Research Article

Chaos Based Joint Compression and Encryption Framework for End-to-End Communication Systems

Nidhi Goel,¹ Balasubramanian Raman,² and Indra Gupta³

¹ Department of Electronics & Communication Engineering, Delhi Technological University, New Delhi 110042, India

² Department of Computer Science Engineering, Indian Institute of Technology, Roorkee 247667, India

³ Department of Electrical Engineering, Indian Institute of Technology, Roorkee 247667, India

Correspondence should be addressed to Nidhi Goel; nidhi.iitr@gmail.com

Received 12 September 2014; Revised 17 October 2014; Accepted 17 October 2014; Published 9 November 2014

Academic Editor: Athanasios V. Vasilakos

Copyright © 2014 Nidhi Goel et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Augmentation in communication and coding technology has made encryption an integral part of secure multimedia communication systems. Security solution for end-to-end image transmission requires content adaptation at intermediate nodes, which consumes significant resources to decrypt, process, and reencrypt the secured data. To save the computational resources, this paper proposes a network-friendly encryption technique, which can be implemented in transparency to content adaptation techniques. The proposed encryption technique maintains the compression efficiency of underlying entropy coder, and enables the processing of encrypted data. Thorough analysis of the technique, as regards various standard evaluation parameters and attack scenarios, demonstrates its ability to withstand known-plaintext, ciphertext-only, and approximation attacks. This justifies its implementation for secure image transmission for end-to-end communication systems.

1. Introduction

The last decade has witnessed significant advancements in communication and networking technologies, paving the way for various distributed network multimedia applications and services. Wider availability of these multimedia services requires data dissemination over open-natured wired or wireless IP networks, which makes the multimedia content vulnerable to eavesdropping. Consequently, preserving data confidentiality has become an important issue for secure transmission and distribution through wired/wireless networks.

Considering the resource constrained, bandwidth limited nature of these communication channels, encryption is performed during compression to achieve better encryption efficiency with less computational resources [1, 2]. Such joint compression and encryption framework is achieved either by embedding key-controlled confusion and diffusion in these source-coding schemes or by incorporating compression in cryptographic algorithms [3–7].

Wen et al. [4] proposed the mapping of variable length codewords to fixed length indices, which are then encrypted using a conventional cipher. Spatial shuffling of macro block, coded 8×8 block, or run level codeword has also been implemented in compressed domain [4]. Its security aspect has been improved by Kankanhalli and Guan [5], by the last bit flipping of randomly chosen Huffman codeword. Wu and Kuo [8] proposed multiple Huffman table (MHT) based encryption, where each symbol is encoded with a different Huffman tree. These multiple trees have been generated by the mutation of four basic trees. However, this requires high computational resources for synchronizing the encoder-decoder pair. Also, the use of weak keys to select multiple Huffman trees makes the encrypted content vulnerable to basic cryptanalytic attacks [9, 10].

Though encryption is performed during compression, these techniques have an intrinsic security gap, especially, when the data is targeted for devices connected in a heterogeneous network. Such a network comprises devices connected through different access networks having different

bandwidth constraints and channel conditions. Further, these devices may have different computation capabilities, screen resolution, and data format requirements. For instance, same data may be required by a high speed workstation connected through a high quality link, and also by a PDA with limited computational capability and low screen resolution, connected through a wireless link. It is not feasible for a server to anticipate the user requirements and save multiple copies of the same data in different formats. An elegant solution to this problem is the implementation of adaptation procedures at the intermediate nodes. The adaptation procedures at the intermediate nodes would cater to varying requirements of different client devices without requiring any change in the existing client or server configuration.

However, most of the encryption processes disturb the data format, making it unsuitable for such processing operations. Consequently, this requires the intermediate nodes to first decrypt, then process, and reencrypt the data using the same or new keys. This generates significant processing overhead and also necessitates the intermediate modules to possess encryption/decryption capabilities. It also imposes an additional burden on the key management system for synchronizing the existing keys or generating the new security keys. Further, a high level of trust is required between the end-points and the adaptation points, for sharing the security keys. Possession of security keys with all but one trusted node can break the security system, despite the strongest encryption technique implemented.

To alleviate this problem, encryption should be performed in a manner such that content adaptation can be accomplished with transparency to the content protection mechanism. Adaptation of encrypted content has been made possible in [4, 11–13], but these techniques adversely affect the compression efficiency of the entropy coder. Technique proposed by Wu and Kuo [8] maintains the compression efficiency, but the existence of invalid Huffman codewords in encrypted bitstream makes it infeasible to perform content adaptation without decryption. Validity of Huffman dictionary has been maintained by Kankanhalli and Guan [5], but it consumes extra computational resources.

This necessitates a network-friendly encryption technique to achieve the twin objectives, that is, high compression efficiency and content adaptation of encrypted data, without increasing the computational resource consumption. Thus, a unique joint compression and encryption framework has been proposed to generate format-compliant encrypted bitstream, so that significant processing overhead can be saved at the intermediate nodes. The proposed technique employs chaos based logistic map to alter the Huffman dictionary at the entropy coding stage, without any change in the compressibility of the encoder. This would allow content adaptation of the encrypted data and, hence, would not compromise on security by sharing the security keys with intermediate nodes.

This paper is organized as follows. Section 2 highlights the intricacies involved in Huffman coding from the viewpoint of the proposed image encryption technique. It also presents a snapshot of the one dimensional logistic map, which has been employed as a pseudorandom number generator. Section 3

describes the proposed technique that overcomes the limitations of existing dictionary-scrambling based encryption techniques. The performance, security, and comparative analysis have been discussed in Section 4. Finally, Section 5 concludes the paper.

2. Preliminaries

The use of Huffman encoding for compression and chaos based maps for cryptography has already been established [14–16]. This section gives a brief description of the Huffman encoding and the logistic map. Huffman encoding has been employed at the entropy coding stage to achieve compression, whereas the logistic map has been used as a pseudorandom number generator to scramble the generated dictionary.

2.1. Huffman Encoding Framework. Huffman encoding is a lossless entropy coding scheme, which requires *a priori* statistical input data generated from a discrete-time, discrete-amplitude information source, with a finite set of unique symbols, $S = \{s_0, s_1, \dots, s_{n-1}\}$. At any given time, the probability that the source outputs symbol s_k is $p_k = P(s_k)$, where $0 \leq k \leq n-1$, with $\sum_{k=0}^{n-1} p_k = 1$. In Huffman coding, for every distinct symbol $s_i \in S$, a unique binary variable length codeword $c_i \in C$ is generated, where $C = \{c_0, c_1, \dots, c_{n-1}\}$ denotes the set of Huffman codewords. The length of these generated codewords is inversely proportional to the occurrence probability of their corresponding symbol in the input data set.

The symbols and their corresponding codewords are mapped to each other using a mapping function M . This function maps the source symbol $s \in S$ to the codeword $c \in C$, that is, $M : s \rightarrow c$. Let M^{-1} be the inverse of M ; then the encoding and decoding can be shown as $c = M(s)$ and $s = M^{-1}(c)$, respectively.

The symbols with their corresponding codewords are usually stored in a symbol-to-codeword mapping table, available to the encoder as well as the decoder. Huffman codes are uniquely decodable due to the imposed prefix condition on the distinct entries of symbol-to-codeword mapping table [17]. The optimality condition for this can be stated as follows.

- (1) No codeword in the dictionary can be a prefix of any other codeword.
- (2) If $P(s_k) > P(s_j)$ ($k \neq j$), then $l_k < l_j$, where l_k and l_j are the lengths of the codewords assigned to symbols s_k and s_j , respectively.
- (3) If the symbols are listed in the order of their decreasing probabilities, the last two symbols in the ordered list are assigned codewords of the same length. These two codewords are alike, except for their final bit.

The formed symbol-to-codeword mapping table is also referred to as a dictionary in the current context. The validity of encrypted bitstream and Huffman dictionary has to be maintained to support adaptation on encrypted content. This can be attained if and only if the prefix condition of codewords is not violated in the modified dictionary.

FIGURE 1: Block diagram for the proposed technique.

2.2. Logistic Map. Single dimension logistic map can be mathematically represented as

$$x_{n+1} = rx_n(1 - x_n), \quad (1)$$

where $0 \leq x_n \leq 1$.

The parameters r and x_0 together form the initial conditions of the logistic map. This map exhibits high sensitivity to initial conditions for r varying between 3.57 and 4. Due to its high sensitivity, chaotic behaviour, and nonuniform probability density function, this map has been found suitable as a pseudorandom sequence generator [18].

Thus, the proposed work employs this map in similar context. The initial conditions of the map are considered as a part of the security keys. However, while selecting the initial conditions, considerations have been made to comply with the key related requirements of chaos based cryptosystems [19]. Hence, the nonchaotic region existing between 3.82 and 3.845 has been carefully avoided to ensure a random sequence.

3. Proposed Technique

A block diagram indicating the placement of proposed encryption function in a communication food-chain is shown in Figure 1. In the proposed encryption framework, an input image has been transformed into the frequency domain, where the quantized transform coefficients form the discrete-time, discrete-amplitude information source. After calculating the source statistics of this finite set of unique symbols, Huffman dictionary has been generated. This Huffman dictionary is then modified using the chaotic output obtained from logistic map. The modification is performed in terms of both symbol values as well as their corresponding codewords, keeping in mind the requirements of a valid Huffman dictionary. The modification of Huffman dictionary can be described as follows.

A Huffman dictionary consists of distinct symbols $s_i \in S$ which corresponds to variable length codewords. The length of codeword varies from $l_{\min}$ to $l_{\max}$, where $l_{\min}$ and $l_{\max}$ denote the shortest and longest existing code-length, respectively. In the proposed technique, different disjoint sets are formed from the set S as per the length of codes such that

$$\begin{aligned} D_1 &= \{x \mid x \in S, \text{code-length}(x) = l_{\min}\} \\ D_2 &= \{x \mid x \in S, \text{code-length}(x) = l_{\min} + 1\} \\ &\vdots \\ D_{l_{\max}-l_{\min}+1} &= \{x \mid x \in S, \text{code-length}(x) = l_{\max}\}, \\ S &= \{D_1 \cup D_2 \cup \dots \cup D_{l_{\max}-l_{\min}+1}\}. \end{aligned} \quad (2)$$

Logistic map is then iterated to generate a pseudorandom sequence of the size of a disjoint set D_i , where i varies from 1 to $l_{\max} - l_{\min} + 1$. The symbols within the formed disjoint subsets are then scrambled using the generated pseudorandom sequence. Different initial conditions of the map have been assumed for different subsets. This yields a different mapping function between symbols and the codewords for each subset.

Apart from changing the mapping function, the symbol values have also been simultaneously modified on the basis of symbol length. It is observed that Huffman symbols are primarily t -digit decimal numbers, where t generally varies from 1 to 4. These symbols are altered according to the following rules.

- (1) All one-digit and two-digit symbols of the dictionary are modified by making LSB as MSB and MSB as LSB. For example, "5" is read as "05," thus giving a modified value of "50," "19" is converted to "91," and "27" is converted to "72."
- (2) All three-digit and four-digit symbols have been scrambled according to distinct random shuffling

TABLE 1: Dictionary scrambling process.

Position number	Original subset		Scrambling vector	Scrambled subset	
	Symbol	Code		Symbol	Code
1	210	000111010111	20	27	000111010111
2	168	010000111000	19	96	010000111000
3	156	010000111011	12	67	010000111011
4	147	000010101111	1	102	000010101111
5	75	000010101110	8	522	000010101110
6	82	000111010110	3	561	000111010110
7	140	010000111010	10	271	010000111010
8	252	001101101011	9	261	001101101011
9	126	000110010111	5	57	000110010111
10	127	010111010001	17	88	010111010001
11	158	000110010000	14	37	000110010000
12	76	000110010110	16	58	000110010110
13	189	010111101000	2	681	010111101000
14	73	001101101010	6	28	001101101010
15	83	000011010001	11	581	000011010001
16	85	010111001010	4	471	010111001010
17	88	000110011101	7	401	000110011101
18	61	000001111100	15	38	000001111100
19	69	000110010000	13	891	000110010000
20	72	001101100010	18	16	001101100010

Reverse number for 1- and 2-digit symbols and shuffling vector "213" for 3-digit symbol.

vectors. For example, if LSB denotes position 1 and the shuffling vector for three-digit symbols is "213," then the symbol "138" is converted to "381" and "672" is converted to "726." Likewise, if shuffling vector is "2413" for four-digit symbols, then the symbol "8725" is changed to "2857" and "3672" is modified to "7326."

In this manner, the proposed encryption technique simultaneously changes the symbol values as well as their corresponding codewords. The original subsets are then replaced with their scrambled counterparts to generate an altered Huffman dictionary.

The technique uses distinct scrambling vectors for different subsets to change the mapping function and same shuffling vector for same length symbols to modify their values. This former preserves the codestream length before and after encryption, while the later ensures distinct symbols even after modification. A snapshot of a subset from the original and the modified Huffman dictionary has been illustrated in Table 1.

4. Performance and Security Analysis

This section assesses the performance and security of the proposed technique, as regards different evaluation parameters and attack scenarios. Simulations for the proposed technique have been performed on various test images; however, results for only three images ("huts," "goldhill," and "car") are shown here.

4.1. Visual Degradation and Peak Signal to Noise Ratio. Decoding of images using scrambled dictionary provides the encrypted image. Subjective analysis of these encrypted images has been performed to observe the amount of information leakage. The encrypted output for few test images is shown in Figure 2. It is observed that the encrypted output is completely incomprehensible and does not give any information about the original image.

To objectively verify the degradation introduced by the proposed technique, peak signal to noise ratio (PSNR) has also been evaluated, using

$$\text{PSNR} = 20 \times \log_{10} \left(\frac{255}{\sqrt{\text{MSE}}} \right) \text{dB}, \quad (3)$$

where MSE is mean square error and is given by

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \|I(i, j) - I'(i, j)\|^2, \quad (4)$$

where $I(i, j)$ and $I'(i, j)$ denote intensity of the original image and the encrypted image at pixel position (i, j) .

PSNR obtained for different images using the proposed technique has been indicated in Table 2. It can be observed that a higher PSNR value is achieved by "huts" image, in comparison to other images. This is due to the fact that "huts" is a low contrast image and generates a small dictionary with relatively small sized subsets. This offers a small scrambling

FIGURE 2: Encrypted results for huts, goldhill, and car image; top row: original images, bottom row: their encrypted counterparts.

TABLE 2: PSNR obtained for various images.

Image	PSNR (dB)
Lena	0.2172
Goldhill	0.9118
Baboon	0.1864
Car	0.2581
Huts	2.3413
Butterfly	1.3684
Woman	0.1987

space, as opposed to a large scrambling space for high contrast images. Thus, shuffling in small subsets causes high PSNR value, indicating that the performance of proposed technique strongly depends on contrast of the test image. Though relatively higher PSNR has been obtained for low contrast images, it is still within the satisfactory limits.

4.2. Key Sensitivity. As per Kerckhoff's principle, security keys are the most important part of any cryptosystem. Decryption using an incorrect key or an approximately correct key should not reveal any details of the original image [19]. To verify the key sensitivity of the proposed technique, three different cases, with slight modifications in decryption

key, have been considered. First case considers wrong initial conditions of the logistic map for only one subset. This generates a wrong scrambling vector, which consequently leads to wrong mapping function for that particular set of symbols. Thus, all but one subset of the entire dictionary is correctly reshuffled. The obtained reshuffled dictionary (with one subset wrongly reshuffled) has been used to decode the output; the results for which are shown in Figure 3 (1st row). This illustrates that an incomprehensible mapping function for even one set of symbols does not give a clear view of the original image.

The second case considers an incorrect reshuffling vector during retrieval of values, for either 3-digit or 4-digit symbols. Third case considers both incorrect initial conditions and wrong reshuffling vector. Second case leads to wrong symbol values, whereas the third case leads to an incorrect descrambled dictionary, from the perspective of symbol values as well as their corresponding codewords. The decoded images for these two cases are illustrated in Figure 3 (2nd and 3rd row), whereas Figure 3 (last row) illustrates the decoded image when all the key parameters are correct. The results clearly indicate that an incorrectly descrambled dictionary leads to an incomprehensible image. It can be observed that no information has been leaked out even with decryption using a slightly different key. This depicts high key sensitivity of the proposed technique.

FIGURE 3: Key sensitivity results for huts, goldhill, and car image.

4.3. Compression Overhead. The proposed technique is also assessed for its impact on compression efficiency of the encoder. This can be quantified by the change in average code-length before and after encrypting the symbol source. Since symbol shuffling has been performed between symbols of same probability distribution, the length of codestream before and after encryption remains unaltered. Also, modification of the symbol values does not affect the length of their corresponding codewords. This has been verified for all the test images; however results for only “Huts” image have been depicted graphically in Figure 4. This clearly reflects that dictionary scrambling does not affect the compression efficiency of the encoder.

The impact on compression efficiency of the encoder is further evaluated by measuring the overhead generated in the entire process. The expression for overhead calculation in dictionary scrambling based encryption techniques can be given as [13]

$$\delta L = (q_i - p_i) l_i, \quad (5)$$

where δL indicates the change in code length and p_i and q_i stand for probability mass function of the symbols, before and after encryption process. Code-length for distribution p_i is given by l_i . As the probability mass function of the symbols remains the same before and after encryption, no change in the bitstream length has been observed. This also ascertains the avoidance of any adverse effect on the compression efficiency, due to the implementation of the proposed technique.

4.4. Brute-Force Attack Analysis. Since the number of subsets and the number of symbols within these subsets come out to be different for different images, the trials required to retrieve the image without the knowledge of security keys have been discussed in a generalized manner. It has been proved that decoding a Huffman encoded bitstream without the knowledge of dictionary is a NP-complete problem. Thus, the retrieval of plaintext, just by knowing the ciphertext, is not possible [20, 21]. This ensures security against ciphertext-only attacks. Also, an intruder requires $2^{n-2} C_{n-1}/n$ (where n is the number of leaves) trials to retrieve information about the Huffman tree, if the probability distribution of source symbols is not known.

To break the proposed cryptosystem, an intruder would require the knowledge of correct symbol values as well as their corresponding codewords. The correct corresponding codewords can be obtained by either speculating the correct initial conditions, or trying different mapping functions for different subsets. Both mathematical and simulation analyses have been performed to verify the resistance of proposed technique against brute-force attack analysis.

Figure 5 (first row) shows the obtained results when correct initial conditions have been speculated and Figure 5 (second row) indicates the results when an assumption for correct mapping function has been made. The obtained visual results indicate that an assumption of the key will not lead to correct image and exact key would be required to break the cryptosystem. However, retrieving the correct key

FIGURE 4: Effect of encryption on codeword length.

is not simple; the relevant mathematical analysis is explained hereafter.

The former way of obtaining the codewords, that is, finding the correct initial conditions, would be difficult due to the high key sensitivity and large key space of the logistic map. Analysis shows that the two parameters denoting initial conditions of logistic map have a sensitivity of 10^{-13} and 10^{-14} , thus giving a key space of $N \times 10^{27}$, where N is the total number of subsets. Thus, an intruder would require a large number of trials to obtain the correct key.

To retrieve the correct dictionary, second alternative is to attempt different mapping functions for different subsets. The number of trials required to find the correct mapping function for a subset depends upon its size and can be described as follows. For a subset of cardinality n , there exist $n!$ different possible mapping functions. The dictionary can be correctly retrieved, if and only if all the subsets are correctly reshuffled back to their original form. This would require $\prod_{j=1}^N n_j!$ trials, where n_j is the number of symbols in the subset j and N denotes the total number of disjoint subsets.

Along with the correct mapping function, an intruder would also require the correct symbol values to correctly decrypt and decode the encrypted data. The number of trials required for retrieving the correct symbol value can be given by $\prod_{k=1}^4 t_k!$, where t_k denotes the symbol-length in the chosen subset. Due to the straightforward approach of modifying the single digit and two-digit symbols, they have not been included in calculating the key space or the trials required to break the cryptosystem. Thus, the total number of trials required for getting the correct dictionary comes out to be $\prod_{j=1}^N n_j! \times \prod_{k=3}^4 t_k!$. Hence, an intruder needs to try keys from a large enough key space in an attempt to unscramble the dictionary and retrieve the plaintext.

FIGURE 5: Brute-force attack analysis.

4.5. Known-Plaintext Attack Analysis. Since, in the proposed technique, both symbol values and their corresponding codewords have been modified, the security against known-plaintext attacks has been achieved. For instance, in the example indicated in Table 1, an attempt by an intruder to find the codeword of original symbol value "72", in the modified dictionary, would not lead to any codeword. This is due to the fact that the value "72" has changed to "27" in the modified dictionary. Let us suppose that the intruder correlates the symbol "72" and "27" (Table 1). In this case also, the codeword obtained by him will not be correct, as the mapping function is also modified in the scrambled dictionary. The symbol "72" correlated to "27" would return a codeword "000111010111" instead of actual codeword "001101100010".

Thus, even after getting the correct symbol values, the retrieval of correct corresponding codeword would not be easy. This is due to the modified symbol values as well as the mapping function. An attempt by an intruder to obtain a codeword corresponding to an original symbol would crash the decoder. This reflects the resistance of proposed technique against the known-plaintext attacks.

4.6. Approximation Attack Analysis. Security of the proposed technique has also been verified against approximation attacks [13]. In this attack, a part of unknown data is replaced with random data, in an attempt to obtain an approximate

copy of the original content. In the proposed technique, replacement of an unknown symbol or codeword with an arbitrary symbol or codeword may lead to duplication of an existing dictionary entry. This would violate the prefix condition of Huffman symbol-to-codeword mapping table. Decoding of the data with an invalid partially assumed dictionary would not be successful. Hence, this will not result in even an incomprehensible or approximate image. This reflects the robustness of proposed technique against approximation attacks.

4.7. Comparative Analysis. This section performs the comparative analysis of the proposed technique with other Huffman encoding based encryption techniques. The comparative analysis has been summarised in Table 3 and discussed hereafter.

The first technique proposed by Wen et al. [4] initially maps the variable length codewords (VLC) to fixed length indices. These indices are then concatenated and encrypted using a conventional cipher. This results in an equivalent number of indices but disturbs the compression efficiency of the encoder. This is due to the probability that an index of a short length codeword may get encrypted to an index of a long codeword. In contrast, the proposed technique contemplates the probability distribution of the symbols to change the mapping function. In the proposed technique

TABLE 3: Comparison of the proposed technique.

Technique	Ideal	Wen et al. [4]	Kankanhalli and Guan [5]	Wu and Kuo [8]	Proposed
Visual degradation	High	High	High	—	High
PSNR	Low	—	—	—	Average
Key sensitivity	High	Average	—	High	High
Compression efficiency	High	Low	High	Average	High
Brute-force trials	High	Average	High	High	High
Validity of scrambled dictionary	Yes	No	Yes	No	Yes
End-to-end dictionary	Yes	Yes	No	No	Yes

modified mapping between same length codewords has been implemented, which does not alter the compression efficiency of the encoder.

Kankanhalli and Guan [5] proposed to flip the last bit representing the codewords. This last bit flipping, however, violates the prefix condition of the dictionary, which can consequently crash the decoder. To avoid such a situation, a rule has been defined, which randomly flips the last bit and modifies all other codewords to maintain the dictionary invalidity. Following this rule, validity of dictionary has been attained, but this would consume additional computational resources with every one bit flip in the dictionary. In contrast, the proposed technique neither disturbs the codeword nor consumes any additional resources to maintain the dictionary validity. The symbol values are modified in a manner, such that a valid modified dictionary becomes a natural outcome of the scrambling process.

In the technique proposed by Wu and Kuo [8], each symbol value has been encoded by a different Huffman tree, which is generated by mutation of four basic trees. To enhance the security level, randomly certain bits have been inserted in the codestream. Since the multiple trees do not have the same length codewords, this technique adversely affects the compression efficiency and consumes high computational resources in synchronizing the encoder-decoder pair. In addition to this, the random bit insertion disturbs the validity of the codestream, and hence this does not support the content adaptation of the encrypted data. Thus, for processing operation at intermediate nodes, the communicating nodes need to either possess or generate all the employed Huffman tables. This consumes additional resources. However, the present proposed technique disturbs neither the compression efficiency nor the validity of codestream of Huffman dictionary. As the dictionary/codestream validity has been maintained in the proposed technique, content adaptation is possible even on the encrypted data. This saves the computational resources at the intermediate nodes and also does not require the intermediate nodes to possess the security keys. Subsequently, this enhances the security level.

The comparative analysis reveals that the proposed technique neither affects the compression efficiency nor consumes additional resources to generate a valid scrambled dictionary. This is due to the consideration of probability distribution for source symbols and the fulfilment of prefix condition, while modifying the dictionary. This ensures data processing or decoding even on the encrypted data and

gives an advantage to the proposed technique over other dictionary-scrambling based techniques. This also justifies its implementation for real time secure end-to-end image transmission over insecure communication channel. Since the shuffling has been performed from the viewpoint of supporting adaption techniques on encrypted data, the proposed technique can be extended to video format or any other multimedia data that employs Huffman encoding at the entropy coding stage.

5. Conclusion

Joint compression and encryption framework provides high encryption and compression efficiency for real time secure multimedia applications. Such existing techniques consume significant overhead at intermediate nodes to decrypt, process, and reencrypt the secured content. This is also a potential security loophole despite the strongest encryption technique used.

Network-friendly encryption technique is thus proposed, so that the processing operations can directly be performed on the encrypted data. The proposed technique alters the Huffman dictionary using chaos based logistic map. With the contemplation of probability distribution for source symbols and the prefix condition of the Huffman dictionary, data decoding has been achieved even with the scrambled dictionary. Thorough analysis of the proposed technique has been done. This reflects the high perceptual degradation, low PSNR, high key sensitivity, and no adverse effect on the compressibility of the used encoder. Security analysis reveals the ability of the proposed technique to resist approximation and cryptanalytic attacks. The ability to support content adaptation, even on the encrypted content, would make the proposed technique a suitable candidate for real time secure end-to-end communication systems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] W. Zeng, J. Lan, and X. Zhuang, "Security for multimedia adaptation: architectures and solutions," *IEEE Multimedia*, vol. 13, no. 2, pp. 68–76, 2006.

- [2] J. Chen, J. Zhou, and K.-W. Wong, "A modified chaos-based joint compression and encryption scheme," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 2, pp. 110–114, 2011.
- [3] A. Pande, P. Mohapatra, and J. Zambreno, "Securing multimedia content using joint compression and encryption," *IEEE Multimedia*, vol. 20, no. 4, pp. 50–61, 2013.
- [4] J. Wen, M. Severa, W. Zeng, M. H. Luttrell, and W. Jin, "A format-compliant configurable encryption framework for access control of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, 2002.
- [5] M. S. Kankanhalli and T. T. Guan, "Compressed-domain scrambler/descrambler for digital video," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 2, pp. 356–365, 2002.
- [6] Z.-L. Zhu, Y. Tang, Q. Liu, W. Zhang, and H. Yu, "A chaos-based joint compression and encryption scheme using mutated adaptive Huffman tree," in *Proceedings of the 5th International Workshop on Chaos-Fractals Theories and Applications (IWCFTA '12)*, pp. 212–216, October 2012.
- [7] H. Shen, L. Zhuo, and Y. Zhao, "An efficient motion reference structure based selective encryption algorithm for H.264 videos," *IET Information Security*, vol. 8, no. 3, pp. 199–206, 2014.
- [8] C. P. Wu and C. C. J. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 828–839, 2005.
- [9] J. Zhou, Z. Liang, Y. Chen, and O. C. Au, "Security analysis of multimedia encryption schemes based on multiple Huffman table," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 201–204, 2007.
- [10] G. Jakimoski and K. P. Subbalakshmi, "Cryptanalysis of some multimedia encryption schemes," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 330–338, 2008.
- [11] M. Pazarci and V. Dipçin, "A MPEG2-transparent scrambling technique," *IEEE Transactions on Consumer Electronics*, vol. 48, no. 2, pp. 345–355, 2002.
- [12] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 118–129, 2003.
- [13] Y. Mao and M. Wu, "A joint signal processing and cryptographic approach to multimedia encryption," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2061–2075, 2006.
- [14] N. Taneja, B. Raman, and I. Gupta, "Selective image encryption in fractional wavelet domain," *International Journal of Electronics and Communications*, vol. 65, no. 4, pp. 338–344, 2011.
- [15] N. Taneja, B. Raman, and I. Gupta, "Combinational domain encryption for still visual data," *Multimedia Tools and Applications*, vol. 59, no. 3, pp. 775–793, 2012.
- [16] N. Taneja, B. Raman, and I. Gupta, "Chaos based cryptosystem for still visual data," *Multimedia Tools and Applications*, vol. 61, no. 2, pp. 281–298, 2012.
- [17] D. S. Hirschberg and D. A. Lelewer, "Efficient decoding of prefix codes," *Communications of the ACM*, vol. 33, no. 4, pp. 449–459, 1990.
- [18] D. Kuo, "Chaos and its computing paradigm," *IEEE Potentials*, vol. 24, no. 2, pp. 13–15, 2005.
- [19] G. Álvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 16, no. 8, pp. 2129–2151, 2006.
- [20] D. W. Gillman, M. Mohtashemi, and R. Rivest, "On breaking a Huffman code," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 972–976, 1996.
- [21] T. Uehara, *Contribution to image encryption and authentication [Ph.D. thesis]*, University of Wollongong, 2003.