Joint Compression-Encryption Technique Based On Arithmetic Coding for JPEG Images

Hassan Y. El-Arsh Mohamed, Amr Abdelaziz, Ahmed Elliethy, Hussein A. Aly and Alaa Eldin Rohiem Shehata
Military Technical College
Cairo, Egypt
Email: hassanmohamed@uvic.ca, a.s.elliethy@mtc.edu.eg, abdelaziz.7@osu.edu, haly@ieee.org, alaa_rohiem@yahoo.co.uk

***Abstract*—An efficient joint compression-encryption technique for JPEG images using nonlinear properties of the arithmetic coder is proposed. The technique performs both entropy coding and encryption in a single step by manipulating the probability maps of the arithmetic coder while preserving the syntax structure of the image. The proposed scheme operates on a bit-by-bit basis while taking into account the previously processed data-bits keeping the advantages of both stream and block ciphers. The proposed technique is implemented using Libjpeg-turbo. The results demonstrate that the proposed scheme attains a security level comparable to that of AES while reducing both delay and system resource requirements while maintaining the compression efficiency of the standard JPEG encoder.**

***Keywords*—JPEG, non-linearity, AES, arithmetic coding.**

I. Introduction

Combining compression and encryption into a single process is a new approach to secure multimedia contents [1]–[5]. Typically, multimedia is secured through traditional coding-encryption methodology. This methodology encodes multimedia first and then encrypts it using cryptographic algorithms like AES, which requires additional time and resources and may not be compatible with some codec features such as compressed domain processing[1] in JPEG2000 [6] and progressive display[2] in JPEG [7] and JPEG2000.

Arithmetic coder is a lossless entropy coder used for most widespread multimedia coding standards [6]–[9]. This is due to its higher compression efficiency than traditional Huffman coder ($5-10\%$ higher than Huffman coder) [10]. Arithmetic coder is included in JPEG image codec [7] and H.264 [8] video codec as an optional advanced option instead of Huffman coder. For more recent multimedia standards, which require more compression efficiency like JPEG2000 [6] image codec and H.265 [9] video codec, the arithmetic coder is the only available option.

In this paper, a lightweight joint-compression-encryption technique is proposed to extend the functionality of the arithmetic coder for JPEG to achieve both coding (compression) and encryption simultaneously in a single process without performing an additional encryption stage while achieving an acceptable security margin.

This paper is organized as follows: the next section describes some detailed information about the arithmetic coder for JPEG, which is necessary to explain the rest of the paper. Section III illustrates the previous work related to the proposed technique. Section IV provides a complete explanation of the proposed technique comparing the proposed technique with related works. Section V presents numerical results assessing the security and performance of the proposed technique. Finally, the conclusion and the contributions of this paper are summarized in Section VI.

[1]This feature enables the extraction and processing of code-blocks from the compressed bit-stream in a random manner. Also called random access.

[2]Using this feature, the image is divided into more than one resolution layers enabling a viewers with limited bit rates or limited monitor resolution to view lower quality versions if the image as soon as a small portion of the image is received.

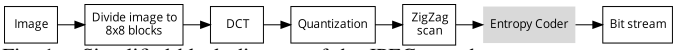


Fig. 1. Simplified block diagram of the JPEG encoder.

Fig. 2. Simplified block diagram of the JPEG decoder.

II. Arithmetic Coding in JPEG

Figure 1 shows the basic components of the JPEG encoder [7]. After the image is converted from RGB to YUV color space (except the gray-scale images), each color component is divided into 8×8 blocks. Then, two-dimensional Discrete Cosine Transformation (DCT) is applied per each block. Each of the resulting DCT blocks is converted to a one-dimensional vector by the zigzag scan followed by the entropy-coder. Figure 2 illustrates the JPEG decoding.

QM-coder, the standard arithmetic coder of JPEG [7], is a multiplication-free adaptive binary arithmetic coder that codes binary data streams by simple lookup-table-based operations in a state-machine-like behavior. To avoid complex multiplications and scaling calculations, the QM encoder specifies a discrete-probability table containing 114 states as given in Table I. Although the simple lookup operations highly reduce the implementation complexity and increase the coding speed, it incurs some approximations which have a relatively small effect (can be negligible) on the compression ratio [7].

According to Table I, each row in QM-coder's table is considered a state representing a certain probability map which models the probability of the Less Probable Symbol (LPS) and the More Probable Symbol (MPS) by Q_e and $1-Q_e$

TABLE I. The standard QM-table for JPEG

Index	Q_e Value	NMPS	NLPS	Switch
0	0x5A1D	1	1	1
⋮	⋮	⋮	⋮	⋮
i	Q_i	M_i	L_i	S_i
⋮	⋮	⋮	⋮	⋮
112	0x59EB	112	111	1
113	0x5A1D	113	113	0

respectively. The QM-coder starts at an initial table entry (state) i, which may be modified through the coding process according to the entries NMPS (Next index after MPS) and NLPS (Next index after LPS). During the coding process, the encoder decides whether the input bit (either "0" or "1") is coded as MPS or LPS. Thus, the next state will be either M_i, or L_i. When the Switch flag bit equals 1, the coder will interchange the values of LPS and MPS. Due to space limitations, the reader may refer to [7] for more details.

The diffusion and the avalanche effect for the arithmetic coder are important criteria for using the arithmetic coder for security. According to [1], [2], [11], [12], the arithmetic coder is characterized by high error sensitivity and error propagation properties. Moreover, it is proven by [13] that the arithmetic coder can be modeled as a chaotic random generator with proven cryptographic nonlinear properties. Furthermore, a practical experiment described in [14] uses the NIST's standard cryptographic test tool for randomness [15] to support these cryptographic properties. Thus, any modification in the input bit-stream on the encoder/decoder end, even in a single bit, triggers a significant avalanche effect on all subsequent output bit-streams that are encoded/decoded. Subsection IV-A demonstrates these properties.

III. Previous work

The first attempt to utilize the arithmetic coder for encryption was the Multiple State Indices (MSI) within the QM-coder [3], where the allowed QM-coder states are restricted based solely on the encryption key. As a result, the compression efficiency experiences a significant reduction, leading to an increase in the size of the compressed images by 139% [16] due to changing the probability model.

Another approach is the forbidden symbols [11], [12] which utilizes the non-linearity and diffusion properties of the arithmetic coders for extending its usage for error detection and encryption in tandem with compression. This is done by inserting one or more dummy symbols within the probability map. Dummy symbols are assigned relatively small probability values (i.e. small regions in the probability map). The cost for this additional feature regarding compression efficiency can be neglected [12].

Interval swapping of the arithmetic coder was first introduced in [17]. This scheme, called the Randomized Arithmetic Coder (RAC), inverts the probability map of MQ-coder[3] for each coded bit according to a pseudo-random bit-stream. Although RAC is considered secure and immune against most known attacks [16] and also preserves the probability model conserving the compression efficiency, it requires generating a pseudo-random bit-stream with the same length as the uncompressed bit-stream which may not be suitable for limited-resources-systems. Additionally, it will be more resources-efficient to encrypt the image with a standard stream-cipher which will generate a pseudo-random bit-stream with a length the same as the compressed bit-stream. This drawback had been solved in [16] by swapping the MQ-coder's probability map using only 94-bit key per each code-block (typically 4096 coefficients [6]) instead of generating a pseudo-random sequence of the same length as the uncompressed stream.

[3]The arithmetic coder of JPEG2000.

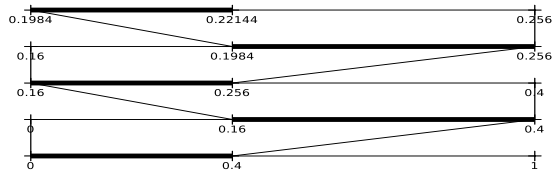
IV. The Proposed Technique

A. Design idea

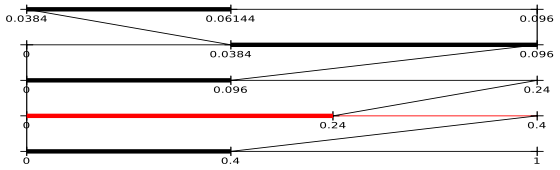
The fundamental design idea of the proposed scheme is the swapping of probability maps of the arithmetic coder while keeping the probability model intact. The following toy example describes this idea. For simplicity and without loss of generality, consider encoding an input sequence "01010" and assume the probabilities for "0" and "1" symbols are fixed and equal to 0.4 and 0.6 respectively. The standard encoder procedures are shown in Fig.3(a) with the resulting interval being $[0.1984 \quad 0.22144]$. In Fig.3(b), the probability map is swapped for only the second bit (marked with red color). In this scenario, the resulting interval is $[0.0384 \quad 0.06144]$. According to [18], the required number of bits (B) to represent an interval of width (W) is:

$$B = \lceil -\log_2(W) \rceil \qquad \text{bits.} \tag{1}$$

Accordingly, in both cases the final resulting interval has the same width of 0.02304, and hence the compression efficiency is maintained as the required number of bits to encode the interval depends solely on the interval's width. However, when attempting to decode the region from Fig.3(b) normally, the decoded sequence will be "00011", which is 40% different from the input sequence with only single swapping for a relatively short stream of only 5 symbols. Conversely, when the decoder considers swapping the probability maps same as in Fig.3(b), the sequence will be recovered correctly. The swapping of the probability map in Fig.3(b) can be associated with a secret key, as described in the next subsection.



(a) Example for original arithmetic coder.



(b) Example for the modified arithmetic coder.

Fig. 3. The original coder (a) versus the swapped one (b).

B. Implementation

The proposed technique is an adapted version of the general framework introduced in [16]. This framework relies on the key-based interval-swapping idea while preserving the probability model avoiding the generation of a pseudo-random sequence like [17]. Unlike [16], the proposed scheme does not change the key per each DCT block for a couple of reasons. First, the QM-coder is designed with 114-states, which allows the use of keys up to 228-bits in length forcing the brute-force attacker to perform an average of 2^{227} trial attempts. In contrast, the MQ-coder has only 47-states, with a maximum key length of 94-bits, which forces the authors in [16] to change the key in every code-block to achieve an acceptable security margin. Since the AES encryption standard employs

TABLE II. THE MODIFIED QM-TABLE FOR JPEG. THE KEY BIT K_i^0 IS APPLIED WHEN THE PREVIOUSLY CODED SYMBOL IS 0, OTHERWISE (IN THE CASE OF 1), K_i^1 IS APPLIED.

Index	Q_e Value	NMPS	NLPS	Switch	KEY
0	0x5A1D	1	1	1	K_0^0,K_0^1
⋮	⋮	⋮	⋮	⋮	⋮
i	Q_i	M_i	L_i	S_i	K_i^0,K_i^1
⋮	⋮	⋮	⋮	⋮	⋮
112	0x59EB	112	111	1	K_{112}^0,K_{112}^1
113	0x5A1D	113	113	0	K_{113}^0,K_{113}^1

key-lengths of 128, 192, and 256 bits, which are proven secure key-lengths, it is unnecessary to change the key for every DCT block. This approach provides an adequate level of security while also reducing the necessary computational resources. Second, as each code block in JPEG2000 comprises 4096 coefficients whereas a DCT block in JPEG contains only 64 coefficients, changing the key for every DCT block would result in a more resource-intensive process. Thus, 228-bit key stream is generated for the whole image(s) and used in conjunction with each destination state in Table II.

The modified coding process runs as follows. Assuming the encoder is at the i^{th} state and the previously coded symbol is 0 (or 1), then if the key bit $K_i^0 = 1$ (or $K_i^1 = 1$), the probability map of the i^{th} state will be inverted. Otherwise, the probability map is kept. Consequently, as the coding process proceeds, the interval denoting the coded sequence keeps changing according to the key, as described in Subsection IV-A. However, the final interval width does not differ from the standard encoder as the proposed scheme keeps the probability model intact preserving the compression efficiency. It is critical to note that the all-zero key results in standard coding operation (i.e. no encryption) as key administration techniques are beyond the scope of this paper. Since the proposed scheme modifies only the QM-coder operation, the syntax of the encrypted JPEG image will remain unaffected. The proposed technique is implemented in [19] using the official open-source library *Libjpeg-turbo* [20].

V. EXPERIMENTAL RESULTS

A. Subjective and comparison assessments

Figure 4 shows a sample image from the dataset [21] encrypted using the proposed QM-coder and the AES in counter-mode of operation. For both techniques, the encrypted image bears no resemblance to the original image. This will be further quantified in the following subsections. In order to assess the effect of the proposed technique on compression, a sample of 150 images was chosen randomly from each of the datasets [21], [22] which provide a variety of image textures and resolutions, with a total number of 300 images.

TABLE III. DIFFERENCE IN SIZE (Δ) BETWEEN THE ORIGINAL ENCODER AND THE PROPOSED TECHNIQUE FOR A THE SELECTED SAMPLE OF IMAGES W.R.T THE ORIGINAL IMAGE SIZE USING (2).

Max	Min	Mean	ST. Dev.
92×10^{-5}	-89×10^{-5}	-9.87×10^{-5}	29×10^{-5}

Table III compares the size of the standard JPEG encoded images to the size of the encrypted images with the proposed technique using the following equation:

$$\Delta = \frac{x - y}{y}, \tag{2}$$

(a) The original image (r018e74fet) from [21]).

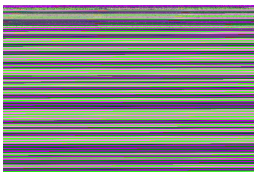
(b) The image (r018e74fet) encrypted using AES and decoded using the original MQ-coder.

(c) The image (r018e74fet) encrypted using the proposed scheme and decoded using the original MQ-coder.

(d) The image (r018e74fet) encrypted using the proposed scheme and decoded with only one bit key error.

Fig. 4. An example subjectively comparing the outputs of the proposed technique and AES.

(a) Sample image (r012d1dbet) from [21].



(b) Replacing all QM-coder data in the image (r012d1dbet) with an AES-encrypted data.

Fig. 5. The effect of JPEG decoder on the encrypted data.

where x and y refer to the size of the encrypted image and the size of the original image in bytes, respectively. It can be concluded from Table III that the proposed technique has a negligible effect on the size of the compressed image. Theoretically, it should not change the size at all. However, the approximations in mathematical calculations of the standard QM-coder and byte-stuffing cause these differences.

B. Peak signal-to-noise ratio (PSNR) measurements:

Peak Signal to Noise Ratio (PSNR) is a widely used metric to assess the quality of images and videos by evaluating the difference between the original and the modified versions. Table IV shows that images encrypted using the proposed scheme yield a higher average PSNR value than those encrypted using AES. However, both encryption techniques result in a very low PSNR, indicating a significant difference between the original and encrypted images. Actually, the achieved PSNR values are comparable to that of comparing two totally dissimilar images. As a practical example, the PSNR between the two images in Fig.4(a) and Fig.5(a) is 9.64 dB, which is higher than the maximum PSNR value of AES and the proposed technique. More details are introduced in Subsection V-D.

TABLE IV. COMPARISON OF PSNR FOR 300 SAMPLE IMAGES ENCRYPTED USING THE PROPOSED TECHNIQUE AND AES.

Technique	Max	Min	Mean	Std. Dev.
AES	9.59	5.4	7.67	0.84
Proposed	9.45	4.04	6.09	0.85

C. Strucure similarity index (SSIM) measurements:

Structure similarity index (SSIM) [23] is another commonly used measure of distortion for images and videos that involves structure comparison, hence it is more compatible with the human eye perception system than PSNR. Table V shows that images encrypted using the proposed scheme achieve a higher average SSIM value than those encrypted using AES. However, the achieved values are sufficiently low indicating no similarity between ciphered and plain images. As a practical example like the previous subsection, the SSIM index for the two images in Fig.4(a) and Fig.5(a) is 0.473636. Additional illustrations are introduced in Subsection V-D.

TABLE V. COMPARISON OF SSIM RESULTS FOR 300 SAMPLE IMAGES ENCRYPTED USING THE PROPOSED TECHNIQUE AND AES

Technique	Max	Min	Mean	STD
AES	0.01	0.004	0.0086	0.0013
Proposed	0.366	0.0134	0.121	0.074

D. Discussion of the results:

Although [14] proves that the output stream of the arithmetic coder passes the standard cryptographic randomness test [15] like AES, the displayed encrypted image of the proposed scheme in Fig.4(c) contains some sort of a structure that does not infer this fact, unlike the encrypted image by AES in Fig.4(b) which contains no structure at all. The reason behind this aspect can be deduced from Fig.2. During image decoding after the QM-coder decoding process, the output stream from the QM-coder is passed through inverse quantization and inverse DCT stages, which have the effect of low-pass-filter, making the incorrectly decoded image appears to have some sort of structure having higher PSNR and SSIM ratios than images encrypted with AES as shown in Tables IV and V. To prove this idea practically, we have replaced all QM-coder data within the image in Fig.5(a) with an AES-encrypted bit stream keeping other JPEG syntax intact. The resultant image is displayed in Fig.5(b) with PSNR=6.1 and SSIM=0.189 when compared to its original version in Fig.5(a).

Regarding the computational resources, the additional time consumption (indicating the required system resources) for the proposed scheme over the standard JPEG encoder/decoder did not exceed 1% for our dataset. On the other hand, as AES regenerates the data (after encoding and before decoding), the average overhead time ratio is 18% for the same dataset.

VI. CONCLUSION

This paper proposes a technique for efficiently compressing and encrypting JPEG images in a single step, utilizing the nonlinear properties of the arithmetic coder. The technique manipulates the probability maps of the arithmetic coder to perform both entropy coding and encryption, resulting in reduced delay and system resource requirements accompanied by high-security margins. The original decoder can still interpret the image's metadata but not the encrypted pixels as the proposed scheme preserves the image syntax. The implementation is performed using *Libjpeg-turbo*, the official open-source JPEG library.

REFERENCES

[1] S. Chaoui, O. Ouda, and C. Hamrouni, "A Joint Source Channel Decoding for Image Transmission," *Advances in Science, Technology and Engineering Systems Journal*, vol. 4, no. 6, pp. 183–191, 2019.

[2] L. Duan, X. Liao, and T. Xiang, "A secure arithmetic coding based on markov model," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, no. 6, pp. 2554–2562, 2011.

[3] C.-P. Wu and C.-C. Kuo, "Design of integrated multimedia compression and encryption systems," *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 828–839, 2005.

[4] R. Hasimoto-Beltran, M. D. Calderon-Calderon, and V. H. Olavarría-Jaramillo, "Secure real-time chaotic partial encryption of entropy-coded multimedia information for mobile devices: Smartphones," *IEEE Access*, vol. 10, pp. 15 876–15 890, 2022.

[5] T. Yang, J. Ma, Y. Miao, Y. Wang, X. Liu, K.-K. R. Choo, and B. Xiao, "Mu-teir: Traceable encrypted image retrieval in the multi-user setting," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1282–1295, 2023.

[6] ISO, *Information technology — JPEG 2000 image coding system — Part 1: Core coding system*. www.iso.org, Oct 2019, no. ISO/IEC 15444-1.

[7] ITU-T, *(JPEG-1)-based still-image coding using an alternative arithmetic coder*. www.itu.int, Aug 2005, no. T.851.

[8] ——, *Advanced video coding for generic audiovisual services*. www.itu.int, 2013, no. E 38445.

[9] ——, *High efficiency video coding*. www.itu.int, 2016, no. E 41298.

[10] G. K. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[11] M. Sinaie and V. T. Vakili, "A low complexity joint compression-error detection-cryptography based on arithmetic coding," in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, May 2010, pp. 233–236.

[12] S. Chokchaitam and P. Teekaput, "Protecting embedded error detection arithmetic coding from eavesdroppers," in *2005 Digest of Technical Papers. International Conference on Consumer Electronics, 2005. ICCE.*, Jan 2005, pp. 59–60.

[13] N. Nagaraj, P. G. Vaidya, and K. G. Bhat, "Arithmetic coding as a nonlinear dynamical system," *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 4, pp. 1013 – 1020, 2009.

[14] M. Sinaie and V. T. Vakili, "Secure arithmetic coding with error detection capability," *EURASIP J. on Information Security*, vol. 2010, pp. 4:1–4:9, Sep 2010.

[15] N. I. of Standards and Technology. (2010, April) Nist statistical test suite. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

[16] H. Y. El-Arsh and Y. Z. Mohasseb, "A new light-weight jpeg2000 encryption technique based on arithmetic coding," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, Nov 2013, pp. 1844–1849.

[17] M. Grangetto, E. Magli, and G. Olmo, "Multimedia selective encryption by means of randomized arithmetic coding," *Multimedia, IEEE Transactions on*, vol. 8, no. 5, pp. 905–917, 2006.

[18] A. Said, April 2004. [Online]. Available: https://www.hpl.hp.com/techreports/2004/HPL-2004-76.pdf

[19] H. Mohamed, "Implementation for joint compression-encryption technique based on arithmetic coding for jpeg images," accessed: 2023-07-14. [Online]. Available: https://github.com/HassanMohamedGit/JPEG-AC-Crypto-Coding

[20] Libjpeg-turbo. Accessed: 2023-07-11. [Online]. Available: https://libjpeg-turbo.org/

[21] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "Raise: A raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference*, ser. MMSys '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 219–224.

[22] G. Schaefer and M. Stich, "UCID: an uncompressed color image database," in *Storage and Retrieval Methods and Applications for Multimedia 2004*, M. M. Yeung, R. W. Lienhart, and C.-S. Li, Eds., vol. 5307, International Society for Optics and Photonics. SPIE, 2003, pp. 472 – 480.

[23] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.