Journal of Informatics and Mathematical Sciences
Vol. 8, No. 4, pp. 273–279, 2016
ISSN 0975-5748 (online); 0974-875X (print)
Published by RGN Publications

http://www.rgnpublications.com

Special Issue
CRIMSTIC 2016
Current Research in Information Technology, Mathematical Sciences, Science and Technology International Conference 2016
April 13-14, 2016, Melaka, Malaysia

Range GLS-Coding: A Scheme of Simultaneous Data Compression and Encryption

Research Article

Muhammad Usama* and Nordin Zakaria

HPC Service Center, Universiti Teknologi PETRONAS, Seri Iskander, 31750 Tronoh, Perak, Malaysia
***Corresponding author:** usama.khanzada@hotmail.com

Abstract. This paper presents an improved version of GLS-coding for simultaneous data compression and encryption called Range GLS-coding. It removes implementation issue of infinite precision real numbers due to long products of real numbers in GLS-Coding. This improvement increases the algorithm efficiency to almost double. Furthermore, encryption quality is improved by masking pseudorandom keystream; the secret key is incorporated by changing the direction and performing cyclic-shift operation in a complex piecewise linear chaotic map (PWLCM). Experimental results demonstrate that proposed algorithm has acceptable security and compression features. The overall algorithm provides simultaneous data encryption and compression and useful for real-time applications.

Keywords. GLS-Coding; Compression; Encryption; Simultaneous compression and encryption

MSC. 68P25; 68P30

Received: December 29, 2015 **Accepted:** August 17, 2016

1. Introduction

Since 1990s, Since the 1990s, the chaos theory of non-linear dynamical systems has been applied in the development of low complexity cryptosystems. The intrinsic properties of non-linear dynamical systems such as ergodicity, mixing, sensitivity to the initial condition, a control parameter, random behavior etc. [1] are useful for practical implementation of cryptosystems. These properties are directly linked to confusion and diffusion properties of ideal cryptosystems [2]. However, performing compression and encryption using non-linear dynamical systems is a relatively new area of research, where the goal is to achieve compression and encryption simultaneously and avoid the separate operation of compression or encryption. In recent years, some approaches [3–5] based on Huffman coding for simultaneous compression and encryption have been suggested. The process of embedding encryption features mainly focused on the control of tree branches swapping via a key. In [3], an integrated compression and encryption approach was presented that swaps Huffman tree branches left and right using the controlled key. Later, an improvement was suggested using chaotically mutated Huffman trees [4] that overcome the multiple codeword issue to enlarge the key space and solve the weak security issues in [3]. The work in [4] was further improved in [5]; two chaotic functions were adopted to avoid known-plaintext attacks. In another related work, a randomized arithmetic coding RAC algorithm [6]was proposed for JPEG 2000 standard, which inserts randomization into the standard arithmetic coding operation for encryption. However, these algorithms still have serious security issues [7] and they did not investigate the relationship between the chaos, encryption, and compression. A chaotic system is only used as a pseudorandom bitstream generator that incorporate key control.

2. GLS Coding

The relationship between source coding and chaos was studied in [8,9] for simultaneous data compression and encryption. In this, a non-linear chaotic dynamical system referred to as Generalized Luroth Series (GLS) was proposed and demonstrated to be Shannon optimal [9]. In [10], the concept of GLS-Coding were implemented for simultaneous arithmetic coding and encryption using Tent map. The Tent map is defined as [11]:

$$f(x) = \begin{cases} x/p & x \in [0,p] \\ (1-x)/(1-p) & x \in [p,1] \end{cases} \tag{2.1}$$

where p is the symbol probability and $p \in (0,1)$. The compression is attained using reverse function of the tent map [8,9] is given by:

$$I\{i\} = \begin{cases} p(I\{i-1\}) & \text{if } 0 \\ 1-p(I\{i-1\}) & \text{if } 1 \end{cases} \tag{2.2}$$

where p is the symbol probability and $I\{i\}$ is the interval for ith symbol. Initially, $I\{0\} = [0,1]$. The binary sequence encoding is performed using equation (2.2), iteratively. At the end of encoding process the final interval is obtained [START, END]. Any value between final interval e.g. (START+END)/2 can be converted into binary as a compressed sequence. In decoding, the decoder requires this value to iterate equation (2.1).

The above explanation looks simple and easy to understand. But it has an implementation issue due to infinite precision real numbers that result from long products of real numbers. One solution is to perform approximation computation and express the numbers in mantissa and exponent [12]; this, however, leads to carry-over and correct positioning issues. Another solution is to use range coding i.e. bring all fractions to the common denominator. This simple multiplication increases the algorithm efficiency. We adopt this second solution, revising the GLS coding as Range-GLS coding.

3. Proposed Word

1. *Range GLS-Coding*: The GLS-coding is turned into range GLS-coding by increasing range from $[0,1]$ to $[0,r]$, where r is the range i.e. use to remove fractions. The Tent map is now defined as follows:

$$f(x) = \begin{cases} (x/rp)r & x \in [0, rp] \\ (r - x/rp - rp)r & x \in [rp, r] \end{cases} \tag{3.1}$$

where p is the symbol probability and r is the range. Let $r = b^n$ where b is base and n is the largest number of base b that algorithm can conveniently handle. The compression is performed using the reverse interval mapping method as:

$$I\{i\} = \begin{cases} p(I\{i-1\}) & \text{if } 0 \\ r - p(I\{i-1\}) & \text{if } 1 \end{cases} \tag{3.2}$$

where $I\{i\}$ is the obtained interval and p is the probability after performing the ith symbol and at first, $I\{0\} = [0,r]$. The binary sequence encoding is performed using equation (3.2), iteratively. At the end of an encoding process, the final interval is obtained [START, END]. Any value between final interval e.g. (START+END)/2 can be converted into binary as a compressed sequence. In decoding, the decoder requires iterating equation (3.1) from this value to recover the binary sequence, correctly.

2. *Chaotic pseudorandom keystream generator*: The Tent map is selected as the underlying chaotic map for pseudorandom keystream generator. The Tent map is an ideal chaotic map which satisfies uniform distribution and invariant density. In [10], it was pointed out that pseudorandom keystream generator based-on tent map has strong cryptographic properties such as initial value sensitivity, random behavior, unpredictability etc. The iterative tent

map is defined as:

$$x_{n+1} = F_\lambda(x_n) = \begin{cases} \frac{x_n}{\lambda} & \text{if } x_n \le \lambda \\ \frac{1-x_n}{1-\lambda} & \text{if } x_n \ge \lambda \end{cases} \tag{3.3}$$

where $\lambda \in (0.25, 0.75)$ is a control parameter, $x_n \in (0,1)$ and x_0 represents the initial value (at $n = 0$). The proposed algorithm requires three initial values of the chaotic map for generating three pseudorandom keystreams as explained earlier. If the precision is 10^{14} for initial value x_0 and control parameter λ, then key space is approximately around 10^{84}. The key space is large enough to resist against brute-force attacks.

4. Experiment Analysis

The analysis parameters such as compression ratio, performance and security of the proposed algorithm are analyzed on a Pentium-IV 2.4 MHz PC, with Windows 7 and 3 GB RAM. For comparison purpose, GLS coding and proposed algorithm are implemented using Java language. The selected files of standard Calgary corpus are used in experiments. In the following subsections, the results of some experiments are given to prove the needed capabilities of compression, performance and security of the proposed algorithm.

1. *Compression Ratio*: Compression ratio indicates the capability of the algorithm. To evaluate the compression capabilities of the proposed algorithm with GLS Coding, the experiment is performed on selected files of standard Calgary Corpus as shown in Table 1. The compression ratio is calculated using equation (4.1).

$$\text{Ratio} = \frac{\text{ciphertext length}}{\text{plaintext length}} \times 100\%. \tag{4.1}$$

The computed average compression ratio of proposed algorithm is 84.66 and GLS-Coding is 84.61. This shows that improve version does not affect the compression capabilities, however, the slight difference in compression ratio is negligible.

Table 1. Comparison of Compression ratio

File	Size KB	Proposed Algorithm	GLS Coding
book1	750.75	82.47	81.64
paper2	80.27	83.14	83.71
paper3	45.44	83.93	85.36
progl	69.97	86.34	85.61
progp	48.22	87.43	86.73
Avg. compression ratio		84.66	84.61

2. *Key Sensitivity and Plaintext Sensitivity*: The cryptographic algorithm should be highly sensitive to key and plaintext, so the slight change in key or plaintext should reflect in output. To evaluate the key sensitivity of proposed algorithm, the keys k_1, k_2 and k_3 are arbitrarily changed, and the two ciphertext files generated from the same plaintext are compared bit-by-bit. The comparison results are given in Table 2 of key sensitivity. Similarly, the plaintext sensitivity is evaluated by randomly toggling one bit and perform encryption with same keys. Finally, the ciphertext files are compared bit-by-bit as given in Table 2. The bit-change-percentage of key and plaintext sensitivity are very close to the ideal value (50%). It proves that proposed algorithm is highly sensitive to key and plaintext.

Table 2. Key and plain text sensitivity analysis

	Key sensitivity analysis				Plaintext sensitivity analysis			
Files	beginning	middle	end	complete	beginning	middle	end	complete
book1	49.413	49.413	49.412	49.413	49.03	51.19	49.02	49.013
paper2	49.418	49.412	49.411	49.414	49.056	50.487	49.004	51.149
paper3	49.414	49.412	49.412	49.413	51.101	51.116	51.217	51.128
progl	49.415	49.414	49.411	49.413	49.139	50.922	50.928	50.996
progp	49.574	49.412	49.41	49.415	50.165	51.208	51.184	50.986

3. *Algorithm Speed*: Apart from the compression and security analysis, some other important issues are needed to be considered. This includes the algorithm speed. Table 3 summarizes the comparison of encryption and decryption time. The average computed time is compared which clearly indicates that proposed algorithm has an edge in performance and it takes less time to perform encryption and decryption

Table 3. Summarizes comparison of compression-encryption time(s)

	Encryption time		Decryption time	
Files	Proposed Algorithm	GLS Coding	Proposed Algorithm	GLS Coding
book1	575	734	670	1496
paper2	40	62	39	115
paper3	25	27	30	58
progl	37	66	63	92
progp	20	25	18	54
Avg. Time	139.4	182.8	164	363

5. Conclusion

This paper proposed an improved version of GLS-coding for simultaneous data compression and encryption called Range GLS-coding. The improved version solved the implementation issue of infinite precision real numbers by removing fractional part generated due to long products of real numbers in GLS-Coding. This improvement also increases the algorithm speed efficiency to almost double. The proposed algorithm is successfully implemented and critically tested for selected Calgary corpus files. The ciphertexts produced by the algorithm are highly key and plaintext sensitive. Experiment results show that compression ratio is acceptable with negligible compression penalty relative to GLS-Coding.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

References

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

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

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

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

[5] Y. U. Hai, A chaos-based joint compression and encryption scheme using mutated adaptive Huffman Tree, in *Fifth International Workshop on Chaos-fractals Theories and Applications* (IWCFTA) (2012).

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

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

[8] M. B. Luca, A. Serbanescu, S. Azou and G. Burel, A new compression method using a chaotic symbolic approach, in *IEEE Commun. Conf.* (2004).

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

[10] K.-W. Wong, Q. Lin and J. Chen, Simultaneous arithmetic coding and encryption using chaotic maps, *IEEE Transactions on Circuits and Systems II: Express Briefs* **57** (2) (2010), 146–150.

[11] G. Alvarez and S. Li., Some basic cryptographic requirements for chaos-based cryptosystems, *International Journal of Bifurcation and Chaos* **16** (8) (2006), 2129–2151.

[12] R. C. Pasco, *Source coding algorithms for fast data compression, Doctoral dissertation, Stanford University* (1976).