incremental updates of probability models, making it particularly suitable for adaptive contexts where symbol probabilities are not known a priori. It progresses to advanced techniques that optimize arithmetic coding's performance. We discuss methods such as adaptive probability estimation, which dynamically adjusts symbol probabilities during encoding and decoding, thus improving compression rates for non-static data sources. Using this method we can encode more number of charters using different groups and sub groups. Furthermore, we explore the use of context-based models, where the probability of each symbol is conditioned on the preceding symbols, leading to more accurate modeling of the data source.

KEYWORDS: Arithmetic Coding, Data Compression, Adaptive Probability Estimation, Context-Based Models, High-Precision Arithmetic, Bitstream Handling

I. INTRODUCTION

Data compression is an essential aspect of modern computing, driven by the need to efficiently store, transmit, and process large volumes of data. As digital content proliferates, ranging from text and images to video and complex scientific data, the demand for effective compression techniques becomes increasingly critical. Compression reduces the amount of data required to represent information, leading to savings in storage costs, bandwidth, and transmission times. Among the myriad compression methods, arithmetic coding stands out for its remarkable efficiency and adaptability. Arithmetic coding, first introduced in the 1970s, revolutionized the field of entropy coding. Unlike traditional Huffman coding, which assigns fixed-length or variable-length codes to individual symbols, arithmetic coding encodes entire messages into a single number within the interval [0,1). This interval representation allows for a more nuanced exploitation of symbol probabilities, leading to higher compression ratios, especially for sources with skewed or complex probability distributions. The core mechanism of arithmetic coding involves recursively partitioning the interval [0,1) based on the probabilities of the input symbols. Each symbol in the input message narrows the current interval according to its probability, effectively homing in on a specific subinterval that uniquely represents the entire message. This process continues until the entire message is encoded into a single floating-point number within the final interval. The decoding process is the inverse, where the interval is successively partitioned to retrieve the original symbols.

While the basic arithmetic coding algorithm is powerful, advanced techniques and optimizations are necessary to harness its full potential in practical applications. Adaptive probability estimation is one such enhancement. Traditional arithmetic coding requires a predefined probability model, which is impractical for dynamic or unknown data sources. Adaptive probability estimation dynamically updates the probabilities as the message is encoded, allowing the model to respond to changing data characteristics. Another significant improvement is the use of context-based models. By conditioning symbol probabilities on preceding symbols, these models capture dependencies within the data, leading to more accurate probability estimations and improved compression performance. Efficient bitstream handling is another crucial aspect, as managing underflow and overflow during encoding is essential for maintaining precision and avoiding errors. Innovations in bitstream handling techniques ensure that arithmetic coding can be efficiently implemented without sacrificing accuracy. Additionally, arithmetic coding involves high-precision arithmetic operations, which can be computationally intensive. Optimizations such as the use of integer arithmetic and precomputed look-up tables for frequent operations significantly reduce computational overhead.

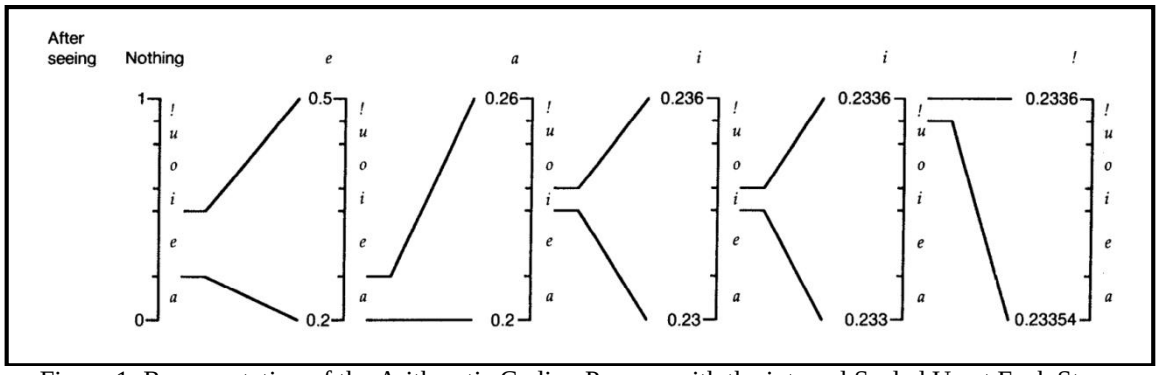

Figure 1: Representation of the Arithmetic Coding Process with the interval Scaled Up at Each Stage

The versatility of arithmetic coding extends its utility beyond conventional data compression. Its applications are diverse and impactful in several key areas. In multimedia codecs, for example, standards such as JPEG2000 and H.264/MPEG-4 AVC leverage arithmetic coding to achieve high compression ratios for images and video. The ability to adapt to varying data characteristics within multimedia content is a critical advantage. In error correction coding, integrating arithmetic coding with error correction schemes enhances the robustness and efficiency of data transmission, particularly in noisy communication channels. Moreover, the combination of arithmetic coding with cryptographic techniques can provide both compression and security, ensuring that compressed data remains confidential and resistant to tampering. This paper is structured to provide a comprehensive understanding of advanced arithmetic coding. It delves into the theoretical foundations of arithmetic coding, providing a detailed explanation of its mechanisms and principles. It then explores advanced techniques and optimizations, discussing adaptive probability estimation, context-based models, efficient bitstream handling, and computational efficiency. Practical implementations are examined, addressing common challenges faced during the deployment of arithmetic coding in real-world applications. The paper also presents various applications of advanced arithmetic coding, highlighting its role in multimedia codecs, error correction coding, and secure communications. Empirical results from a series of experiments and simulations demonstrate the efficiency gains and practical benefits of the proposed techniques. Finally, the paper concludes by summarizing the key findings and suggesting directions for future research.

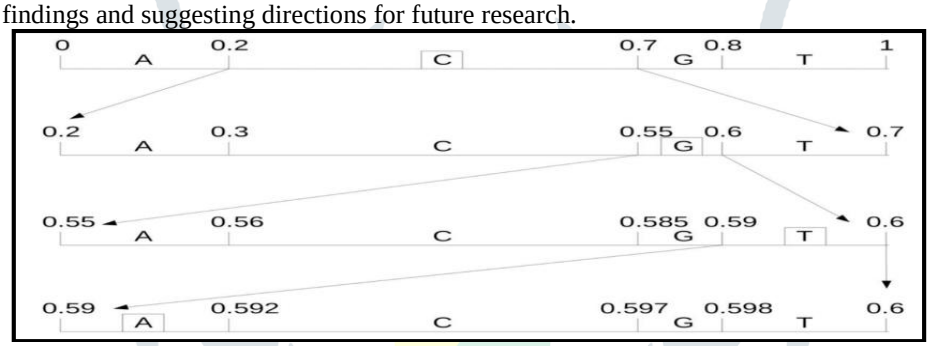

Figure 2: Arithmetic encoding process for single base sequence

The contribution of this paper lies in providing a comprehensive overview of advanced arithmetic coding, showcasing its theoretical advancements, practical implementations, and diverse applications. By addressing both the fundamental principles and cutting-edge techniques, this work contributes to the ongoing development of data compression technologies. The findings presented herein underscore the significance of arithmetic coding in modern data processing and signal the potential for further innovation in this critical field. In summary, advanced arithmetic coding represents a pivotal evolution in data compression. Its ability to adapt to varying data characteristics, coupled with efficient implementation strategies, positions it as a cornerstone technology for the future of data management. This paper seeks to illuminate the depth and breadth of arithmetic coding, providing a valuable resource for researchers, practitioners, and technologists in the field of data compression and beyond.

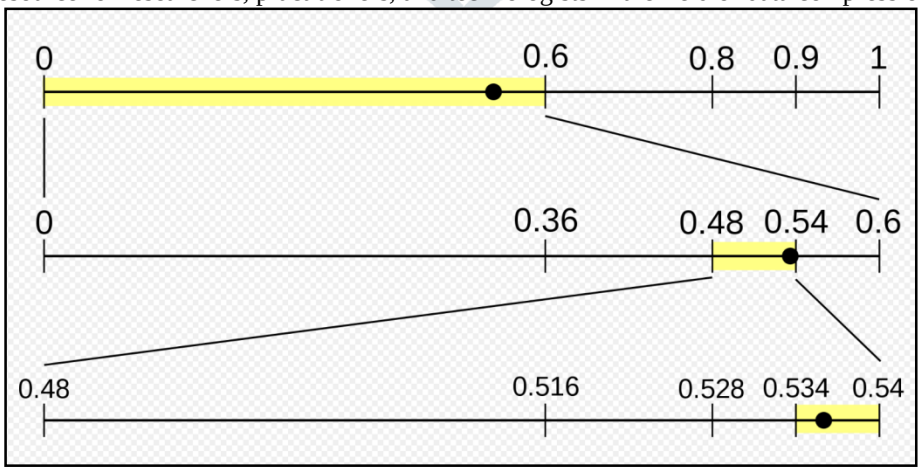

Figure 2.3: Sub region containing the point

A significant portion of the paper is dedicated to the practical implementation challenges and solutions. Efficient bit stream handling is critical, and we present innovations in the management of underflow and overflow during the encoding process. Additionally, we address the computational overhead associated with arithmetic operations in high-precision arithmetic coding and propose techniques to mitigate these costs, such as the utilization of integer arithmetic and the employment of look-up tables

for frequent operations. The applications of advanced arithmetic coding extend beyond conventional data compression. We examine its role in multimedia codecs, particularly in image and video compression standards like JPEG2000 and H.264/MPEG-4 AVC, where its ability to adapt to varying data characteristics significantly enhances performance. Moreover, we explore its application in error correction coding and secure communications, demonstrating how arithmetic coding can be integrated with cryptographic methods to achieve both compression and security.

Through a series of experiments and simulations, we provide empirical evidence of the efficiency gains afforded by advanced arithmetic coding techniques. We compare these methods with traditional coding schemes and illustrate the scenarios in which arithmetic coding excels. The results underscore the practical benefits and potential of advanced arithmetic coding in various domains, paving the way for its broader adoption and further innovation. In conclusion, this paper presents a comprehensive study of advanced arithmetic coding, offering insights into its mechanisms, optimizations, and diverse applications. The findings underscore the significance of arithmetic coding in the future of data compression and signal processing, providing a foundation for ongoing research and development in this critical field.

II. Related Works

G. G. Langdon [1] posited that arithmetic coding surpasses Huffman coding due to its superior method, highlighting its efficiency in compactly representing information, optimal performance without input data blocking, and clear distinction between data representation models and encoding techniques. The paper's primary objective is to counter the prevalent lack of awareness surrounding arithmetic coding by presenting an easily accessible implementation and delineating its performance characteristics. It encompasses essential compression concepts, introduces arithmetic coding using illustrative examples, furnishes separate programs for encoding and decoding, enabling versatile model usage, explores fixed and adaptive models, assesses compression efficiency across varying word lengths, and outlines practical applications. This work addresses a critical knowledge gap by providing a comprehensive understanding and practical implementation of arithmetic coding.

P.G. Howard ET. AL. [2] advocated for arithmetic coding, emphasizing its effectiveness in eliminating data redundancy during encoding. The paper elucidates the operational mechanisms of arithmetic coding and introduces an efficient implementation that employs table lookup as an initial strategy rather than arithmetic operations, demonstrating minimal compression impact despite reduced precision. Furthermore, the abstract explores the possibility of accelerating implementation through parallel processing techniques. It underscores the importance of probability models in furnishing crucial probability information to the arithmetic coder and concludes by providing insights into the comparative advantages and disadvantages of various arithmetic coding techniques.

Ian H. Witten ET. AL. [3] assert the superiority of arithmetic coding over Huffman coding by delineating several advantages: its ability to represent information as compactly as, or more so than, Huffman coding, optimal performance without requiring input data blocking, and its facilitation of a clear demarcation between data representation models and the encoding process. The paper accentuates the ease of accommodating adaptive models and computational efficiency as primary strengths. However, despite these advantages, there remains a widespread lack of awareness about arithmetic coding among authors and practitioners, who often hold the belief that Huffman coding cannot be surpassed. The paper aims to rectify this by presenting an easily accessible implementation of arithmetic coding, delineating its performance characteristics, reviewing fundamental compression concepts, adopting a model-based approach, illustrating arithmetic coding through examples, providing separate programs for encoding and decoding to enable versatile model usage, discussing both fixed and adaptive models, analysing compression efficiency, execution time, and the impact of varying word lengths. Finally, it outlines various suitable applications where arithmetic coding demonstrates its benefits.

Ujala Razaq ET. AL. [4] underscore the significance of arithmetic coding as an entropy encoding method for lossless data compression, acknowledged as a state-of-the-art technique for over four decades. Recognizing extensive research efforts aimed at enhancing its performance, the paper highlights successful experiments and the technique's integration with other methods that have yielded remarkable results. The survey aims to encapsulate achievements in arithmetic coding from 1976 to 2017, providing valuable insights into its evolution and major breakthroughs, serving as a crucial resource for new researchers. It includes a comparative analysis between arithmetic coding and Huffman Coding, showcasing arithmetic coding's superior performance across various scenarios when compared to its contemporaries. This comprehensive study effectively showcases the enduring relevance and effectiveness of arithmetic coding in the field of data compression.

Patel Sanket Mukesh ET. AL. [5] delve into the amalgamation of Arithmetic Coding and the Advanced Encryption Standard (AES) to achieve dual objectives of data security and compression. The study emphasizes the efficacy and adaptability of arithmetic coding while highlighting AES's pivotal role in data security through transformation and permutation techniques. The proposed method involves initial arithmetic encoding of data, succeeded by encryption using AES before transmission, allowing for simultaneous encoding/decoding and compression. Focusing on input sizes of 128 or 256 bits in AES, the paper aims to compress data prior to encryption using digital arithmetic coding. It draws parallels between arithmetic coding and Huffman coding, showcasing their shared ability to reduce the average number of bits necessary to represent symbols, highlighting their similarity in achieving compression goals.

The paper "Introduction to Arithmetic Coding -- Theory and Practice," authored by Zhuohao He ET. AL. [6], provides an extensive exploration of arithmetic coding, a pivotal technique in data compression. It meticulously explains the encoding process, where data symbols are encoded one at a time, each assigned a real-valued number of bits, distinguishing it from other coding methods. This unique approach involves mapping coded messages to real numbers within the interval [0, 1), which enhances the efficiency of data compression. The authors delve into the fundamental properties of arithmetic coding, detailing the concept of code value representation.

The paper "Deep Lossless Compression Algorithm Based on Arithmetic Coding for Power Data," authored by Yue Lu Et. AL. [7], presents an innovative approach to enhancing lossless data compression by integrating deep learning techniques with arithmetic coding. The research specifically targets minute-level power data, which is characterized by its density in the time-frequency domain, posing a significant challenge for traditional compression methods. The authors employ advanced deep learning models, including Bi-directional Long Short-Term Memory (Bi-LSTM) networks and Transformers, to automatically

extract meaningful features from the power data. These models are adept at capturing complex patterns and dependencies within the data, which are crucial for effective compression.

III. WORKING PRNCIPLE

Algorithm – Our implementation of Arithmetic Coding

3.1 Arithmetic Encoding:

Initialize Probability Table:

Create a list `probabilityTable` containing Symbol objects with their respective symbols and probability ranges.

User Input:

Prompt the user to input a string consisting only of characters 'a', 'b', 'c', or 'd'.

Encoding:

Initialize `low` as 0.0 and `high` as 1.0.

For each character `c` in the input string:

Calculate the `range` as the difference between `high` and `low`.

For each symbol `s` in the `probabilityTable`, If the symbol `s` matches the current character `c`:

Update `high` and `low` using the symbol's probability range (`s.high` and `s.low`).

Break the loop.

Calculate the encoded value as ` (low + high) / 2` and round it to the input length using `RoundingMode.HALF_UP`.

3.2Arithmetic Decoding:

Convert Encoded Value:

Convert the encoded value to a double `code`.

Decoding:

Initialize an empty StringBuilder called `output`.

For each character position `i` in the input string:

For each symbol `s` in the `probabilityTable`, If `code` is greater than `s.low` and less than `s.high`:

Append the symbol `s` to the `output`.

Update `code` using the arithmetic decoding formula.

Break the loop.

Display Results:

Print the encoded value and the decoded value of the input string.

Close Scanner:

Close the Scanner object `sc`.

The encoding process uses arithmetic calculations based on probability ranges to encode the input string into a single decimal value. The decoding process reverses this encoding by converting the decimal value back into the original string using the same probability ranges. This algorithm employs a simple arithmetic encoding and decoding technique based on a predefined probability distribution of symbols.

IV. Results

Here are some sample results of the code implemented.

```
Enter your string (Permissible Characters: 'a', 'b', 'c', 'd'): dad
Encoded value: 0.836
Decoded value: dad
Enter your string (Permissible Characters: 'a', 'b', 'c', 'd'): abcd
Encoded value: 0.1024
Decoded value: abcd
Enter your string (Permissible Characters: 'a', 'b', 'c', 'd'): dcba
Encoded value: 0.9296
Decoded value: dcba
Enter your string (Permissible Characters: 'a', 'b', 'c', 'd'): abcddcba
Encoded value: 0.10377472
Decoded value: abcddcba
```

V. CONCLUSION

In conclusion, the exploration of arithmetic coding within this project has illuminated its significance as a powerful and adaptive data compression technique. Its ability to dynamically adapt to varying symbol probabilities, considering the context of entire sequences, has been highlighted as a cornerstone in achieving higher compression rates compared to traditional methods. Throughout this project, we've witnessed arithmetic coding's prowess in efficiently encoding data with non-uniform symbol distributions. By dynamically assigning variable-length codes based on observed probabilities, it optimizes the representation of symbol sequences, leading to significant advancements in compression efficiency.

As evidenced in various real-life applications across image and video compression, text and document handling, communication systems, genomic data, and more, arithmetic coding stands as a versatile solution. Its adaptability, precision, and ability to capture

contextual dependencies among symbols make it an invaluable tool in the realm of data compression. While acknowledging the challenges related to precision, computational complexity, and finite arithmetic constraints, this project underscores the ongoing pursuit of refining arithmetic coding implementations. Addressing these challenges opens doors to more efficient and accurate compression methodologies, paving the way for further advancements in data compression and storage techniques.

Arithmetic coding's role as a key player in the landscape of data compression remains prominent. As technology evolves, optimizing arithmetic coding techniques and integrating them into diverse applications promises continued improvements in compression ratios, transmission efficiency, and storage optimization. In essence, this project serves as a testament to the significance of arithmetic coding, showcasing its adaptability, efficiency, and potential for revolutionizing how we compress and handle data across a multitude of domains.

VI. Future Scope of the Project

Arithmetic coding is a mature compression technique with a rich history and a wide range of applications. However, there are still ongoing research efforts to explore new advancements and expand its capabilities. Here are some areas where future development in arithmetic coding is expected:

Adaptive arithmetic coding: Further refinements to adaptive arithmetic coding algorithms to better handle non-stationary data and improve compression performance across a wider range of data types.

Context-based arithmetic coding: Development of context-based arithmetic coding techniques that incorporate local context information into the probability model, potentially leading to further compression gains.

Hybrid arithmetic coding: Exploration of hybrid compression schemes that combine arithmetic coding with other compression techniques, such as dictionary-based or entropy-coding methods, to achieve even higher compression ratios.

Error-resilient arithmetic coding: Research into error-resilient arithmetic coding techniques that can tolerate transmission errors without compromising data integrity, making it suitable for noisy communication channels.

Arithmetic coding for emerging data types: Investigation of arithmetic coding for compressing emerging data types, such as genomic sequences, sensor data, and multimedia content with complex statistical patterns.

Theoretical advancements: Further theoretical analysis of arithmetic coding to better understand its performance limitations and identify potential improvements in compression efficiency.

As research in these areas progresses, arithmetic coding is expected to remain a valuable tool for data compression in various applications, offering improved compression ratios, reduced computational overhead and enhanced adaptability to diverse data types.

BIBLIOGRAPHY

1. Langdon, G. G. (1984). An Introduction to Arithmetic Coding. In IBM Journal of Research and Development (Vol. 28, Issue 2, pp. 135–149). IBM. https://doi.org/10.1147/rd.282.0135

2. Howard, P. G., & Vitter, J. S. (1994). Arithmetic coding for data compression. In Proceedings of the IEEE (Vol. 82, Issue 6, pp. 857–865). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/5.286189

3. Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. In Communications of the ACM (Vol. 30, Issue 6, pp. 520–540). Association for Computing Machinery (ACM). https://doi.org/10.1145/214762.214771

4. Ujala Razaq, Xu Lizhong, Changli Li, & Muhammad Usman. (2020). Evolution and Advancement of Arithmetic Coding over Four Decades. In Open Journal of Science and Technology (Vol. 3, Issue 3, pp. 194–236). Readers Insight Publisher. https://doi.org/10.31580/ojst.v3i3.1664

5. Mukesh, P. S., Pandya, M. S., & Pathak, S. (2013). Enhancing AES algorithm with arithmetic coding. In 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE). 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE). IEEE. https://doi.org/10.1109/icgce.2013.6823404

6. Moffat, A., Neal, R. M., & Witten, I. H. (1998). Arithmetic coding revisited. In ACM Transactions on Information Systems (Vol. 16, Issue 3, pp. 256–294). Association for Computing Machinery (ACM). https://doi.org/10.1145/290159.290162

7. Boyd, C., Cleary, J. G., Irvine, S. A., Rinsma-Melchert, I., & Witten, I. H. (1997). Integrating error detection into arithmetic coding. In IEEE Transactions on Communications (Vol. 45, Issue 1, pp. 1–3). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/26.554275

8. Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. In Communications of the ACM (Vol. 30, Issue 6, pp. 520–540). Association for Computing Machinery (ACM). https://doi.org/10.1145/214762.214771

9. Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. In Communications of the ACM (Vol. 30, Issue 6, pp. 520–540). Association for Computing Machinery (ACM). https://doi.org/10.1145/214762.214771

10. Howard, P. G., & Vitter, J. S. (1992). Analysis of arithmetic coding for data compression. In Information Processing & Management (Vol. 28, Issue 6, pp. 749–763). Elsevier BV. https://doi.org/10.1016/0306-4573(92)90066-9

11. Elmasry, G. F. (1999). Embedding channel coding in arithmetic coding. In IEE Proceedings - Communications (Vol. 146, Issue 2, p. 73). Institution of Engineering and Technology (IET). https://doi.org/10.1049/ip-com:19990105

12. Singh, A., & Gilhotra, R. (2011). Data Security Using Private Key Encryption System Based on Arithmetic Coding. In International Journal of Network Security & Its Applications (Vol. 3, Issue 3, pp. 58–67). Academy and Industry Research Collaboration Center (AIRCC). https://doi.org/10.5121/ijnsa.2011.3305

13. Raita, T., & Teuhola, J. (1994). Arithmetic coding into fixed-length codewords. In IEEE Transactions on Information Theory (Vol. 40, Issue 1, pp. 219–223). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/18.272486

14. Mi, B., Liao, X., & Chen, Y. (2008). A novel chaotic encryption scheme based on arithmetic coding. In Chaos, Solitons & Fractals (Vol. 38, Issue 5, pp. 1523–1531). Elsevier BV. https://doi.org/10.1016/j.chaos.2007.01.133

15. Ishibashi, H., & Tanaka, K. (2001). <title>Data encryption scheme with extended arithmetic coding</title> In M. S. Schmalz (Ed.), Mathematics of Data/Image Coding, Compression, and Encryption IV, with Applications. SPIE. https://doi.org/10.1117/12.449585