Efficient and secure selective cipher scheme for MIoT compressed images

Hassan Noura, Ola Salman, Raphael Couturier, Ali Chehab

Efficient and Secure Selective Cipher Scheme for MIoT Compressed Images

Hassan N. Noura[1], Ola Salman[2], Raphael Couturier[1], and Ali Chehab[2]

[1]Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, Belfort, France
[2]American University of Beirut, Department of Electrical and Computer Engineering, Lebanon

***Abstract*—Recently, the pervasiveness of multimedia contents raised serious security and privacy concerns, especially with limited devices such as the ones used in the Multimedia Internet of Things (MIoT), which are constrained in terms of computations, memory capacity and power consumption. In this paper, we propose a lightweight format-compliant compression-selection cipher scheme, which could be adopted with different types of image compression algorithms (lossless or lossy). The two proposed cipher variants leverage the compressed data characteristics such as randomness and uniformity, as compared to uncompressed data; this distinguishes the proposed solution from the existing ones. Also, the proposed cipher supports configurable encryption parameters, such as the encryption data rate, to satisfy the requirements of the different MIoT applications, and to optimize the trade-off between security and efficiency. The two proposed variants, which do not require additional operations to produce a format-compliant code-stream, are based on the dynamic key approach, and they require a single round of simple operations. The first variant consists only of one cryptographic operation, permutation or substitution, while the second variant consists of two operations, permutation and substitution. The low number of rounds, combined with simple operations, result into low computational complexity and consequently, low energy consumption and latency. The process for updating the permutation and substitution tables is also lightweight, and it could be implemented in parallel for each sub-compressed part (tile or frequency level), which increases the security level at a minimal computational cost. The proposed substitution, permutation, and substitution-permutations variants exhibit a high throughput with an enhancement of at least 311%, 176%, and 125% compared to the optimized Advanced Encryption Standard (AES) implementation, respectively.**

***Index Terms*—Lightweight MIoT compression-cipher scheme; One round selective cipher; dynamic key-dependent cryptographic primitives.**

I. Introduction

The continuous and rapid advancements in information technology is giving rise to new types of applications, and new emerging networks such as the Internet of Things (IoT) [1]. An IoT system connects diverse sorts of physical objects and devices to the Internet [2]; these devices process and exchange massive amounts of data and enable a multitude of applications in various domains. The sheer number of IoT devices, their heterogeneous types, and the large amounts of exchanged information impose strict constraints on the underlying network in terms of security, scalability and quality of service [3], [4]. Typically, the low-cost IoT devices are restricted in terms of computational capabilities, memory capacity and power resources [5].

In this context, the introduction of cheap hardware devices such as CMOS cameras and microphones, gave rise to the design of large-scale Multimedia IoT (MIoT) networks [2], [6] (see Figure 1). Within such networks, numerous MIoT applications have been developed such as live monitoring and surveillance that require the transmission of large amounts of multimedia data including images, audios and/or videos.

Multimedia data is normally compressed to reduce the storage capacity and communication overhead. One main category of multimedia image compression algorithms is the JPEG family such as the original JPEG [7] and JPEG 2000 [8] that can be used in restricted environments such as MIoT.

A. Problem Formulation

The majority of MIoT devices are battery-powered, and hence, they have limited computational and communication resources, in addition to limited energy consumption. On the other hand, MIoT systems suffer from several security issues when multimedia streams are transmitted over insecure channels, making them prone to different passive and active security attacks. Accordingly, in the face of such a challenge, it is essential to define lightweight security solutions.

Thus, the main issue that the proposed solution addresses is the design of an efficient and flexible crypto-compression scheme for limited MIoT devices that handle compressed images. The solution aims to strike an optimal balance between the security level and the system performance, that is to minimize the computational complexity, energy consumption and thus, the associated delay. The solution significance and impact stem from enabling secure communications of MIoT compressed images between MIoT end devices and application server(s), with fewer computations, and reduced encryption latency, when compared to existing cipher solutions such as AES.

Therefore, in this paper, the objective is to propose an efficient and secure Data Confidentiality (DC) security service for MIoT systems, which is typically achieved through a symmetric cipher scheme.

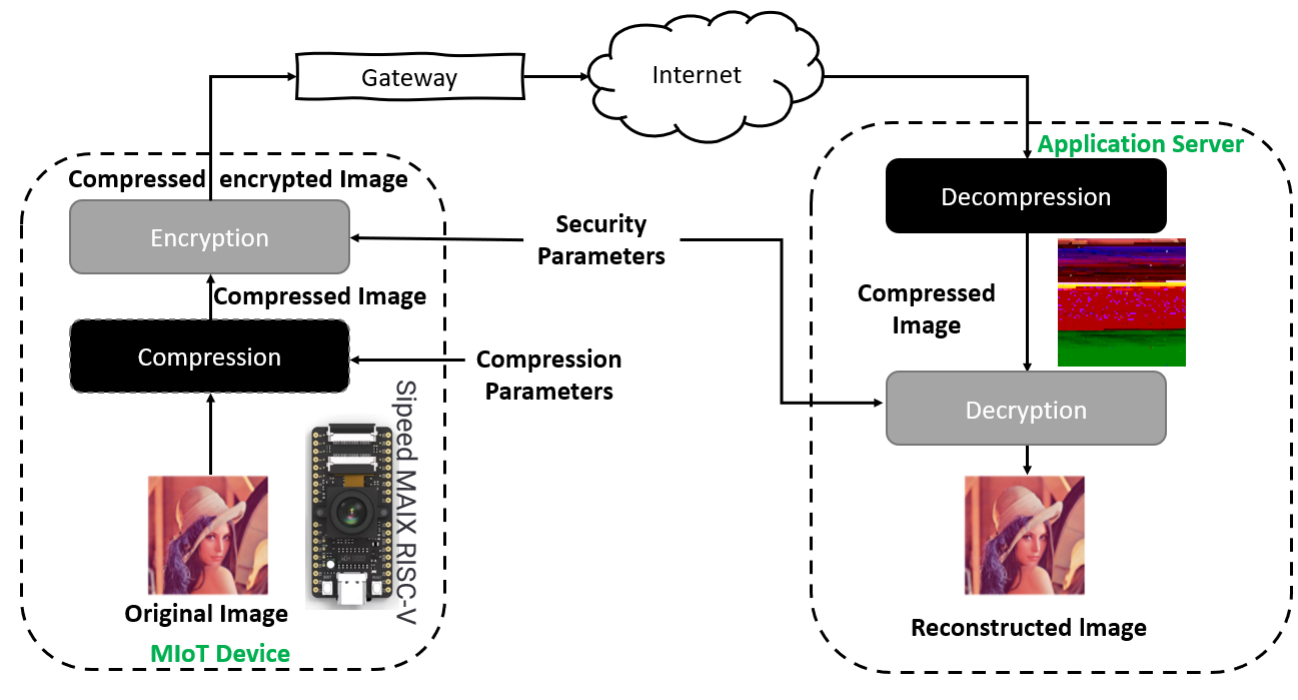


Figure 1: Example of MIoT system with MIoT devices that connect to the Internet via a Gateway and then to an application server(s)

For emerging MIoT devices, a data confidentiality scheme needs to optimize two key points:

1) Minimize the compressed data part to be encrypted for higher efficiency. However, it is important that the selected part leads to a high visual degradation to prevent attackers from recovering useful information from the visual contents. Such approach of **selective encryption** has been presented in [9]. In general, when hard visual degradation is required, the selected portion represents a significant part of the compressed data, and the overhead is proportional to the size of the selected portion.
2) Minimize the computational complexity and required resources by adopting a lightweight cipher scheme. The current encryption algorithms, such as the Advanced Encryption Standard (AES), are based on multiple iterations of a round function that includes several operations such as addition of round key, substitution and diffusion [10]. The associated delays and consumed resources depend on the number of rounds r and the type of operations. Unfortunately, when adopting a static encryption structure with fixed cryptographic primitives, there is always a need for a large number of rounds, the overhead of which cannot be offset by optimized implementations. Hence, the need for a cipher scheme with the minimum possible number of rounds, and operations per round.

Recently, researchers have been targeting the the design of "Lightweight" cryptographic algorithms [11], [12] through the design of a simple round function, or the reduction of the number of rounds, r.

The authors of this paper have been working on the design of solutions based on the dynamic key approach. In such schemes, the cryptographic primitives are variable (dynamic) enabling the reduction of the number of rounds to just 1 or 2 rounds, and thus, leading to minimal latency and requiring just few resources, while maintaining a high level of security [13], [14], [15], [16], [17] [18], [19].

Moreover, from an implementation perspective, it would be more practical to define a generic framework for selective crypto-compression independent of the underlying multimedia compression algorithm. Several encryption solutions were proposed recently, but they are based on a specific compression algorithm such as JPEG or JPEG 2000. Moreover, the **joint compression-encryption** approaches [20], [21], [22], where data encryption and compression operations are performed together cannot ensure such a property. One can choose to encrypt before or after compression, however, doing so before compression leads to a non-efficient compression process. Therefore, it is preferable to apply encryption after compression to preserve the efficiency of compression algorithms and to ensure that the encryption scheme is independent of the compression algorithm.

B. Motivation & Contribution

In this work, we aim to preserve the resources and lifetime of MIoT devices when protecting their enormous amounts of data. Given that the encryption process is the most energy-consuming function [], we propose a lightweight selective format-compliant cipher scheme, an efficient crypto-compression solution that accounts for the limitations of constrained MIoT devices. The scheme leverages the

randomness and uniformity of the compressed data to reduce the required iterations and operations, and thus, reducing the latency and resources overhead.

In the proposed cipher variants, the encryption level depends on the compressed data structure and can be performed either at the packet level, if the compressed data is organized into packets such as JPEG 2000, or at any other possible level such as the tile level.

The first variant of the proposed one-round cipher scheme consists of one operation, either permutation or substitution, whereby the permutation or substitution tables vary at the different levels. The second variant requires two operations of permutation and substitution. Both variants do not employ the addition round key operation, taking advantage of the randomness and uniformity already introduced by the compression process of the multimedia contents.

In order to highlight the contribution and originality of this work, as compared to the authors' previous works on dynamic key-dependent ciphers, we list below the main differences:

1) The previous solutions were encryption schemes with one or two rounds, including multiple operations per round (permutation, addition round key, matrix diffusion and substitution), and they could be applied to different data types. The proposed solution is designed for compressed multimedia contents, and as a proof of concept, it was tested with different image compression algorithms such as BMP, JPEG, and JPEG 2000.
2) The proposed solution has two variants that could be adopted based on the sensitivity of the data; the first variant is recommended for confidential MIoT compressed images, while the second variant for highly confidential images.
3) The cryptographic primitives are variable; they get updated, as configured, for each input packet, tile, or image.
4) The proposed scheme has minimal computational complexity as compared to the previous lightweight dynamic key-dependent ciphers; one variant requires a single operation, while the second just two operations, which in turn results into lower latency and required resources.
5) The existing lightweight ciphers can be used as full encryption schemes, but they do not satisfy the format-compliant property in a similar manner to the proposed one.
6) The proposed cipher scheme can be implemented in a parallel manner, at the packet or tile level, which further enhances its performance and efficiency.

The characteristics of the proposed scheme allows it to outperform the existing optimized standard ciphers such as AES in CTR mode [23], as well as the existing dynamic cipher schemes [24], [25], [26], [16].

Moreover, the proposed solution has a flexible design since a configurable percentage of data pr ($\geq 5\%$ for JPEG family) is selected in a random and uniform manner from the compressed data tiles or packets to be encrypted. The minimum allowable percent is selected based on the visual degradation representing the minimum introduced overhead. Increasing this percentage results in achieving a higher security level but at the expense of additional computations and latency.

Another advantage of the proposed scheme is its simple implementation in software and hardware, since the decryption process is very similar to encryption, except for using the inverse permutation and substitution tables, and most importantly, the error propagation is limited since the avalanche effect is avoided using the dynamic key approach.

C. Organization

The rest of this paper is organized as follows. Section II presents the related work of the proposed solution. Then, in Section III, we present the dynamic key derivation scheme and cipher primitives construction technique. Next, the two cipher variants are presented in Section IV. A set of extensive security tests are discussed in Section V. Then, Section VI analyzes the encryption ratio threshold for several codecs: BMP, JPEG, and JPEG2000, and a cryptanalysis discussion is provided in Section VII. Then, the performance analysis of the proposed cipher variants is presented in Section VIII. Finally, Section IX concludes this paper.

II. Background

In this section, we describe the concept of image compression algorithms that can be employed with MIoT devices and the existing crypto-compression schemes. Let us indicated that the compression parameters will be selected according to the employed MIoT devices. These parameters will define the percentage of overhead in terms of computation, energy consumption and communication delay.

A. Image compression algorithms

The image compression techniques can be divided into two classes: lossless and lossy (as illustrated in Figure 2). The lossless compression techniques preserve better the visual content compared to the lossy ones since the decompressed image with lossless compression is the same as the original one. However, the lossless compression results are in larger compressed data size compared to the lossy one. In general, the lossy compression is used to reduce better the communication overhead and delay in existing networks such as the MIoT one. However, the compression ratio is inversely proportional to the image quality. Using a low compression ratio results into a high perceived image quality and vice versa. Thus, selecting a compression ratio depends on MIoT applications and hence, different compression parameters can be used with the different MIoT devices.

The existing lossy compression techniques, that can be used in the MIoT context, convert data from the spatial

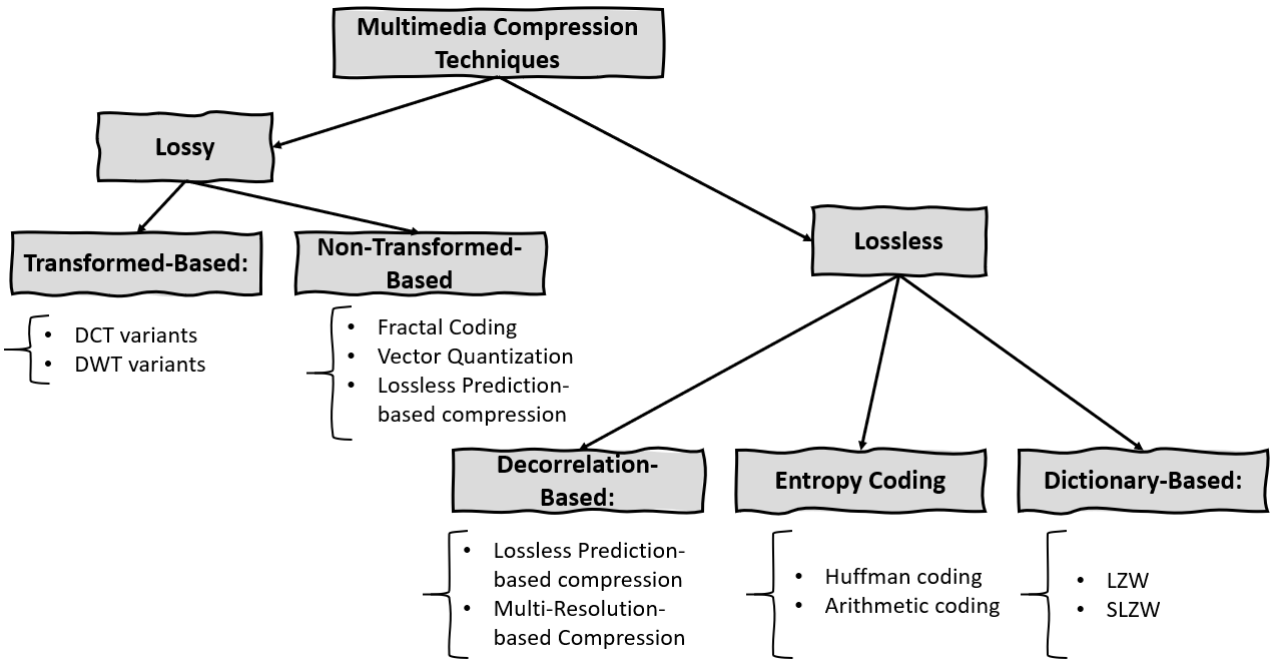


Figure 2: Taxonomy of multimedia compression algorithms

to the frequency domain by applying a spatial-frequency transformation such as the Discrete Cosine Transform (DCT2) used in JPEG [7] or Discrete Wavelet Transform (DWT2) as in JPEG 2000 [8]. On the other hand, DCT2 is also used in the well-used lossy video compression standards such as MPEG 1/2, AVC/H.264, MPEG-4, and HEVC. After the spatial-frequency transformation, the frequency values are quantized (according to the frequency level) and ordered, and then, they get through the Huffman or arithmetic encoding process. Given that the human eye is able to detect changes in low frequency regions, while disregarding the changes in pattern-heavy regions. Moreover, the less important data is represented by the high frequency components, while the most important information is represented by the lower frequencies. Accordingly, a lossy compression scheme tries to eliminate more data from the high and middle frequency components than from the low frequency ones, by increasing the compression ratio. Therefore, the frequency coefficients have different importance levels and thus, they are quantized according to their importance and the desired compression ratio. This can lead to different visual effects. A high compression ratio means that high and mid-frequency coefficients are quantized more coarsely compared to low-frequency coefficients. In the context of MIoT, The produced compressed image contents (after the source encoding step) are divided into packets, which are transmitted from MIoT devices to application server(s) through a gateway that will be connected to the Internet.

On the other hand, the compression ratio and other compression parameters can vary for different MIoT applications, and hence, for different classes of MIoT devices, as shown in Figure 3. Besides, increasing the compression ratio will reduce the size of compressed code-stream, and introduces a visual degradation. An example that illustrates the trade-off between the compressed data size and visual quality (measured by using PSNR and SSIM metrics) is shown in Figure 4 for the JPEG image compression with two standards images (Lena and pepper).

B. Existing crypto-compression schemes

In the literature, three main categories of crypto-compression schemes have been presented, according to the encryption order, as indicated in [27]:

1) **Transform-based schemes:** the encryption is applied on frequency values, after the frequency transformation step such as discrete cosine transform (dct2) or discrete wave transform (DWT) [28], [29], [30]. Unfortunately, the transform-based cipher schemes introduce a compression overhead [31] and hence, a communication overhead. Moreover, when using a static key, such schemes may not provide the required security level [32].
A recent scheme, presented in [20], uses distinct keys for encrypting different JPEG images to achieve a high security level. However, the technique is restricted to a specific image compression algorithm such as JPEG. Also, its computational overhead, in a similar manner to the one in [31], is related to all DC and AC values, which also introduces a large memory overhead. The algorithm in [20] relates the encryption key to the plain text, which leads to error propagation, and it uses chaotic maps (floating computing), which is typically associated with a high computational when compared to integer-based schemes. On the other hand, the crypto-compression scheme in [31] exhibits a high computational overhead since it requires two rounds of permutation, 8 iterations of scan for each block, in addition to the virtual decomposition operation.

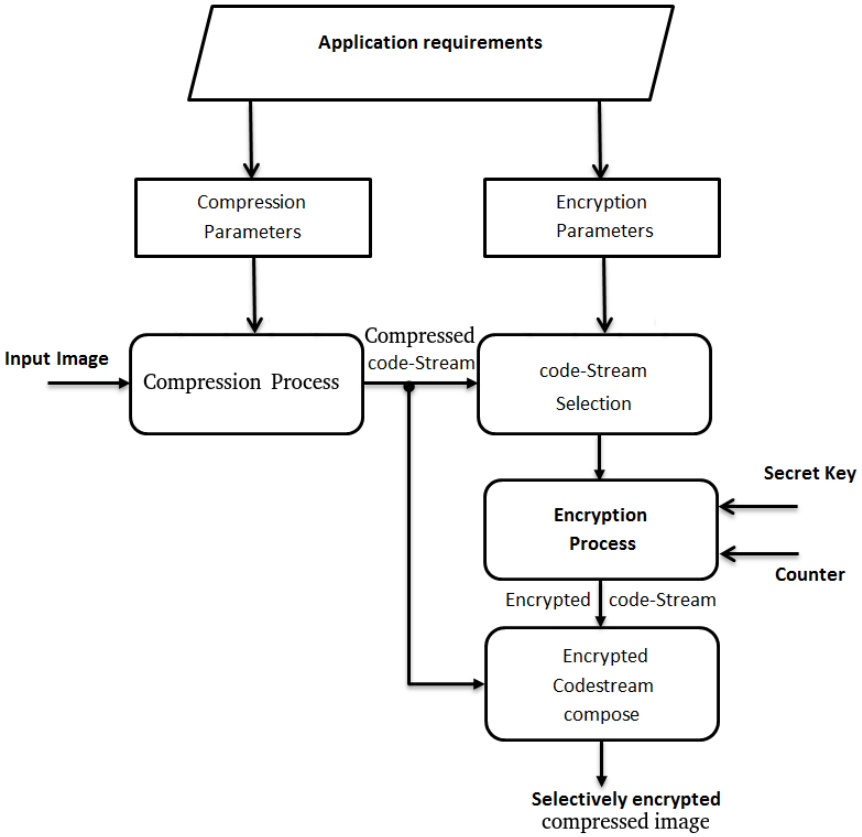
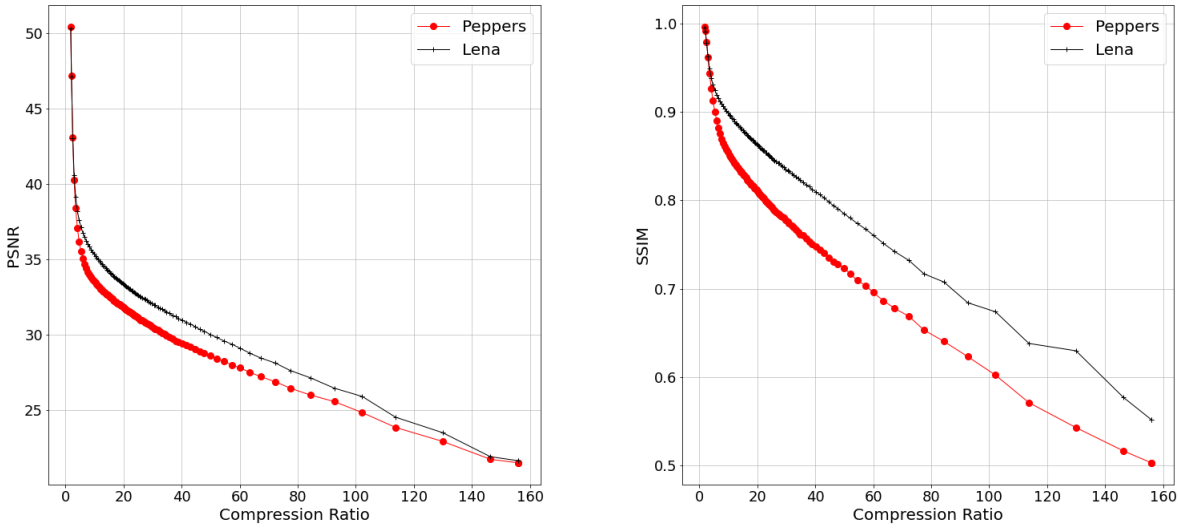


Figure 3: Proposed approach at the MIoT end device



(a) (b)

Figure 4: Variation of PSNR and SSIM versus compression ratio using the JPEG image compression algorithm, for two standard images Lena and peppers.

2) **Coding-based schemes:** This kind of schemes consists of coding-based encryption by relating the coding step to

a secret key. Therefore, encryption and entropy coding are performed in one step [21], [22]. However, this class of encryption algorithms requires modifying both the encoder and the decoder.

3) **Package/code-streams-based schemes:** the encryption process is performed on the code-streams, where a packet consists of a header and a body, as in JPEG 2000, unlike the case of traditional JPEG. Several approaches were presented under this class as in [33], [34], [35].
Some of the proposed techniques preserve the format-compliant property, as in [23]; it is a selective image encryption scheme that ensures confidentiality by encrypting only 5.43% of each data packet. In [36], the AES-128 block cipher standard is used with a modified Counter (CTR) mode (using a conditional modular addition instead of Exclusive OR (XOR)) to achieve the format-compliant property. Recently, an enhancement of the Masoudi et al. approach [23] was presented by Fawaz et al. [37], where only 4% of the compressed bytes of each packet are selected for encryption. The cipher structure is a dynamic key-dependent block cipher, and it requires only two rounds of substitution and diffusion operations. Also, the key is changed for every input image. This enhancement is based on adaptive cryptographic primitives to preserve the format-compliant criterion. Using the dynamic key-dependent cipher structure, the proposed method achieves a high level of security with reduced number of rounds. However, the approach requires two rounds in addition to the use of diffusion matrix operation for each round, which yields a significant overhead when compared to a substitution or permutation operation [16].

   On the other hand, the scheme in [38] encrypts the first 32 bits of every box (the box length Lbox) using the asymmetric cipher scheme (RSA) for JPEG 2000 images. However, using an asymmetric cipher to just encrypt the box length is not sufficiently secure since partial information can be recovered, in addition to a high associated overhead.

Note that performing encryption before compression introduces several challenges such as a high encryption rate, and since an encrypted image follows a uniform distribution with no redundancy among the pixels, this renders the compression algorithm inefficient.

The existing solutions in the literature are not appropriate for limited MIoT devices since they typically involve a large number of iterations (rounds), and/or several complex operations per round. To address such limitations, it is essential to adopt an efficient, lightweight, and flexible post-compression solution for MIoT devices, especially for the tiny devices. Such a solution needs to minimize the encryption computational overhead, which in turns reduces the associated latency and energy consumption, while maintaining an appropriate security level.

The objective of this work is to propose a simple and flexible, yet robust selective cipher scheme, for MIoT image compressed contents, which is either format- or non-format compliant as per the application requirements. Moreover, the parallel structure of the proposed cipher variants make them suitable for a powerful application server, which further decreases the decryption delay.

The encryption parameters, such as the percentage of compressed data per packet to encrypt, pr, can be configured according to the requirements of MIoT devices and applications. The optimal value of pr depends on the image compression algorithm, and it is defined as the minimum percentage of compressed data that should be encrypted while ensuring a hard visual degradation. Note that the optimal value, for lossless image compression algorithms, is higher than the value for lossy algorithms. We quantified the optimal value of pr for the lossless Bitmap (BMP) algorithm, $pr \geq$ 60%, and for the JPEG and JPEG 2000 algorithms, $pr \geq 5\%$.

In the following section, we describe the proposed solution that is based on the dynamic key dependent approach, and which caters for the limitations of MIoT devices in terms of computations, resources and delay. Note that to maintain the file format-compliant property, the header and sub-header should be kept unenrypted, and the ciphertext should avoid the range of the header and sub-header to enable decoding without decryption. For example, the range of header/sub-header marker vary within the [0xFF90, 0xFFFF] interval for the JPEG compression family.

III. Cipher Primitives Generation

The proposed scheme adopts the dynamic structure, and it can be configured such that the cipher primitives (permutation and/or substitution tables) vary for each packet, a set of packets, input image, or a set of input images. Also, the primitives are updated for each new authenticated session or a sub-session. As such, the primitives are unknown to attackers. The frequent update of the cryptographic primitives provides a higher level of security, yet at the expense of additional initialization overhead. The primitives are derived from the dynamic secret key, which has a large key space, and generated from a secret session key that is exchanged between MIoT devices and an application server. Note that there are different mechanisms for session key exchange, as presented in [10], which are based either on symmetric or asymmetric schemes.

A. Dynamic Key Derivation Scheme

Figure 5 illustrates the key derivation function that is performed at MIoT devices and the application server. It requires two inputs, a secret session key SK and a nonce N_o, and it is updated for each new session. The sender and receiver must be synchronized to generate the same nonce.

The secret key SK and the nonce N_o are XORed, and the output is hashed to generate the dynamic key DK. This leads to a higher sensitivity against any bit change either in the nonce or the secret key. The secure hash function ($SHA-512$)

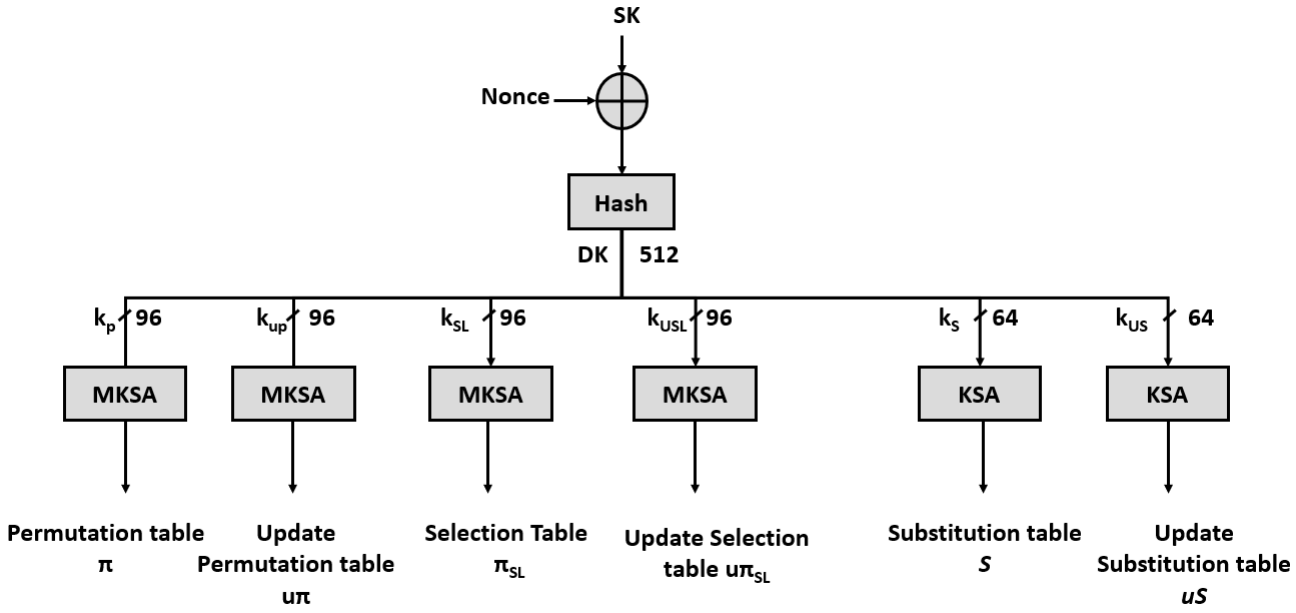


Figure 5: Dynamic key generation process, and the components used in the construction of cipher primitives and their update primitives.

is selected due to its strong resistance against collisions. Next, the dynamic key, Dk, is split into two sub-keys; the first one is used to construct the cipher primitives, and the second one for updating these primitives, as described next.

B. Cipher and Update Cipher Primitives Construction

DK consists of 512 bits, and it is divided into six sub-keys; the first 4 sub-keys have each a size of 96 bits, and the remaining two have each a length of 64 bits:
$DK = \{K_p,\ K_{up},\ K_{SL},\ K_{uSL},\ \ K_S,\ \ K_{uS}\}$.

The sub-keys K_p, K_{SL}, and K_S are used to generate the permutation table π, the selection table π_{SL}, and the substitution table S, respectively. On the other hand, K_{up}, K_{uSL} and K_{uS} are used to generate the primitives needed for the update process, the update permutation table $u\pi$, the update selection table $u\pi_{SL}$, and the update substitution table uS, respectively).

- K_p and K_{up} are used as seeds to generate the required permutation (π) and update permutation ($u\pi$) tables by using the Modified Key-Scheduling Algorithm (MKSA) of the Rivest Cipher 4 (RC4), as presented in [16].
- Similarly, K_{SL} and K_{uSL} are used as seeds with MKSA to produce the selection π_{SL} and update selection ($u\pi_{SL}$) tables, respectively.
- Moreover, as presented in [16], KSA is used with K_S and K_{uS} to produce the substitution (S) and update substitution (uS) tables.

Note that the sizes of the update primitives are identical to those of the corresponding cipher primitives.

The selection table π_{SL} is a permutation table of lp elements, where lp represents the size in bytes of the compressed data per packet. Hence the values of π_{SL} vary between 1 and lp. The size of a selected element depends on pr and the size of the compressed packet length, $l = ceil(pr \times lp)$. As such, only l elements are selected from each packet/tile, however, as the compression ratio increases, lp decreases, and l decreases for the same value of pr. The selected parts of the code-stream to encrypt and the size of the code-stream can be configured according to the MIoT application requirements.

On the other hand, the permutation table has l elements with values between 1 and l. The size of the substitution table is 255, instead of 256, since the 256^{th} element having the value 255 is removed to preserve the format compliant property. Thus, the substitution table and its corresponding update table contain each 255 elements.

Updating cipher primitives: For each part of the compressed data (packet, tile, or image), the cipher primitives (selection, permutation, and substitution tables) are updated to improve the security of the proposed cipher variants. Typically, regenerating the cipher primitives requires a high overhead. To simply the updating process, we propose a lightweight update based on a simple permutation; the permutation table π is updated by permuting its elements using $u\pi$. Similarly, S is updated via a permutation using uS, and Sl is updated using U_{SL}.

IV. Proposed Cipher Algorithm

The proposed cipher variants can be applied at any MIoT device and with any image compression algorithm. The notations used in this work are summarized in Table I.

Table I: Table of notations

Notation	Definition
SK	Secret Key
N_o	Nonce
DK	Dynamic Key
K_s	Substitution sub-key
π	Permutation table
π^{-1}	Inverse permutation table
$u\pi$	Update permutation table
S	Substitution table
S^{-1}	Inverse substitution table
uS	Update substitution table
SL	Selection table
uSL	Update selection table
p_i	i^{th} packet (compressed data), tile of the compressed image or any other data level
ep_i	i^{th} encrypted packet (compressed data), tile of the compressed image or any other data level by using the proposed cipher variants
N_i	Length of the i^{th} packet body(compressed data)/tile
h_i	Number of bytes selected from the i^{th} packet body/ tile to be encrypted
pr	Percentage of selected bytes from the compressed packets/tiles
lp	Packet length.
l	Length of selection table is dependent of pr, and it is equal to $l = ceil(pr \times lp)$ bytes length
K_p	Permutation sub-key used to generate a primary permutation table
K_s	Substitution sub-key used to generate a primary substitution table
K_{up}	Permutation sub-key used to generate an update permutation table
K_{SL}	Selection sub-key used to generate a primary selection table
K_{uSL}	Update selection sub-key used to generate an update selection table
K_{uS}	Update substitution sub-key used to generate an update substitution table

A. Proposed Selective Format Compliant Scheme

The proposed solution is based on selective encryption and it is used to protect the original visual contents of a compressed MIoT image code-stream, excluding the header/sub-header part. For example, in the case of the JPEG codecs family, all marker codes have hexadecimal values varying from (FF, 90) to (FF, FF), and should not appear in the encrypted/decrypted code-stream. Hence, the encrypted compressed data should vary between (00, 00) and (FF, 8F).

This is achieved by selecting a compressed byte with a value less than 0x(FF) along with its consecutive byte. The permutation process will change the order but not the value, which keeps the permuted compressed bytes between 0x(00) and 0x(FE). Also, the substitution process uses a table with values between 0x(00) and 0x(FE).

The scheme ensures that the produced ciphertext is compliant, without the need for any re-iteration mechanism, as presented in [39], or via a block cipher having a large number of rounds and using modulo addition such as AES [23].

B. Proposed cipher variants

The proposed cipher variants are shown in Figure 6 and Figure 7. The first variant requires only one operation, either a permutation or a substitution, while the second one requires both operations. Both variants could be applied onto a packet, a set of packets or a tile.

A compressed MIoT image, D, is divided into packets or tiles, depending on the compression capability. The scheme can process in parallel several packets, p_i, each with lp bytes, and where l bytes out of lp are selected for encryption. The value of l, and accordingly the percentage pr depend on the application and/or the sensitivity of image contents, as well as the available memory space. However, the minimum acceptable value of pr has to be 5 according to [37]. Similarly, in case the compressed input image is divided into tiles, the same steps are applied at the tile level.

First, the input packets are sorted based on their encryption parameters, and they are stored in a single buffer in ascending order $\{p_1,\ p_2,\ \ldots,\ p_n\}$, followed by the selection process, as shown in Figure 3. The bytes to be encrypted are selected using the table SL, and they form a vector to be either permuted or substituted in the case of the first cipher variant (see Figure 6). However, for the second variant (see Figure 7 and 8), the selected bytes are first permuted using π, and then substituted using S. The output values constitute the compressed ciphertext.

The cipher variants consists of two main functions, BytesSelection, and a RoundFunction (f) that consists of a single operation, either ByteSubstitution or BytePermutation for the first variant, and both operations for the second variant. Algorithm 1 summarizes the encryption scheme of the second variant.

Algorithm 1 The proposed second cipher variant with an input data part p_i)

procedure SECOND_VARIANT_ENCRYPTION($p_i,\ SL,\ \pi,\ S$)
  $temp \leftarrow Selection(p_i,\ SL)$
  $temp \leftarrow Permutation(temp,\ \pi)$
  $c_i \leftarrow Substitution(temp,\ S)$
  $ep_i[SL] \leftarrow c_i$
  return ep_i

After processing a packet or a set of packets, the cryptographic primitives are updated according to Algorithm 2. The update process is simple and uses only a permutation operation.

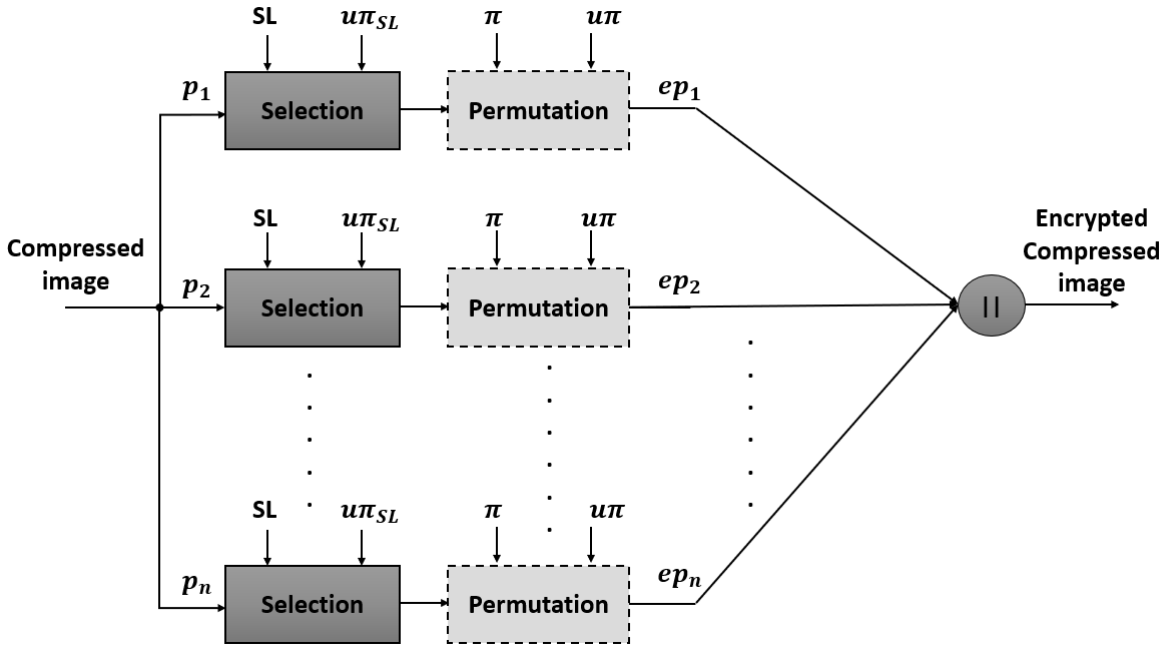


Figure 6: The first selective encryption variant that uses either a substitution or a permutation operation, and that can be applied at the packet or at any other data level.

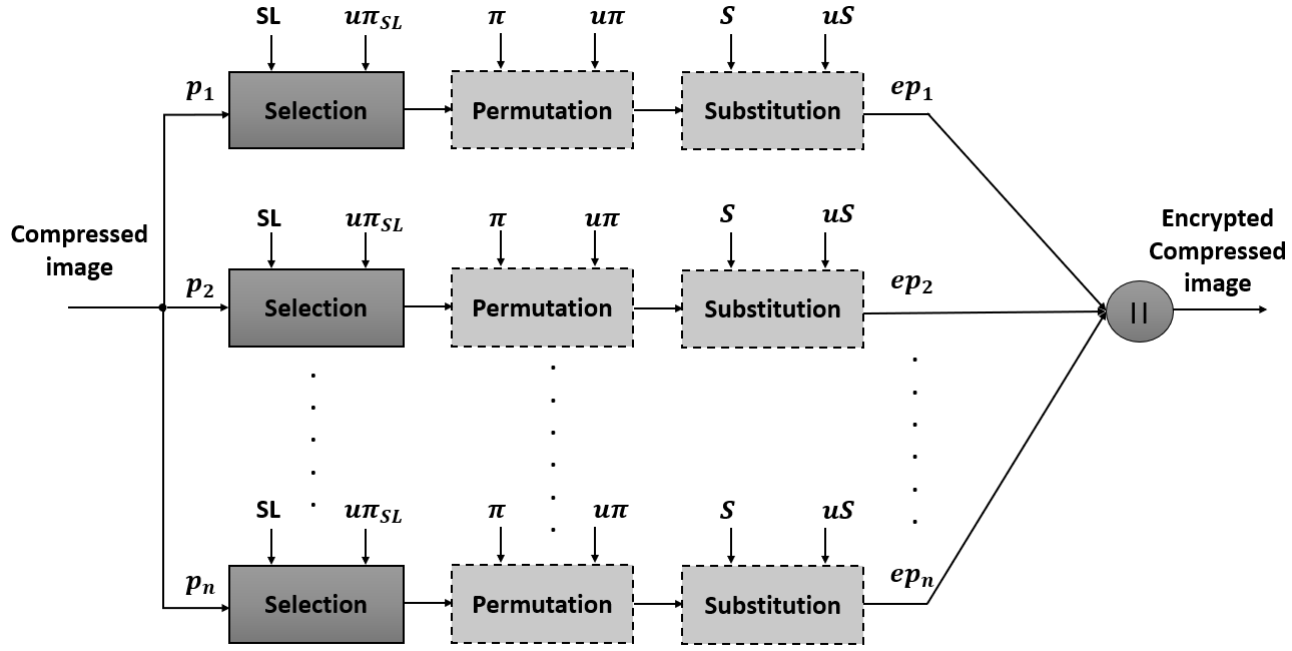


Figure 7: Architecture of the second cipher variant that can be applied at the packet or any other data level.

C. Decryption Algorithm

At the application server, the original image contents are recovered by applying the decryption algorithm on the selected encrypted packets, as illustrated in Figure 8 for the second cipher variant.

The receiver buffering model sorts the packets, in ascending order, according to their encryption parameters, and stores them in a single buffer. The decryption process, shown in Algorithm 3, for the second variant, is similar to encryption, but it is applied in reverse order and using the inverse substitution operation, followed by the inverse permutation. The inverse permutation and substitution tables are described in the following sub-sections.

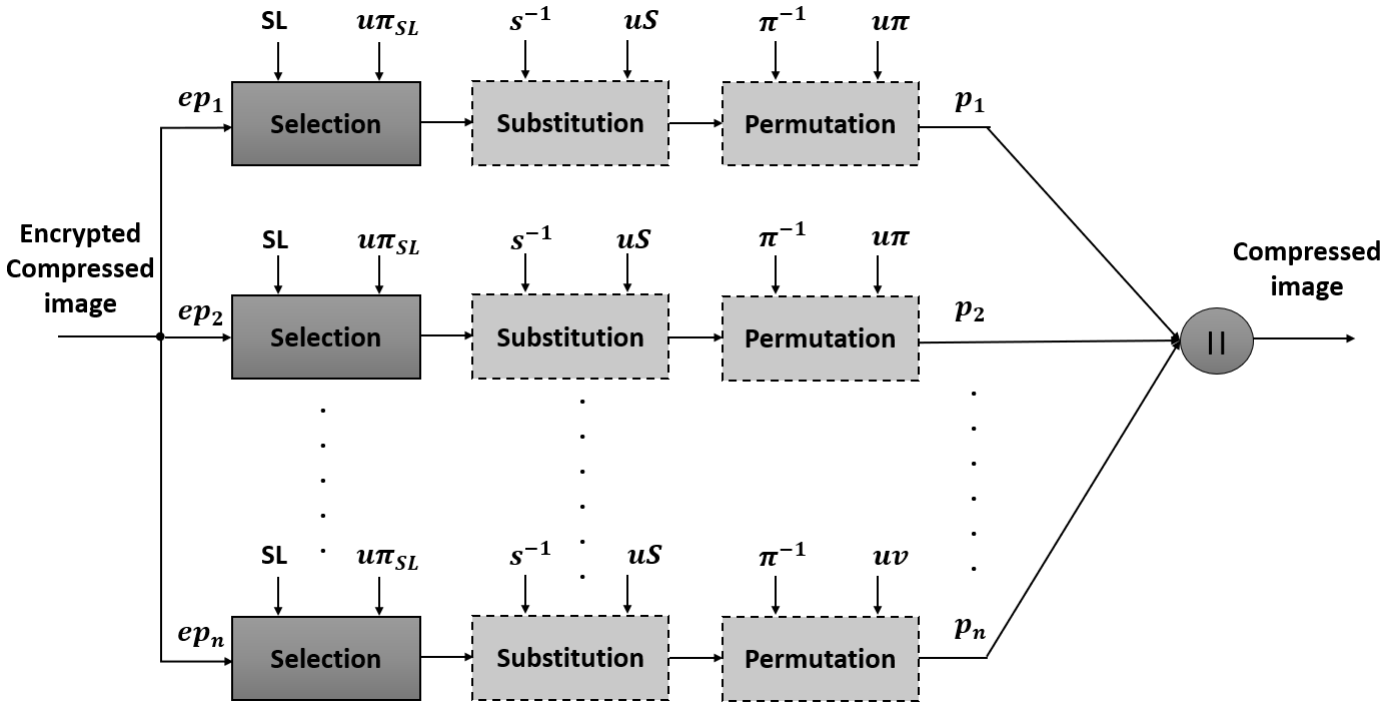


Figure 8: Architecture of the inverse cipher scheme of the second variant

Algorithm 2 The proposed Update cipher primitive algorithm.

procedure UP_CIPHER_PRIMITIVES(π, π_{up}, S, uS, SL, uSL)
  $\pi \leftarrow Permutation(\pi,\ \pi_{up})$
  $S \leftarrow Permutation(S,\ uS)$
  $SL \leftarrow Permutation(SL,\ uSL)$
  return (π, S and SL)
Output: Updated SL, S, π cipher primitives

Algorithm 3 The corresponding decryption algorithm of the proposed second variant)

procedure SECOND_VARIANT_DECRYPTION(ep_i, SL, π^{-1}, S^{-1})
  $temp \leftarrow Selection(ep_i,\ SL)$
  $temp \leftarrow Substitution(temp,\ S^{-1})$
  $d_i \leftarrow Permutation(temp,\ \pi^{-1})$
  $p_i[SL] \leftarrow d_i$
  return p_i

*1) **Inverse Byte Substitution** S^{-1}:* The substitution table S is bijective and its corresponding inverse substitution table S^{-1} can be obtained from S by applying the following equation:

$$S^{-1}[S[j]] = j, \text{where } j \text{ and } S(j) \in \{0, ..., 254\}. \quad (1)$$

*2) **Inverse Byte Permutation**:* The permutation table π is also bijective, and its corresponding inverse permutation table is obtained by using the following equation:

$$P^{-1}(P(j)) = j \ ; \text{ where } j \text{ and } P(j) \in \{1, \ldots, N\}. \quad (2)$$

V. SECURITY ANALYSIS

In this section, we analyze and evaluate the cryptographic properties of the proposed selective crypto-compression scheme, and the update process of cryptographic primitives. We assess cryptographic properties such as uniformity and randomness, in addition to the sensitivity of the cipher-text and the dynamic cipher primitives. These metrics are commonly used to quantify the cryptographic properties and security of new cipher algorithms [15], [17], [40], [16].

In the following experiments, the encryption is performed at the packet level, and we set the number of iterations to $1,000$. As a proof of concept, we apply the proposed scheme to JPEG and JPEG 2000 image compression algorithms. Note that, for other codecs, minor modifications are required such as the start of the compressed data, and the minimum acceptable percentage of data to encrypt.

A. Cryptographic Primitives Tests

To ensure a high level of security in the proposed cipher variants, the permutation and substitution tables should be changed pseudo-randomly and independently, as compared to future or previous ones. the tests of Recurrence and correlation coefficient ρ are used to assess the randomness of the permutation and substitution tables for different input packets [41], [42]. The tests were carried out on a set of $1,000$ random dynamic keys.

1) Recurrence Test: The recurrence test analyzes the variations in a data sequence to determine its randomness degree, and thus, it captures the correlation between the data sequence $x_i = x_{(i,1)},\ x_{(i,2)},\ x_{(i,3)},\ \ldots,\ x_{(i,m)}$, and another vector with a delay $t \geq 1$ given by $x_i(t) =$

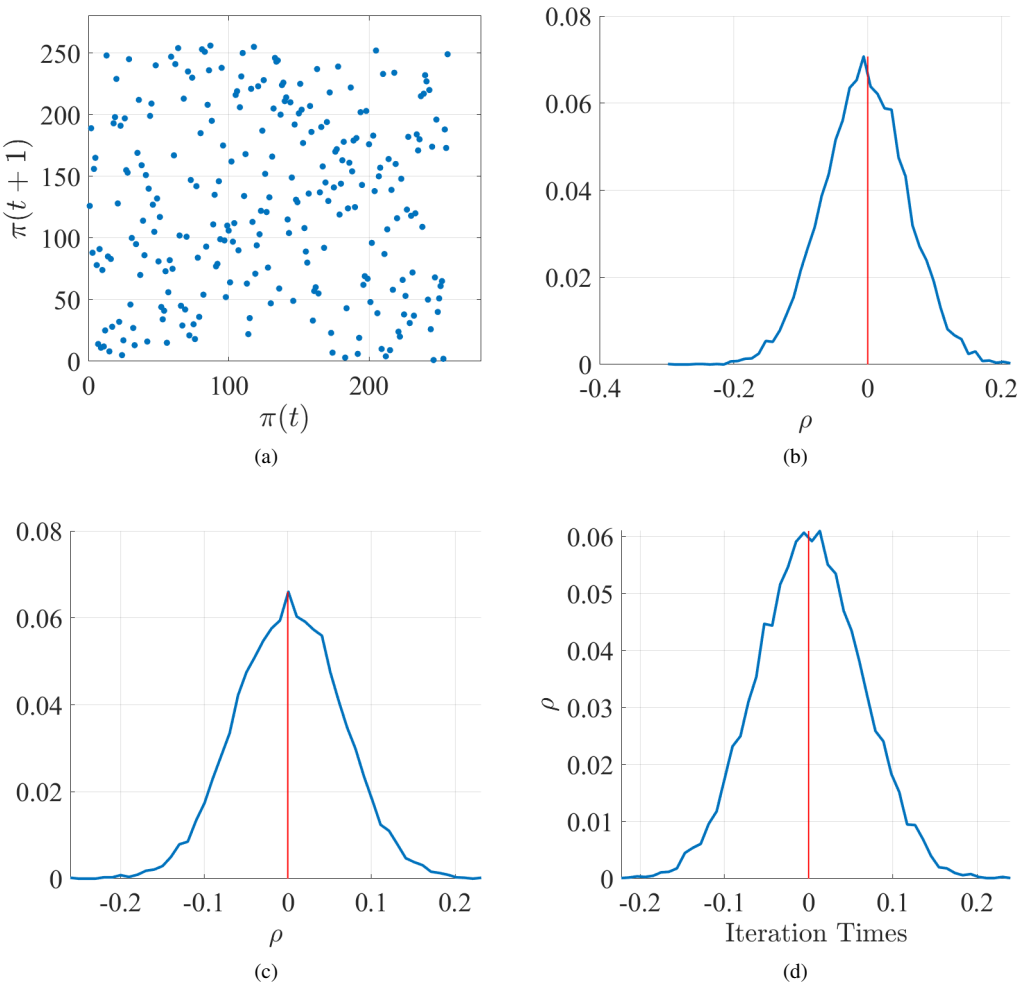


Figure 9: (a) The recurrence plot of a basic permutation table generated at random, (b) The correlation coefficient of recurrence for the generated permutation tables, (c) the correlation coefficient between a permutation table and its updated version, and (d) the correlation coefficient between two successive permutation tables, for 1,000 times.

$x_{(i,t)}$, $x_{(i,2t)}$, $x_{(i,3t)}$, $\ldots$, $x_{(i,mt)}$. The correlation coefficient r_{xy} between two vectors x and y is computed using the formula:

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(x) \times D(y)}} \tag{3}$$

where

$$E_x = \frac{1}{N} \times \sum_{i=1}^{N} x_i$$

$$D_x = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - E(x))^2$$

$$cov(x,y) = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - E(x))(y_i - E(y))$$

Figure 9-a shows the recurrence of the permutation table elements, generated using a random dynamic sub-key. The permutation table, as can be observed, has a random recurrence map and not a linear one.

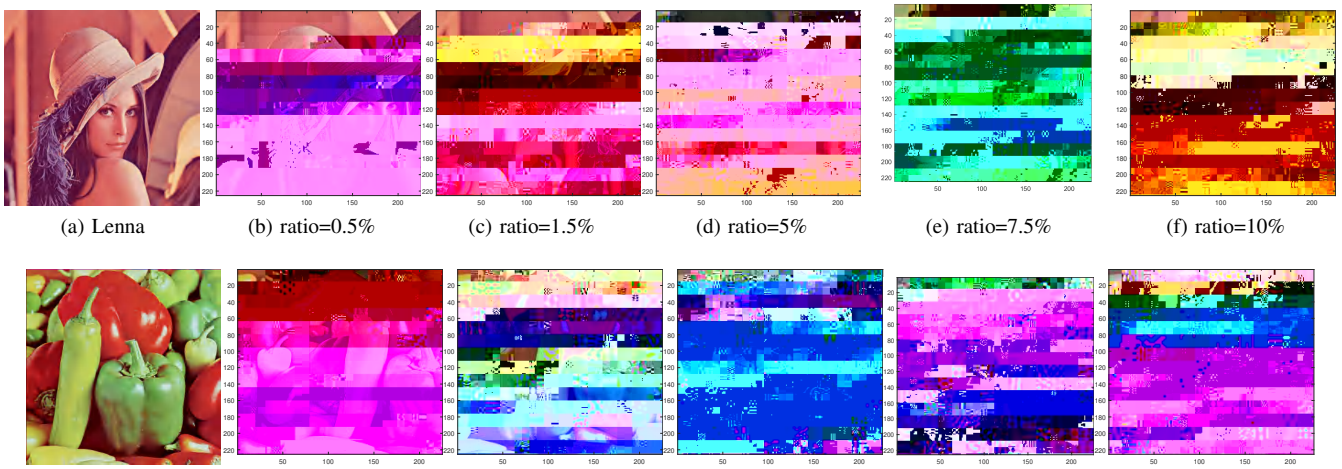
Furthermore, Figure 9-b displays the Probability Distribution (PD) of the correlation coefficients between the recurrence of the permuted index (of permutation table) for 1,000 random dynamic keys (for $l = 256$, the same length of the substitution table). Since the recurrence correlation is close to zero, with the majority of correlation values varying between $\{-0.2, 0.2\}$, the generated permutation tables, using the MKSA, exhibit a high degree of randomness.

Figure 9-c displays the PD of the correlation coefficients between the original P-box and the updated one as a function of the number of iterations. These obtained correlation coefficients vary between -0.15 and 0.15 (range close to zero). This indicates that the original and updated P-boxes are uncorrelated and independent.

Furthermore, the PD of the correlation coefficients between two successively updated P-boxes for 1,000 iterations is shown in Figure 9-d. The correlation coefficients approach

Table II: Statistical results for the proposed update (permutation) cipher primitives scheme for 1,000 iterations

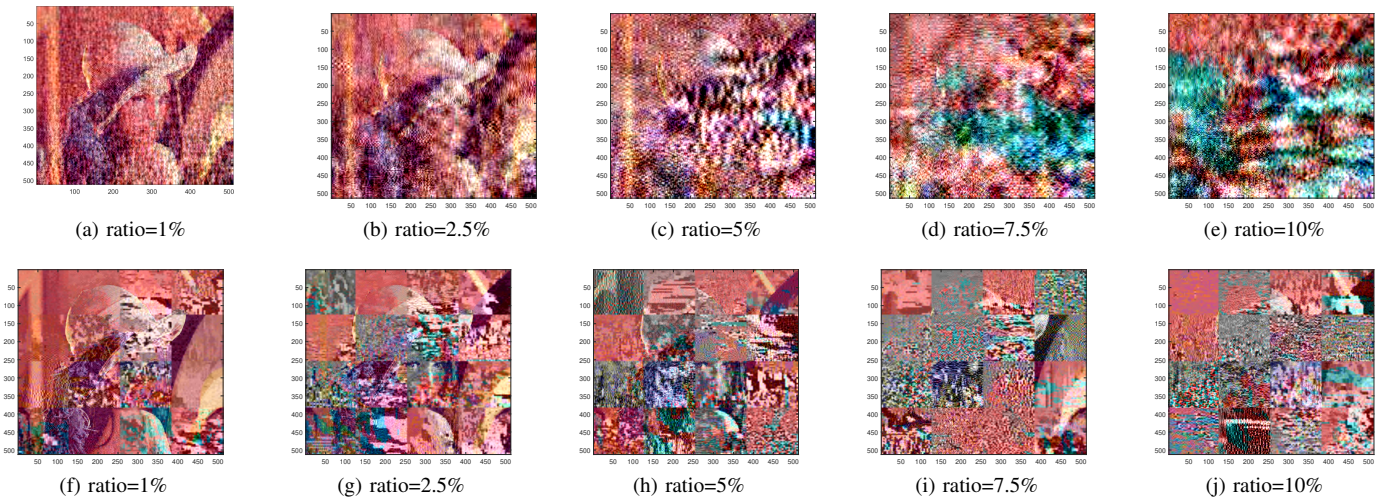
Coefficient Correlation Tests	**Min**	**Mean**	**Max**	**Standard Deviation**
ρ of the recurrence of produced dynamic permutation tables	-0.3040	-0.0041	0.2184	0.0628
ρ between the modified version (permuted version) of the primary permutation table and the original one	-0.2393	0.0009	0.2441	0.0622
ρ between two successive permutation tables	-0.2651	0.0017	0.2364	0.0631



(a) Lenna (b) ratio=0.5% (c) ratio=1.5% (d) ratio=5% (e) ratio=7.5% (f) ratio=10%

(g) Pepper (h) ratio=0.5% (i) ratio=1.5% (j) ratio=5% (k) ratio=7.5% (l) ratio=10%

Figure 10: Visual results of the proposed solution with the JPEG compression algorithm: Original compressed Lenna (a) and pepper (g), the corresponding encrypted images (b)-(f) and (h)-(l), respectively, for various encryption ratios



(a) ratio=1% (b) ratio=2.5% (c) ratio=5% (d) ratio=7.5% (e) ratio=10%

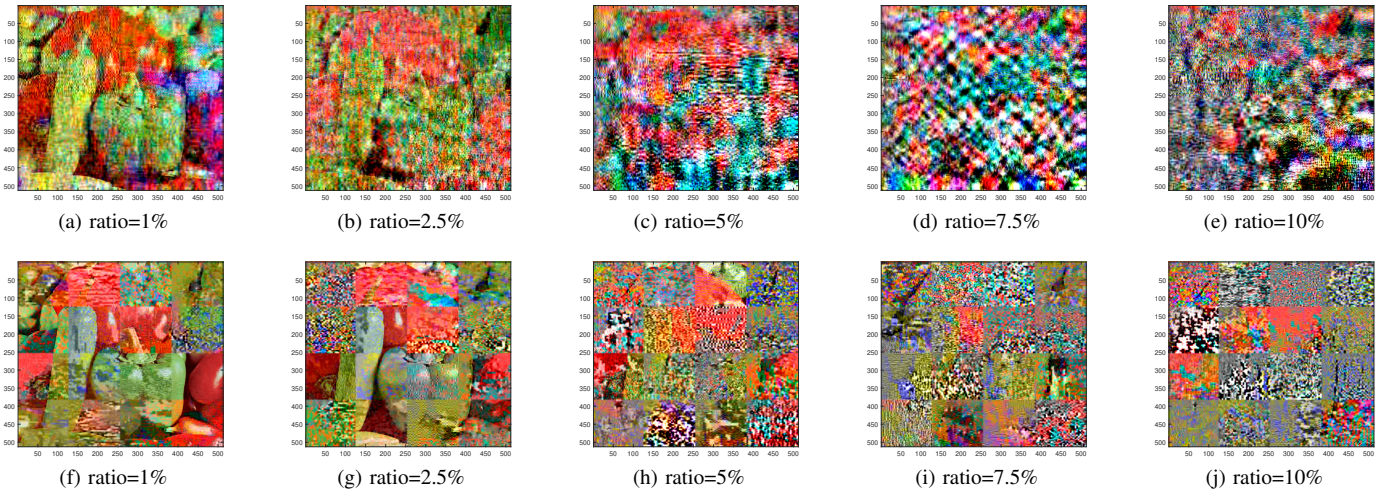
(f) ratio=1% (g) ratio=2.5% (h) ratio=5% (i) ratio=7.5% (j) ratio=10%

Figure 11: Encrypted compressed (JPEG 2000) Lenna image for one tile (a)-(e) and for 16 tiles (f)-(j), for different encryption ratios.

the ideal value of zero since they vary between $\{-0.2, 0.2\}$). This confirms that any two produced P-boxes are unrelated (independent). These results are also supported by the statistical analysis results presented in Table II. The standard deviation is around 0 for all three cases in the table, and hence, most correlation coefficient values are very close to the mean value of zero.

Since the generated permutation tables have high degrees of dynamicity and uniqueness, the proposed cipher variants can resist eavesdropping, prevent unauthorized users from obtaining any relevant information, and prevent cryptanalysis. The produced dynamic substitution boxes show similar results to the permutation boxes, with distinct and uncorrelated substitution tables.

The cryptographic primitives are updated frequently and pseudo-randomly, as shown in Algorithm 2. Thus, each input packet, tile or any other data unit, will be encrypted using a different set of selection tables, as well as different cryptographic primitives (permutation and/or substitution tables).



(a) ratio=1% (b) ratio=2.5% (c) ratio=5% (d) ratio=7.5% (e) ratio=10%

(f) ratio=1% (g) ratio=2.5% (h) ratio=5% (i) ratio=7.5% (j) ratio=10%

Figure 12: Encrypted compressed pepper image (JPEG 2000) for different encryption percentages, for one tile (a)-(e) and for 16 tiles (f)-(j).

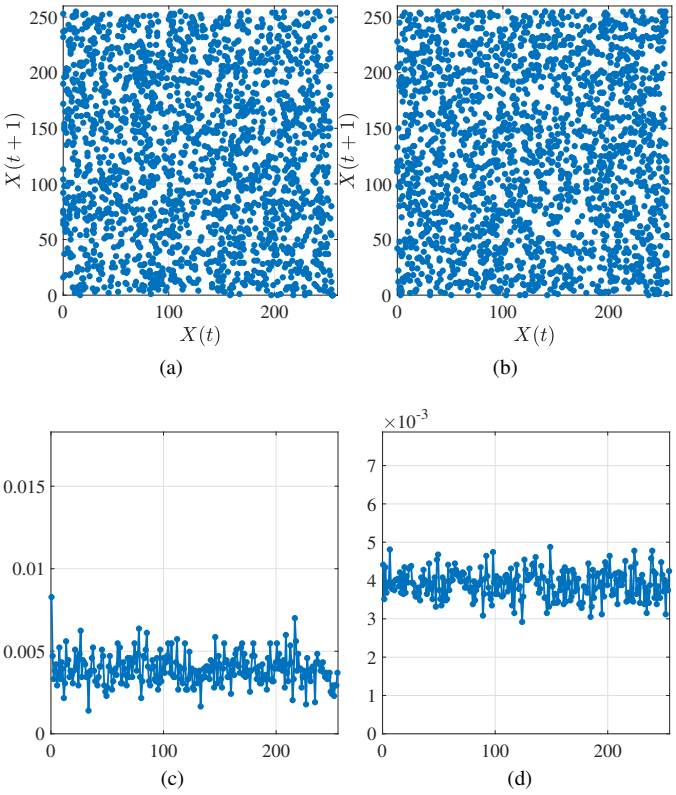


Figure 13: Recurrence plot of the compressed bytes for Lenna with (a) JPEG, and (b) JPEG 2000, and their corresponding PDF (c) and (d), respectively.

This obscures the original packet/tile and ensures that different ciphertexts will be obtained for the same original input, if encrypted again. This mitigates the threats posed by chosen-plaintext and chosen-ciphertext attacks.

B. Data Related Tests

To guard against statistical attacks, the encrypted compressed image should exhibit a high level of randomness and uniformity. Visual Results of the encrypted images with JPEG and JPEG 2000 image compression algorithms are presented in Figure 10 Figure 11, and Figure 12 for different encryption percentages. In fact, the compressed data exhibits already a certain degree of randomness and uniformity, which depends on the compression codec. The proposed variants leverage the compression characteristics to achieve efficiently a high security level, by relying on a single round. In the following, we detail the randomness and uniformity tests results.

1) Uniformity Test: The recurrence of the compressed bytes (code-stream) of the compressed Lenna image with JPEG and JPEG 2000 are shown in Figure 13-(a) and (b), respectively. Moreover, the distribution of the compressed bytes of Lenna with JPEG and JPEG 2000 are shown in Figure 13-(c) and (d), respectively. The results confirm that the compressed bytes with JPEG and JPEG 2000 have a uniform distribution and a random recurrence. Hence, with a single round and one operation, the first cipher variant achieves the desirable statistical properties, after the compression process.

The first variant uses the permutation operation to change the order of the selected bytes, and hence, the same distribution (uniform) is preserved. Figure 13 shows that the random recurrence property is also preserved. Similarly, the first variant using substitution instead of permutation, also preserves the uniformity and randomness properties. Also, the second variant preserves these properties since it is based on substitution and permutation operations.

In fact, the uniformity and the randomness properties of the compressed data must be preserved to prevent statistical attacks. This condition is achieved with the proposed cipher variants. For the first variant, the permutation-encryption scheme preserves the values with random order, which indi-

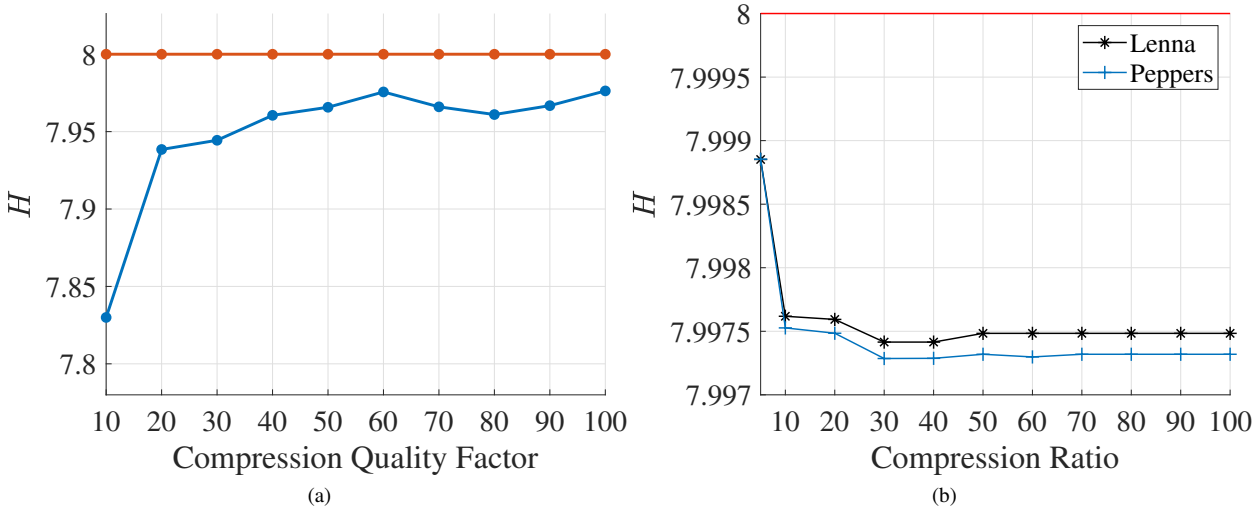


Figure 14: Variation of the entropy values of compressed Lenna for (a) JPEG quality factors, and (b) JPEG 2000 compression ratio.

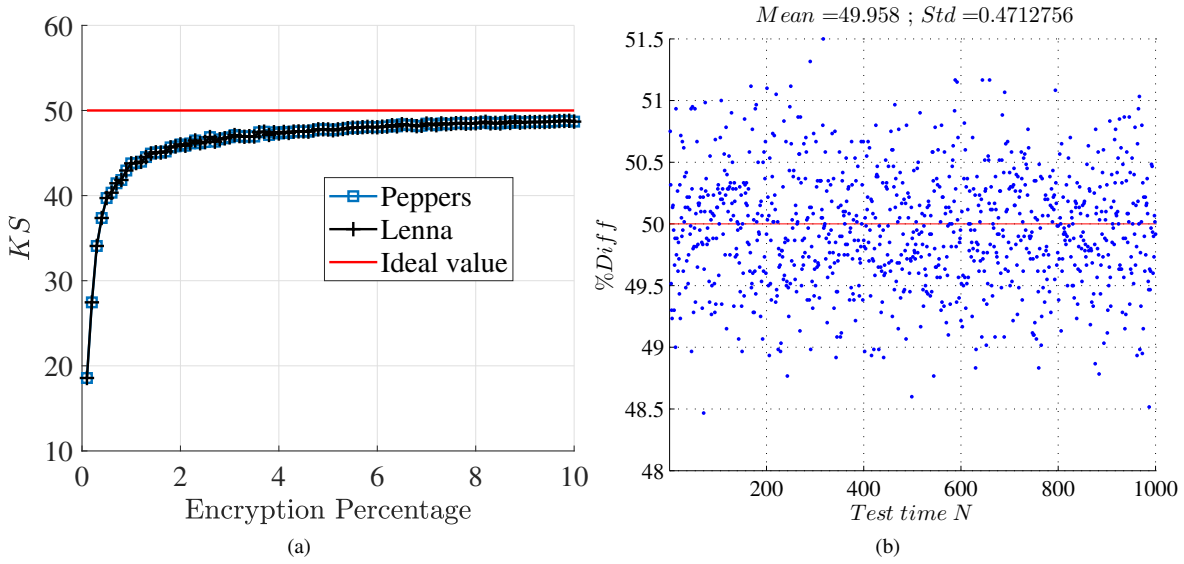


Figure 15: Percentage bit difference between original and encrypted decompressed images versus encryption percentage, for 1,000 random keys with (a) JPEG 2000 codec and (b) the corresponding values for encryption percentage of 5%(b).

cates that the contents of the code-stream and its corresponding encrypted stream still have a uniform distribution and random recurrence.

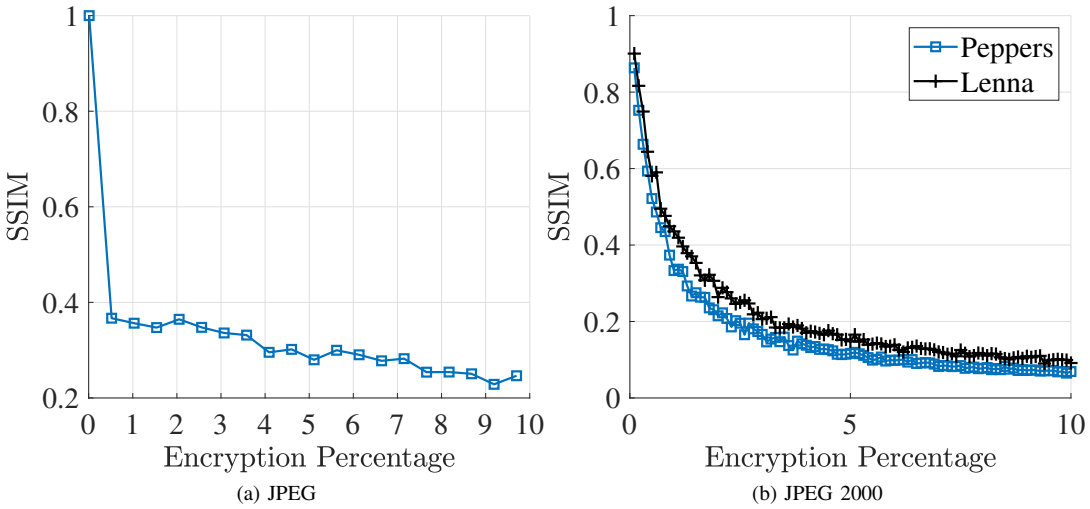
In Figure 13-(c) and (d), it can be observed that both histograms are very close to a uniform distribution.

We apply another test to verify the uniformity at the block level, by applying the information entropy concept as shown below:

$$H(S) = \sum_{i=0}^{k-1} p(s_i) \times log_2(p(s_i)) \quad (4)$$

Where k is the number of levels (here 256), and s_i is the $i-th$ symbol (here, it varies between 0 and 254 since the 255 value is eliminated). The obtained values for the information entropy are very close to 8, as shown in Figure 14, which is the maximum value of the entropy $log_2(256) = 8$, for a uniform distribution. These results illustrate the variation of the entropy values, which were measured for the compressed bytes of the Lenna image with JPEG (a) versus the image quality (inverse of the compression ratio), in addition to the entropy values that were measured for compressed data with JPEG 2000 codec (b) versus the compression ratio. The entropy values for both codecs are always close to the desired value of 8. This confirms that the compressed data, for both image compression algorithms, have a sufficient level of uniformity.

Hence, the compressed data behaves as being generated from random sources, and having a uniform distribution. Consequently, this test ensures that the proposed encryption process



(a) JPEG (b) JPEG 2000

Figure 16: Variation of the SSIM mean versus encryption ratio percentage for (a) JPEG and (b) JPEG 2000.

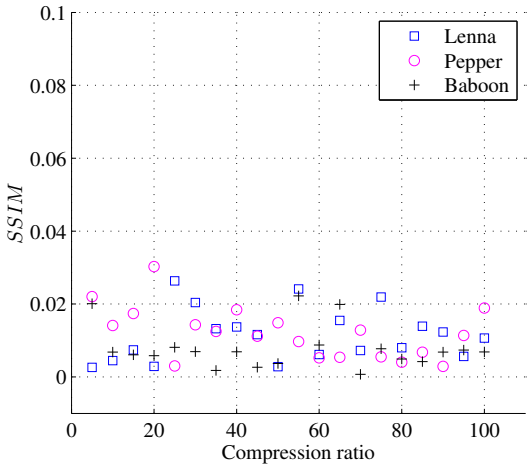


Figure 17: SSIM mean variation between the encrypted and original decompressed images (JPEG 2000) versus the compression ratio, for 1,000 random dynamic keys and encryption percentage of 6%.

does not impact the uniformity property of the compressed data. Therefore, the encryption process, with the first or second variant, does not introduce any degradation to the randomness and uniformity properties.

2) Independence Test: The encrypted image should exhibit a high independence level from the original image. In order to quantify this, the difference percentage between the original and encrypted decompressed images, at the bit level, should be close to 50%. Figure 15-a) shows the difference between the original and encrypted images (decompressed Lenna and pepper images with JPEG 2000) versus the encryption percentage pr. Also, the same difference tests were applied for $1,000$ random keys with encryption percentage equals to 5%, and the obtained results are shown in Figure 15-b). Similar results are obtained with the original JPEG, which confirm that the cipher schemes (first or second) achieve the required independence property between the original and encrypted decompressed images.

3) Visual Degradation: Visual degradation quantifies the visual distortion between the original decompressed image and its corresponding encrypted one. The structural similarity (SSIM) criteria is widely used as a metric for this criterion [43]. Thus, a high visual degradation can be achieved with JPEG and JPEG 2000 codecs for encryption percentage above 5% since the SSIM becomes very low, close to zero. The results, illustrated in Figure 16-a) and b), clearly show a hard visual degradation for JPEG and JPEG 2000. In Figure 17, we show the mean variation of SSIM between encrypted and original decompressed standard images (JPEG 2000) versus the compression ratio, for 1,000 random dynamic keys, and for encryption percentage of 6%. The results show that when encrypting a small percentage of the compressed image, and for any compression ratio, the proposed scheme produces hard visual degradation.

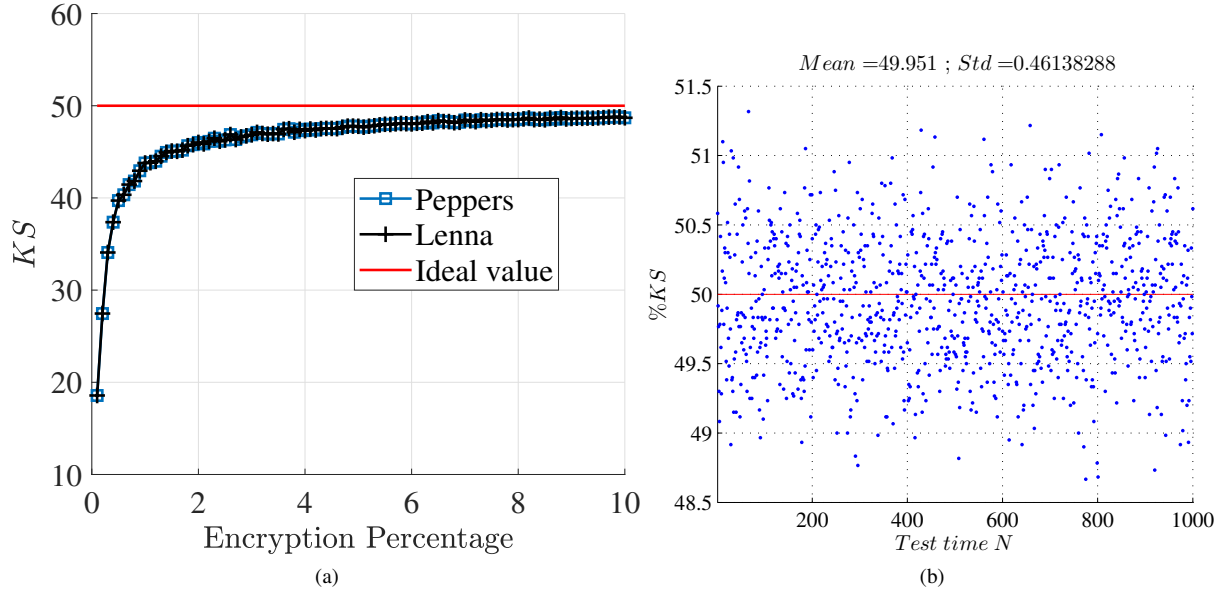
C. Sensitivity Tests

The sensitivity tests aim to validate the avalanche effect of the plaintext and key. These tests measure the difference percentages between the resulting encrypted images, when one bit differs in the original image, or in the secret or dynamic key; the desired value is a 50% difference at the bit level.

1) Key Sensitivity: This test measures the change in the ciphertext, after flipping one bit in the secret key SK, or the nonce. In the proposed scheme, such a single bit change results into a different dynamic key, and thus, different cipher and update cipher primitives (substitution and permutation tables). For 1,000 random keys, the sensitivity of the dynamic key DK is computed using the percent Hamming distance:

$$KS_w = \frac{\sum_{k=1}^{Tb} C_w \oplus C'_w \oplus}{T} \times 100\% \quad (5)$$

$$= \frac{\sum_{k=1}^{T} E_{DK_w}(P) \oplus E_{DK'_w}(P)}{T} \times 100\%$$



(a) (b)

Figure 18: Key sensitivity over 1,000 random dynamic keys: (a) percentage variation versus percentage encryption, and (b) the corresponding values for encryption percentage of 6%.

The encrypted images, corresponding to DK_w and DK'_w, are C_w and C'_w, respectively. Note that all bits of DK'_w are equal to those of DK_w, except for a single bit. Actually, SK and N_o ensure the key sensitivity since they are mixed and hashed to produce the dynamic key, and then, the hash function ensures the avalanche effect.

Figure 18-a) shows the sensitivity of the dynamic key versus 1,000 random keys by using the JPEG 2000 compression versus the encryption ratio for the Lenna image. The results show that the proposed scheme satisfies the desired key sensitivity, for an encryption percentage above 5%. Moreover, Figure 18-b) shows that the majority of the dynamic key sensitivity samples, for an encryption ratio of 5%, are close to the optimal value of 50% at the bit level. Table III shows their minimum, maximum, average, and standard deviation. As such, the dynamic key approach achieves a higher resistance against chosen/known plain-text/cipher-text attacks. Similar results were obtained with the JPEG compression.

Table III: Statistical results of the key sensitivity and entropy tests for 1000 iterations.

	$\%KS$	H
min	48.4	3.5
max	51.5	4
Avg	50.015	3.9425
STD	0.062	0.0824

2) Plain-text Sensitivity Test: In the proposed cipher, the dynamic key changes for each input compressed image. Moreover, the cipher primitives are updated for each input packet or tile. As a result, the same compressed image is encrypted using different cipher primitives, resulting in different encrypted compressed images with a difference close to 50%, as seen in Figure 18. This confirms the plain-text sensitivity (avalanche effect) of the dynamic key approach.

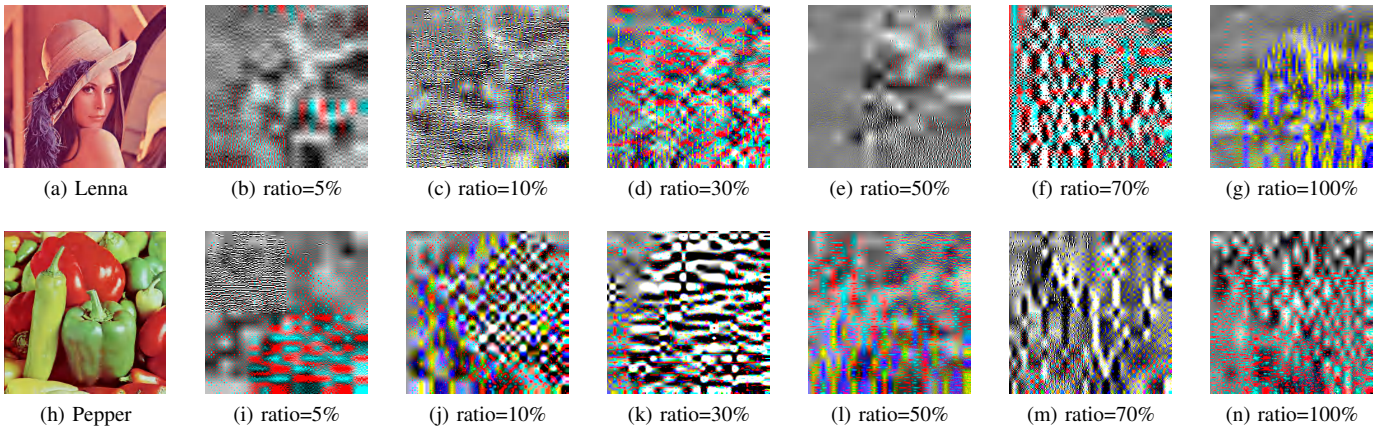
VI. Encryption Ratio Threshold

In this section, we present the optimal minimum encryption percentage (rate) that ensures a hard visual degradation; this minimizes the computations and the resources overhead for MIoT devices. We consider the original standard images of Lenna and pepper, compressed with JPEG and JPEG 2000. The corresponding encrypted images versus different encryption percentages are shown in Figure 10 and Figure 11, respectively. On the other hand, the results of the compressed Lenna and pepper, with an encryption percentage of 100%, versus the compression ratio are shown in Figure 19.
According to the obtained results of SSIM and Peak Signal-to-Noise Ratio (PSNR) in Figure 16-a) and b), a minimum of 5% of the compressed bytes (selected in a pseudo-random manner) is required for the JPEG and JPEG 2000 compressed images to protect their visual contents.

VII. Cryptanalysis

The proposed cipher variants follow the dynamic key-dependent structure; the cipher primitives can be updated for each packet, tile, or input image, depending on the configuration. The permutation and substitution operations are performed in a variable manner, where the compressed data has a random order if the permutation operation is used, or modified values in case of substitution. This leads to an important visual degradation due to the decompression effect. Also, the compressed images exhibit good uniformity and randomness levels as seen in Figure 13, 14, and 15. Hence, statistical attacks on the ciphertext are unable to retrieve any relevant information.

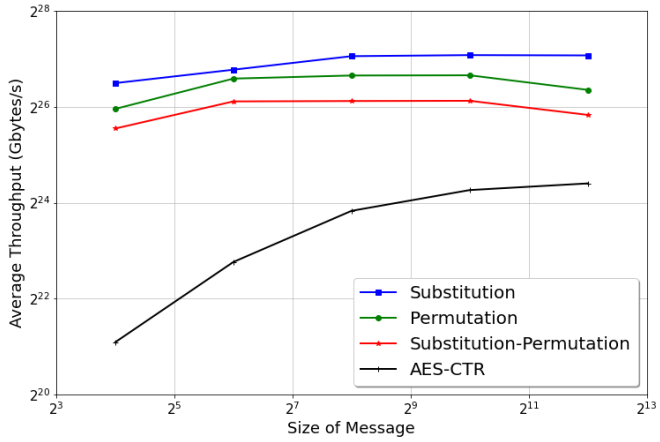
Moreover, according to Figure 15, the independence of the original and encrypted compressed images has been verified.



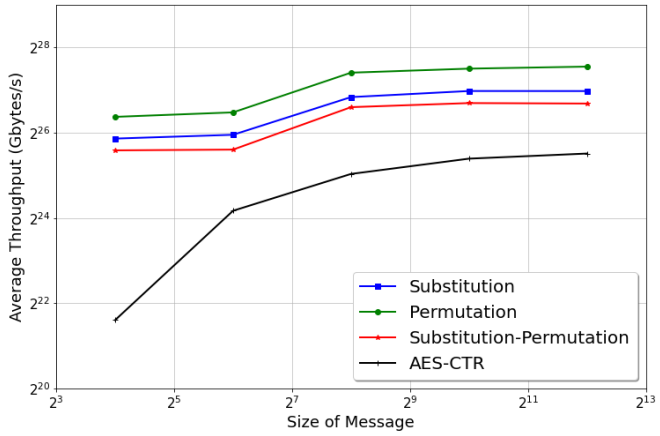
(a) Lenna (b) ratio=5% (c) ratio=10% (d) ratio=30% (e) ratio=50% (f) ratio=70% (g) ratio=100%

(h) Pepper (i) ratio=5% (j) ratio=10% (k) ratio=30% (l) ratio=50% (m) ratio=70% (n) ratio=100%

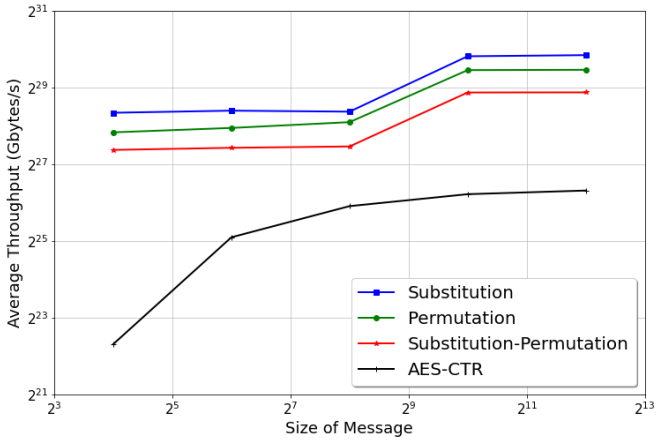
Figure 19: Full encryption results: Original Lenna (a) and pepper (h) and their corresponding encrypted decompressed images with JPEG 2000 (b)-(g) and (i)-(n), respectively, versus different compression ratios, where all compressed data are encrypted.



(a) RPI0



(b) RPI3



(c) RPI4

Figure 20: Throughput variation (MB/s) versus message size (in bytes) for the proposed cipher variants in function of plaintext length (bytes) on: (a) RPI0, (b) RPI3, and (c) RPI4.

The proposed cipher scheme applies different substitution and/or permutation tables for an input compressed image. The use of the update cipher primitives complicates the relationship between original and encrypted compressed images. According to Figure 18, the secret key sensitivity is satisfied, as well as the strong resistance to collision, with the probability of producing the same dynamic being $\frac{1}{2^{512}}$, which makes key-related attacks extremely difficult to execute. Also, differential attacks are not feasible since the dynamic cryptographic primitives are modified for each input packet, tile, or image, resulting into a high level of unpredictability, uniformity, and key sensitivity.

Note that analytic attacks are designed to break ciphers with static primitives, which is not the case with the proposed cipher variants. Finally, the secret key of the proposed cipher variants can have a size of 128, 196, or 256 bits, while both the nonce and dynamic key have a size 512 bits, which is sufficiently large to make brute-force attacks impractical.

Thus, we conclude that the proposed cipher variants provide a high security level with low overhead for MIoT devices; they require a single round and one operation for the first variant, and two operations for the second variant.

VIII. Performance Analysis

In this section, the performance of the proposed crypto-compression variants is analyzed towards quantifying their effectiveness. Two important metrics, the effect of error propagation and the associated computational complexity and throughput, are discussed.

A. Error Propagation

Error propagation is considered as one of the most important performance criteria for any practical cipher [16], [15]. Technically, a one-bit error, on the receiving side, is the flipping of a bit from '0' to '1', or vice-versa. The proposed cipher variants exhibit a low rate of error propagation since the bit error(s) in the decrypted compressed data occur at the same bit position(s) as in the erroneous ciphertext, in the case of permutation and substitution, which covers the three variants. Hence, the cipher variants restrict the errors to their corresponding bytes, and prevent their propagation to other bytes.

B. Computational Delay

In this section, we assess the computational delay of the proposed cipher variants. Note that the objective of the proposed scheme is to provide a high degree of security with the least possible number of operations and rounds, which in turn decreases the associated delay. To that end, we define several delay components to compute the total delay:

1) T_S denotes the time required for the byte substitution of a block of h bytes.
2) T_D represents the delay of the AES diffusion mix-column operation (for all 4 columns), which is the highest among all AES operations.
3) T_{xor} denotes the time required to perform logical XOR between two blocks of h bytes.
4) T_{SR} represents the delay for the AES permutation "Shift-rows" operation, and r represents the number of rounds.

The total computational delay to encrypt a block of 16 bytes, with the AES standard [36], is:

$$CD_{AES} = rT_S + (r+1)T_{xor} + (r-1)T_D + rT_{SR} \quad (6)$$

The minimum value of r in AES is 10 for a 128-bit secret key. Hence, the minimum AES computational delay is given by:

$$CD_{AES(r=10)} = 10T_S + 11T_{xor} + 9T_D + 10T_{SR} \quad (7)$$

The total Computational Delay (CD) of the proposed scheme to encrypt one block for the first and the second variant are:

$$CD_{V1} = T_S \text{ or } T_\pi \quad (8)$$

$$CD_{V2} = T_S + T_\pi \quad (9)$$

where T_π denotes the required time to permute an input block of h bytes.

On the other hand, the Delays (CD) associated with the update process, for the first and second variants, are:

$$CD_{UpdateV1} = T_\pi \quad (10)$$

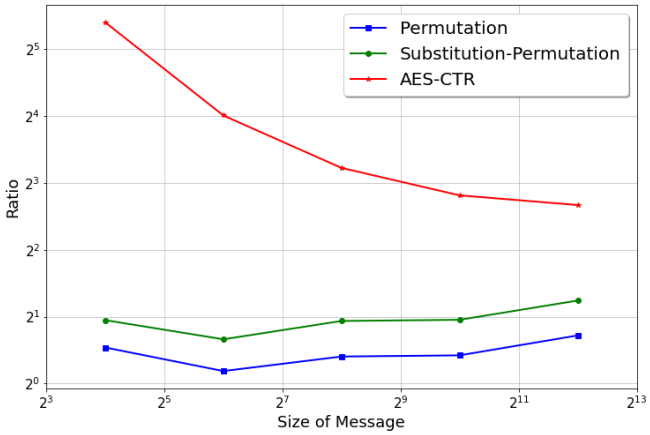
$$CD_{UpdateV2} = 2 \times T_\pi \quad (11)$$

C. Experimental Throughput

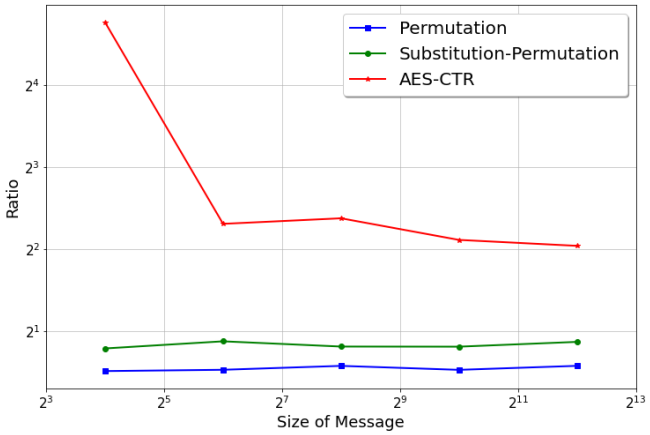
The execution time is a very important factor, especially for tiny devices when huge amounts of data are transmitted in multimedia real-time applications. The existing cryptographic algorithms, such as AES, exhibit a large execution time due to the relatively large number of rounds when compared to the proposed scheme.

Theoretically, the first proposed cipher variant requires only one operation (substitution or permutation), while two simple operations are required for the second variant. An additional overhead is associated with the update process, which consists of a single permutation operation. This overhead is minimized when the update process is applied for each new input image.

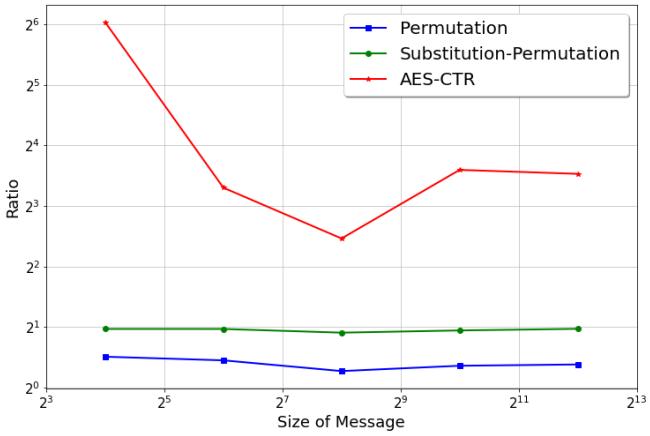
We implemented, using the C language, the cipher variants on different Raspberry Pi device classes (RPI0, RPI3, RPI4), and for different sizes of the plaintext. The experiments are repeated for 10,000 times, and the average throughput results are compared against AES (OpenSSL implementation with CTR operation mode), as shown in Figure 20. Note that the AES OpenSSL implementation code is well optimized in assembly language. The results show that the proposed cipher variants outperform AES on all RPIs. The numerical results of the throughput ratio are shown in Table IV, for the different Raspberry Pi classes. The best performance over AES is achieved with RPI0. The performance is slightly reduced with RPI3 and RPI4 due to the optimized OpenSSL implementation in assembly. In summary, considering RPI0, the throughput of the proposed cipher variants is higher compared to AES, and a significant gain can be achieved, in the case of RPI3 and RPI4, by optimizing the cipher variants using assembly as in AES.



(a) RPI0



(b) RPI3



(c) RPI4

Figure 21: Throughput ratio of the cipher variant with substitution process versus message size (in bytes) against the other proposed variants (Permutation, Substitution-Permutation) and AES-CTR on: (a) RPI0, (b) RPI3, and (c) RPI4.

Table IV: Statistical throughput Ratio results of the proposed cipher variants compared to the optimized OpenSSL AES-CTR.

	Cipher variant	*Min*	*Mean*	*Max*	*Std*
RPI0	First variant with substitution	6.36	16.24	42.33	13.49
	First variant with permutation	3.86	11.9	29.15	9.32
	Second variant	2.69	8.67	21.95	7.12
RPI3	First variant with substitution	4.11	9.17	27.27	9.05
	First variant with permutation	2.76	6.37	19.15	6.39
	Second variant	2.25	5.24	15.81	5.29
RPI4	First variant with substitution	5.51	20.89	65.43	22.38
	First variant with permutation	4.56	15.19	45.93	15.45
	Second variant	2.93	10.71	33.43	11.415

On the other hand, the first variant exhibits a better throughput compared to the second variant, and the first variant with substitution achieves the maximum throughout, and thus, the minimum delay.

Figure 21 shows the ratio of the throughput results of the first variant with substitution operation compared to the other variants and AES-CTR, for the different Raspberry Pi classes. the results confirm that AES-CTR requires a larger execution time than the proposed cipher variants.

In the case of RPI0, the first variant with substitution achieves a throughput enhancement of at least 536% as compared to AES, 58.14% compared to the second variant, and 13.7% compared to the first variant with permutation. As for RPI4, the enhancement is at least 451% as compared to AES, 87% compared to the second variant, and 42.39% compared to the first variant with permutation.

Table V: Energy consumption results of proposed solutions versus AES for message length of 4096 bytes

Hardware	A	Volts	Algorithm	Energy consumption (W/S)	Ratio compared to V1 (Substitution)
RPI0	0.18	5	First variant with Substitution	2.611e-05	-
			First variant with permutation	4.297e-05	1.645
			Second variant	6.17e-05	2.36
			AES-CTR	1.66e-04	6.35
RPI3	0.375	5	First variant with Substitution	3.899e-05	-
			First variant with permutation	5.803e-05	1.488
			Second variant	7.109e-05	1.82
			AES-CTR	1.61e-04	4.13
RPI4	0.61	5	First variant with Substitution	1.29e-05	-
			First variant with permutation	1.68e-05	1.30
			Second variant	2.53e-05	1.96
			AES-CTR	1.49e-04	11.55

On the other hand, the energy consumption of the proposed variants was compared to the one of AES-CTR in Table V, for different RPI devices (RPI0, RPI3, RPI4). The results show that the energy consumption of proposed solutions is less than that of AES-CTR, and for the different RPI devices, which make them more suitable for limited MIoT devices. Also, the results show that the second variant requires more energy consumption than the first variant with either substitution or permutation, which is expected since the second variant includes the two operations. The first variant, with the substitution operation, requires lower energy consumption compared to the first variant with the permutation operation.

Based on these results, the first variant is recommended for very limited MIoT devices. when compared to AES-CTR, the energy consumption is reduced by approximately 75 % when using substitution, and 65% when using permutation. On the other hand, the second variant reduces the energy consumption by approximately 55% as compared to AES-CTR.

The energy consumption results can be further confirmed based on the execution time (throughput) results, which show that second variant requires more execution time, which translates into additional energy consumption compared to the first variant.

Note that there is a trade-off between the performance and compression ratio. The energy consumption could be further reduced by increasing the compression ratio, which reduces the size of the data to be encrypted and communicated. However, this would require the application of a denoising technique at the application server, to reduce the visual degradation introduced by the compression algorithm. On the other hand, the size of the input image affects the compression execution time, compressed data size, and as such, the required energy consumption. Therefore, reducing the size of the input image (down-sampling) would reduce the compression/decompression time, communication delay and the energy consumption.

In summary, the results show that the first variant with substitution is better suited for constrained devices and for real-time applications. The second variant provides a higher security level at the cost of an additional delay and energy consumption.

Note that a parallel implementation of the proposed variants enhances further the throughput, and reduces the associated delay. When increasing the number of threads, the execution time decreases at the expense of additional memory consumption, which is not an issue for the application server(s).

IX. Conclusion

In contrast to the existing selective cipher schemes that are applied after a compression operation, the proposed solutions benefit from the randomness and uniformity level of the compressed image data. They adopt the dynamic cryptographic approach to provide a simple, yet highly secure, cipher design that meets the requirements of limited multimedia IoT devices, and the stringent quality requirements of real-time applications. The proposed cipher variants require a single round and a maximum of two operations, for the second variant (permutation and substitutions). A dynamic key is generated for each input and used to produce a set of cryptographic primitives and the corresponding update primitives. The update process can be configured to be applied at different levels such as a packet, tile, or input image; it is done via a simple permutation operation. Furthermore, a set of the compressed data (packets or tiles) is selected based on a dynamic selection table. The compression ratio and encryption data rate are defined according to the requirements of the MIoT application and/or desired security level. Both cipher variants are efficient, and the produced cipher-texts satisfy the uniformity and randomness properties. The experimental results proved that the proposed solutions, with one round and one or two operations, achieve the data confidentiality security service between MIoT devices and application server(s) with fewer computations and resources compared to the existing solutions that require multiple rounds.

Acknowledgment

This work has been funded by the EIPHI Graduate School (contract "ANR-17-EURE-0002").

References

[1] Shalini Sharma Goel, Anubhav Goel, Mohit Kumar, and Germán Moltó. A review of internet of things: qualifying technologies and boundless horizon. *Journal of Reliable Intelligent Environments*, pages 1–11, 2021.

[2] Qin Wang, Yanxiao Zhao, Wei Wang, Daniel Minoli, Kazem Sohraby, Hongbo Zhu, and Ben Occhiogrosso. Multimedia iot systems and applications. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2017.

[3] Jean-Paul A Yaacoub, Mohamad Noura, Hassan N Noura, Ola Salman, Elias Yaacoub, Raphaël Couturier, and Ali Chehab. Securing internet of medical things systems: limitations, issues and recommendations. *Future Generation Computer Systems*, 105:581–606, 2020.

[4] Hamad Naeem, Farhan Ullah, Muhammad Rashid Naeem, Shehzad Khalid, Danish Vasan, Sohail Jabbar, and Saqib Saeed. Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. *Ad Hoc Networks*, 105:102154, 2020.

[5] Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. *Multimedia big data computing for IoT applications: Concepts, paradigms and solutions*, volume 163. Springer, 2019.

[6] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. Multimedia internet of things: A comprehensive survey. *IEEE Access*, 8:8202–8250, 2020.

[7] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[8] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, 2000.

[9] Howard Cheng and Xiaobo Li. Partial encryption of compressed images and videos. *Signal Processing, IEEE Transactions on*, 48(8):2439–2451, 2000.

[10] William Stallings. *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, NJ, 2017.

[11] Kerry A McKay, Larry Bassham, Meltem Sönmez Turan, and Nicky Mouha. Report on lightweight cryptography. *NIST DRAFT NISTIR*, 8114, 2016.

[12] Axel York Poschmann. Lightweight cryptography: cryptographic engineering for a pervasive world. In *PH. D. THESIS*. Citeseer, 2009.

[13] Hassan Noura, Lama Sleem, Mohamad Noura, Mohammad M. Mansour, Ali Chehab, and Raphaël Couturier. A new efficient lightweight and secure image cipher scheme. *Multimedia Tools and Applications*, Sep 2017.

[14] H. Noura and D. Courousse. Method of encryption with dynamic diffusion and confusion layers, June 9 2016. WO Patent App. PCT/EP2015/078,372.

[15] Hassan Noura, Ali Chehab, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. Lightweight, Dynamic and Efficient Image Encryption Scheme. *Multimedia Tools and Applications*, pages 1–35, 2018.

[16] Hassan Noura, Ali Chehab, Lama Sleem, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. One round cipher algorithm for multimedia iot devices. *Multimedia tools and applications*, 77(14):18383–18413, 2018.

[17] Hassan N Noura, Mohamad Noura, Ali Chehab, Mohammad M Mansour, and Raphaël Couturier. Efficient and Secure Cipher Scheme for Multimedia Contents. *Multimedia Tools and Applications*, pages 1–30, 2018.

[18] Hassan N Noura, Ali Chehab, and Raphaël Couturier. Overview of efficient symmetric cryptography: dynamic vs static approaches. In *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6. IEEE, 2020.

[19] Hassan N Noura, Ola Salman, Nesrine Kaaniche, Nicolas Sklavos, Ali Chehab, and Raphaël Couturier. Tresc: Towards redesigning existing symmetric ciphers. *Microprocessors and Microsystems*, page 103478, 2020.

[20] Junhui He, Shuhao Huang, Shaohua Tang, and Jiwu Huang. Jpeg image encryption with improved format compatibility and file size preservation. *IEEE Transactions on Multimedia*, 20(10):2645–2658, 2018.

[21] Marco Grangetto, Enrico Magli, and Gabriella Olmo. Multimedia selective encryption by means of randomized arithmetic coding. *Multimedia, IEEE Transactions on*, 8(5):905–917, 2006.

[22] Jiang-Lung Liu. Efficient selective encryption for jpeg 2000 images using private initial table. *Pattern Recognition*, 39(8):1509–1517, 2006.

[23] Ayoub Massoudi, Frédéric Lefebvre, Christophe De Vleeschouwer, and François-Olivier Devaux. Secure and low cost selective encryption for jpeg2000. In *ISM*, pages 31–38, 2008.

[24] Hassan N. Noura, Ola Salman, Raphaël Couturier, and Ali Chehab. Lorca: Lightweight round block and stream cipher algorithms for iov systems. *Vehicular Communications*, page 100416, 2021.

[25] Hassan Noura, Raphaël Couturier, Congduc Pham, and Ali Chehab. Lightweight stream cipher scheme for resource-constrained iot devices. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2019.

[26] Hassan N Noura, Ali Chehab, and Raphael Couturier. Efficient & secure cipher scheme with dynamic key-dependent mode of operation. *Signal processing: Image communication*, 78:448–464, 2019.

[27] Tao Xiang, Chenyun Yu, and Fei Chen. Secure mq coder: An efficient way to protect jpeg 2000 images in wireless multimedia sensor networks. *Signal Processing: Image Communication*, 29(9):1015–1027, 2014.

[28] Jeffrey Ting, KokSheik Wong, and Simying Ong. Format-compliant perceptual encryption method for jpeg xt. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4559–4563. IEEE, 2019.

[29] Dominik Engel and Andreas Uhl. Secret wavelet packet decompositions for jpeg 2000 lightweight encryption. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006.

[30] Dominik Engel and Andreas Uhl. An evaluation of lightweight jpeg2000 encryption with anisotropic wavelet packets. In *Electronic Imaging 2007*, pages 65051S–65051S. International Society for Optics and Photonics, 2007.

[31] SimYing Ong, KokSheik Wong, Xiaojun Qi, and Kiyoshi Tanaka. Beyond format-compliant encryption for jpeg image. *Signal Processing: Image Communication*, 31:47–60, 2015.

[32] Dominik Engel, Thomas Stütz, and Andreas Uhl. Assessing jpeg2000 encryption with key-dependent wavelet packets. *EURASIP Journal on Information Security*, 2012(1):1–16, 2012.

[33] Hongjun Wu and Di Ma. Efficient and secure encryption schemes for jpeg2000. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–869. IEEE, 2004.

[34] Yongdong Wu and Robert H Deng. Compliant encryption of jpeg2000 codestreams. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3439–3442. IEEE, 2004.

[35] Osamu Watanabe, Akiko Nakazaki, and Hitoshi Kiya. A scalable encryption method allowing backward compatibility with jpeg2000 images. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 6324–6327. IEEE, 2005.

[36] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[37] Zeinab Fawaz, Hassan Noura, and Ahmed Mostefaoui. Securing jpeg-2000 images in constrained environments: a dynamic approach. *Multimedia Systems*, 24(6):669–694, Nov 2018.

[38] Med Karim Abdmouleh, Hedi Amri, Ali Khalfallah, and Med Salim Bouhlel. A fast jpeg2000 based crypto-compression algorithm: Application to the security for transmission of medical images. In *International workshop soft computing applications*, pages 164–175. Springer, 2016.

[39] Yongdong Wu and R.-H. Deng. Compliant encryption of jpeg2000 codestreams. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 5, pages 3439–3442 Vol. 5, Oct 2004.

[40] Hassan Noura, Lama Sleem, Mohamad Noura, Mohammad M Mansour, Ali Chehab, and Raphaël Couturier. A New Efficient Lightweight and Secure Image Cipher Scheme. *Multimedia Tools and Applications*, 77(12):15457–15484, 2018.

[41] Hassan N. Noura, Reem Melki, Mohammad Malli, and Ali Chehab. Design and realization of efficient & secure multi-homed systems based on random linear network coding. *Computer Networks*, 163:106886, 2019.

[42] Hassan Noura, Steven Martin, Khaldoun Al Agha, and Khaled Chahine. Erss-rlnc: Efficient and robust secure scheme for random linear network coding. *Computer Networks*, 75:99 – 112, 2014.

[43] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402 Vol.2, Nov 2003.