An efficient secure image encryption algorithm based on total shuffling, integer chaotic maps and median filter

Eihab Bashier[1,3*] and Taoufik Ben Jabeur[2]
[1]Sohar University, Sohar, Sultanate of Oman.
ebashier@su.edu.om
[2]Dhofar University, Salalah, Sultanate of Oman
tjabeur@du.edu.om
[3]University of Khartoum, Khartoum, Sudan.
eihabbashier@gmail.com

Received: February 21, 2021; Accepted: May 16, 2021; Published: May 31, 2021

Abstract

Developing a secure image encryption algorithm that can be implemented in devices with different hardware structures has remained a problematic issue, particularly for the real time applications. In this paper, we introduce two integer versions of chaotic maps, namely the quadratic chaotic map used to generate the parameters of the system and an integer version of Chebyshev chaotic maps used to generate permutations and chaotic sequences of integers. Then, we propose an image encryption algorithm that applies many rounds, each containing confusion and diffusion step in which a permutation is used to shuffle the pixels' locations and the chaotic sequence is used to change the pixels' densities. The proposed method is tested against many security measurements and some kinds of attacks for five standard images. It shows better performance over recent methods from the literature. The paper moreover discusses the performance of the decryption algorithm against the salt & pepper and crop attacks on the cipher image. It is found that the decipher image is seriously affected by high levels of noise or crop attacks. To overcome the limitations of the decryption algorithm at high levels of attacks, we proved first that the crop attack has same distribution as the salt & pepper noise attack. Then, an enhancement technique based on the use of median filter is proposed to reduce the effect of the attacks. It is shown from simulations that the enhancement decryption algorithm can double the image quality in terms of the PSNR.

Keywords: Chaotic maps, Crop attack, Image encryption, Key sensitivity, Median filter, Permutations, Salt and pepper attack

1 Introduction

The development of science and technology in the recent years has facilitated the process of transferring data between devices in various forms, including files that contain texts, audios, images, and videos in high speed. Also, the digital revolution has provided many solutions and benefits; and at the same time, it provided the risks and damage in several areas such as security and privacy [4, 22].

In the press areas, images represent half of the news, and no one can imagine newspapers or magazines free of comic material cartoons, graphics, and illustrative figures. Images are also used in the military field, such as recognizing classified security zones through images taken from Google Earth and

Journal of Internet Services and Information Security (JISIS), volume: 11, number: 2, pp. 46-77
DOI:*10.22667/JISIS.2021.05.31.046*

*Corresponding author: Faculty of Education and Arts, Sohar University, P.O. Box: 44, Postal Code: 311, Sohar, Oman, Tel: +968-9207-9850

satellites. Images of criminals are also exchanged between the competent authorities, making it easier to identify and reach them.

In many applications, maintaining the confidentiality of images is of great necessity to preserve the security of the community and facilities. Given the increasing number of cybercrime in the modern world, it was necessary to devise methods to protect digital images, especially during transmission on the Internet. Image encryption is considered of the most practical way to keep it confidential [8, 19, 7].

Image encryption is usually faced by many challenges that affect its efficiency and speed. Some of these challenges include the high correlation between its pixels, which require algorithms different from those used in text encryption. Another challenge is that the data size of an image is large compared to the text, and the process of encrypting images is more expensive than for text, hence it requires high performance and the use of more than one channel to reach the receiver quickly [20, 11]. Protecting the image against crypto-analysis attacks or perceptual analysis is one of the biggest challenges [24, 1].

An image encryption method is usually composed of multiple rounds each including two crucial steps:

- **confusion step:** the locations of image pixels are exchanged, without changing the pixels' densities. At the end of this stage, the perceptual features of the image may disappear and instead appears a blurry image that cannot be distinguished by an eye. But many statistical properties of the image remain unchanged such as the image histogram, information entropy etc., leaving it under the risk of many kinds of statistical analysis techniques [22].

- **diffusion steps:** in this step, the densities of the image pixels are changed using an encryption key. The purpose of this step is to change the statistical properties of the image, such as the pixels correlation and image histogram analysis, so that the encryption algorithm can resist against the statistical encryption methods. An efficient image diffusion is one that causes the image's pixel densities to be uniformly distributed [2].

Image encryption techniques based on chaotic maps have emerged in recent years as of the most important encryption methods. This is due to the practical characteristics of the chaotic maps, such as their high sensitivity to small changes in the parameters of the model or its initial values [25, 26, 29]. Both the sender and receiver are able to generate the same sequence of random numbers if they start from the same initial point and use the same parameters of the map.

In the literature, the chaotic maps used in image encryption are mostly defined in real intervals and thus are characterized by broad range of key selection space. However, the decryption algorithm might fail if the sender and receiver use devices with different floating points representation. On the other hand, integers representation in different devices does not affect the decryption process when the representation of large integers is allowed. Therefore, the use of chaotic maps defined on an integer domain helps to develop cryptography algorithms for different terminal devices [3, 14].

Examples to chaos-based encryption algorithms from the literature include the following. In [27], an affine modular map acting on the unit interval is used to establish a chaos-based image encryption scheme. Two dimensional permutation is constructed with two affine modular map to shuffle the image pixels. Then, other two affine modular maps were used to produce pseudo-random sequences of gray values for the diffusion of the image. The authors showed that the key space for their algorithm is large enough to resist against the brute-force attack. Also, they showed that their algorithm is resistant against statistical, differential, known plain-text and chosen plain text attacks. In [9], a high-dimensional chaotic system and cycle operation for DNA sequences produced by Chen's hyper-chaos are used to construct an image encryption scheme. Within the proposed scheme, DNA coding rule were used to encode the plain image pixels. The authors proved that their scheme is robust against brute-force, statistical and differential attacks. Ghebleh et. al [6] proposed an image encryption algorithm based on chaotic maps

and least squares approximations. It consists of two phases: a shuffling phase and a masking phase based on one–dimensional piecewise linear chaotic maps. These two phases are applied in many rounds. The authors stated that the simulation results of their algorithm is robust against common statistical and security attacks. Yu et. al [28] proposed an spatio-temporal chaos based image encryption algorithm that combines DNA insertion and deletion operations to encrypt two images simultaneously. Their design to the keystream is such that the plain keystream is related to the plain image, and the confused image is related to another confused image through the DNA insertion and deletion. The authors showed that their algorithm has a large key space and high sensitivities to changes in key and plain image. It also has a high information entropy and a good resistance to crop occlusion and noise attacks. An other approach of image encryption was presented by Tang et. al [23] that exploited jointly random overlapping block partition, double spiral scans, and Henon and Lü chaotic maps. The authors provided many experimental results including the correlation, histogram and entropy analyses, and they also illustrated the performance of their algorithm under the differential, cropping and noise attacks. Karawia [12] presented an encryption algorithm using a 3D chaotic map and logical operator bit-wise XOR, with a key space of 10^{182} for both coloured and grey scale images, with either 256×256 or 512×512 pixels. In his simulation results, many measurements were considered including statistical and differential analysis attacks. The author concluded that the proposed method has high sensitivity to the security keys. Based on simultaneous permutation and diffusion operation (SPDO), Liu et. al [16] presented an image encryption scheme. In their scheme, a pixel value is permuted and diffused simultaneously by row and column using a sine-sine map. They measured the performance of their scheme against statistical, differential and chosen plain text attacks. They compared the performance of their method to existing methods that are based on the SPDO. In [30], an image encryption based on S-boxes and fractional-order logistic map is proposed. The authors justified their choice of the fractional-order logistic map to their findings that it has larger key space and parameters compared to the classical logistic map. The authors stated that their key space is bigger than 2^{383}. Different measures are used to evaluate the performance of their algorithm including the histogram, correlation and entropy analyses, differential and ciphertext only attacks, and the NIST measures. Liu et. al [17] proposed a region of interest (ROI) based multidimensional chaotic image encryption algorithm. In their algorithm, the ROI is separated from the whole image and then the pixels of the ROI are confused by using Henon and Joseph sequences and diffused by using a unified chaotic sequence. Their algorithm is tested against many kinds of analyses including key sensitivity, statistical, information entropy and quality analyses.

Although all the above mentioned image encryption methods are reported to have high performance measures, but they generally have some drawbacks. Some of the methods put in consideration only the key space, statistical analysis techniques, broad key space, and robustness against differential attacks, but have not considered crop or noise attacks. Also, almost these methods used chaotic maps defined over real intervals, hence practically the decryption algorithms are expected to fail if devices with different hardware structures are used. Finally, all these methods fail to obtain high decipher image quality in the presence of high intensity crop or noise attacks.

This paper presents the concepts of generating integer chaotic maps using truncation from Taylor series on a multiplicative group $\mathbb{Z}_p^*$ with p a prime. It also presents an algorithm that uses an integer chaotic map to generate pseudo-random permutations. Based on these two concepts, an efficient image encryption algorithm containing a number of rounds is introduced. Each round contains a confusion and a diffusion step. In the confusion step, a random permutation is used to shuffle the positions of image pixels. Then, in the diffusion step, pseudo-random sequences of integers generated by a chaotic map are added to the transformed image pixels to change their densities. The performance of the proposed algorithm must be comparable to the ones appeared in the literature that are using real domain chaotic maps. Also, the proposed method aims to be resistant to different kinds of attacks.

The main contributions of the paper are as follows:

(i) Provide new kinds of chaotic maps defined over finite multiplicative groups of prime orders.

(ii) Provide a quick and easy algorithm for generating pseudo-random permutations based on integer chaotic maps and cyclic finite groups.

(iii) Show that the crop attack has the same pixel distribution as the salt and pepper noise attack of the same power in the decipher image.

(iv) Propose an enhancement algorithm to improve the quality of decipher image based on median filters.

Several standard measures are used to quantify the performance of the proposed encryption/decryption algorithm. The key secrecy and key sensitivity are used to evaluate the performance of the proposed key scheme. Pixels correlations, information entropy, image histogram and chi-square test are used to measure the performance of the encryption algorithm. In practice, the encrypted image could be corrupted by various attacks, including the salt & pepper noise attack and crop attack. To measure the robustness of the decryption algorithm under presence of attacks, the mean squares errors (MSE), peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) between the plain and decipher images are used.

The rest of this paper is organized as follows. Section 2 discusses the proposed image encryption/decryption algorithms. The performance of the proposed method, discussions of results and comparisons with methods from the literature are presented in sections 3 and 5, respectively. Section 6 is dedicated for conclusions.

2 Proposed image encryption algorithm

In this section, we present the image encryption algorithm, which is based on shuffling the image pixels and adding random numbers generated by chaotic maps in several number of rounds.

The most crucial components of the encryption and decryption processes are two integer versions of chaotic maps. The first chaotic map is an integer version of the quadratic map. It is used to generate system parameters for both the encryption and decryption processes. The second chaotic map is an integer version of the Chebyshev map, and it is employed to generate sequences of pseudo-random numbers for the diffusion process. Hence, before running the encryption or decryption algorithm, the sender and receiver must agree about these system parameters.

We assume that the sender and receiver use a key agreement protocol to agree about public system integer parameters. These parameters are $nRounds \in \mathbb{Z}^+$, $A \in \mathbb{Z}^*_{P_q}$, $x_0^q \in \mathbb{Z}^*_{P_q}$, $x_0^c \in \mathbb{Z}^*_{P_c}$ and $N_c \in \mathbb{Z}^*_{P_c}$, where P_q and P_c are prime numbers.

The flowchart of the encryption algorithm is shown in Figure 1. As soon as the sender and receiver agree about the global system parameters, each party runs an initialization algorithm that contains four important steps.

(i) Convert the plain/cipher image to one-dimensional signal from $\mathbb{Z}_{256}$,

(ii) Compute the other system parameters that are necessary for the encryption/decryption algorithms,

(iii) Generate *nRounds* permutations

(iv) Generate *nRounds* chaotic sequences using the integer Chebyshev chaotic map.

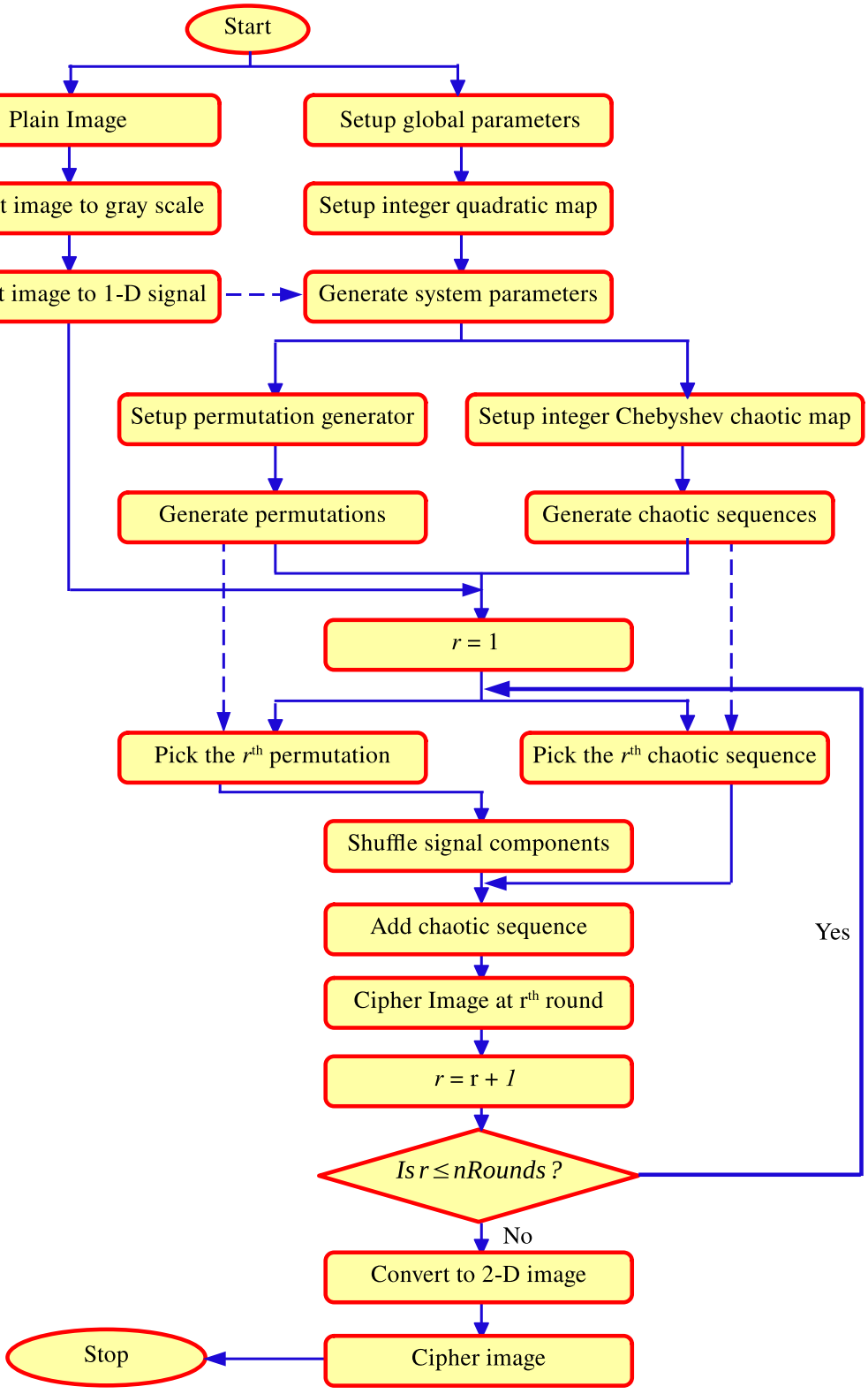


Figure 1: Flow chart of the encryption algorithm.

Algorithm 1 The Encryption algorithm executed at the side of sender.

procedure RUNENCRYPTION(Img, A, x_0^q, P_q, x_0^c, N_c, P_c, nRounds)
  Img ← GrayScale(Img)
  {Img1D, Perms, PermShifts, ChaoticSeqs, Rows, Cols} ← EncInitialize(Img, A, x_0^q, P_q, x_0^c, N_c, P_c, nRounds)
  CImg ← Img1D
  for Round ← 1 **to** nRounds **do**
    CImg ← Perms(Round, CImg)
    CImg ← CImg + ChaoticSeqs(Round) mod 256
  end for
  CipherImg ← ReshapeImage(CImg, Rows, Cols)
  return CipherImg
end procedure

In order to encrypt the image, the sender runs Algorithm 1 which receives an input image *Img* and returns an output image *CipherImg*.

Then, the sender sends the encrypted image *CipherImg* to the second party, who runs the decryption algorithm (Algorithm 2) to retrieve the plain image *DecipherImg*.

Algorithm 2 The decryption algorithm executed at the side of sender.

procedure RUNDECRYPTION(Img, A, x_0^q, P_q, x_0^c, N_c, P_c, nRounds)
  {CImg1D, invPerms, invChaoticSeqs, Rows, Cols} ← DecInitialize(A, x_0^q, P_q, x_0^c, N_c, P_c, NumPerms, nRounds)
  DImg ← CImg1D
  for Round ← 1 **to** nRounds **do**
    DImg ← DImg + invChaoticSeqs(nRounds − Round + 1) mod 256
    DImg ← invPerms(nRounds − Round + 1, DImg)
  end for
  for r ← 1 **to** Rows **do**
    for c ← 1 **to** Cols **do**
      DecipherImg[r, c] ← DImg[r(Rows − 1) + c]
    end for
  end for
  return DecipherImg
end procedure

The initialization of encryption/decryption algorithms is described in the next section.

2.1 Initialization algorithm for encryption/decryption processes

The initialization algorithm has the following tasks:

(i) An image *Img* of dimensions $Rows \times Cols$ is converted into one-dimensional signal $Img1D$ with length *ImgDim* using the relation

$$Img1D[r(Rows-1)+c] = Img[r,c], r = 1,\ldots,Rows, c = 1,\ldots,Cols$$

(ii) The quadratic chaotic map is used to generate parameters required for the generation of permutations and chaotic sequences.

(iii) Find the closest prime number $P \geq ImgDim$, such that $Q = (P-1)/2$ is also prime.

(iv) Generate *nRounds* permutations that will be used in the confusion step for shuffling the signal components.

(v) Generate *nRounds* chaotic sequences that will be used in the diffusion steps for changing values of the signal components.

The sender and receiver have almost the same initialization functions, except that the initialization algorithm at the sender side (`EncInitialize`) calls functions `GeneratePermutations` and `GenChebChaoticSeq` to generate permutations and chaotic sequences of same signal length. The initialization algorithm at the receiver side `DecInitialize` calls functions `GenerateInversePermutations` and `GenerateInverseChaoticSeqs` to generate inverse permutations and chaotic sequences.

A pseudo-code of the function `EncInitialize` and `DecInitialize` are described by Algorithms 3 and 4, respectively.

Algorithm 3 Initializing one-dimensional image signal, *nRound* permutations and *nRound* sequences of pseudo-random numbers for the image encryption process.

procedure ENCINITIALIZE(Img, a, x_0, nRounds)
  {Rows, Cols} = size(Img)
  ImgDim ← Rows × Cols
  Img1D ← Array(1, ImgDim)
  for i ← 1 **to** Rows **do**
    for j ← 1 **to** Cols **do**
      Img1D[i · (Rows − 1) + j] ← Img[i, j]
    end for
  end for
  {P_r, i2, i4, i6, PermParams, CMParams} ← GETSYSPARAMS(ImgDim, P_c, A, x_0^q, P_q, m, nRounds)
  Permutations ← GENERATEPERMUTATIONS(PermParams, PermShifts, nRounds, ImgDim, P)
  return {Img1D, Permutations, CMSeqs, m, n}
end procedure

Algorithm 4 Initializing one-dimensional image signal, *nRound* permutations and *nRound* sequences of pseudo-random numbers for the image decryption process.

procedure DECINITIALIZE(CipherImg, a, x_0, nRounds)
  {Rows, Cols} = size(CipherImg)
  ImgDim ← Rows × Cols
  CImg1D ← Array(1, ImgDim)
  for i ← 1 **to** Rows **do**
    for j ← 1 **to** Cols **do**
      CImg1D[i · (Rows − 1) + j] ← Img[i, j]
    end for
  end for
  {P_r, i2, i4, i6, PermParams, PermShifts, CMParams} ← GETSYSPARAMS(ImgDim, P_c, A, x_0^q, P_q, nRounds)
  invPermutations ← GENERATEINVERSEPERMUTATIONS(PermParams, PermShifts, nRounds, ImgDim, P)
  invCMSeqs ← GENERATEINVERSECHAOTICSEQS(CMParams, nRounds, ImgDim)
  return {CImg1D, invPermutations, invCMSeqs, Rows, Cols}
end procedure

To perform tasks 3, 4 and 5, several system parameters are defined in the next section.

2.1.1 Generating system parameters

The quadratic chaotic map

$$x_{k+1}^q = A - \left(x_k^q\right)^2 \ mod\ P_q$$

is used to generate the other encryption-decryption system parameters. To avoid obtaining $x_k^q = 0$ at any iteration, it is appropriate to choose A as a quadratic non-residue modulo P_q. A pseudo code of the quadratic map is given in Algorithm 5.

Algorithm 5 Generating system params using a quadratic chaotic map

```
procedure QUADRATICMAP(A, x_0, P_q, NumParams)
    Declare global param N
    for k ← 1 to N + 1 do
        x_k ← (A − x_{k−1}^2) mod P_q
    end for
    Params[1] = x_{N+1}
    for k = 2 to NumParams do
        Params[k] ← (A − (Params[k − 1])^2) mod P_q
    end for
    return Params
end procedure
```

Each of the two communicating parties has to generate $3 \cdot nRounds + 5$ parameters. These paremeters are the following:

(i) A prime number $P > ImgDim$ such that $Q = \frac{P-1}{2}$ is also prime,

(ii) *nRound* permutation generators and *nRounds* permutation shifts, both are generated by the quadratic chaotic map,

(iii) The inverses of the elements $2, 4, 6$, and 24 modulo P_c named i_2, i_4, i_6, and i_{24}, respectively. These parameters are generated by using the extended Euclidean algorithm,

(iv) *nRound* starting points for chaotic sequences, generated by the quadratic chaotic map.

The function `getSysParams` (Algorithm 6) is used to compute the $3nRounds + 5$ system parameters.

2.1.2 Generating permutations

The generation of permutations required for the confusion steps, depends on the concept of cyclic groups. Therefore, we refer to the following theorems and result in cyclic groups.

Definition 1. Let $\mathbb{G}^*$ be a finite multiplicative group, with $|\mathbb{G}^*| = n$. Then, $\mathbb{G}^*$ is *cyclic*, if found an element $a \in \mathbb{G}^*$ (called group generator) such that $\text{ord}(a) = |\mathbb{G}^*| = n$. The generator a is called a `primitive root` of group $\mathbb{G}$.

An important multiplicative group is $\mathbb{Z}_n^* = \mathbb{Z}^*/n\mathbb{Z}^* = \{a \in \mathbb{Z}^* : gcd(a,n) = 1 \text{ and } 1 \leq a < n\}$. The order of the multiplicative group $\mathbb{Z}_n^*$ is $|\mathbb{Z}_n^*| = \varphi(n)$, where $\varphi(n)$ is the Euler totient function ($\varphi(n)$ is the number of elements in $\mathbb{Z}_n^*$ that are co-prime to n).

The multiplicative group $\mathbb{Z}_n^*$ is cyclic if and only if

$$n = \begin{cases} 2^m, & m = 0, 1, 2 \\ P^k, & k \in \mathbb{Z}^+ \\ 2P^k, & k \in \mathbb{Z}^+ \end{cases}$$

where $P > 2$ is a prime.

Algorithm 6 Obtaining the system parameters needed for the encryption decryption algorithms

procedure GETSYSPARAMS(ImgDim, P_c, A, x_0^q, P_q, m, nRounds)
  Q ← CEIL(ImgDim/2)
  if ISEVEN(Q) **then**
    Q ← Q + 1
  end if
  while (**not** ISPRIME(Q) **or not** ISPRIME(P)) **do**
    Q ← Q + 2
    P ← 2 · Q + 1
  end while
  i_2 ← EXTENDEDEUCLIDEAN(2, P_c)
  i_4 ← EXTENDEDEUCLIDEAN(4, P_c)
  i_6 ← EXTENDEDEUCLIDEAN(6, P_c)
  i_{24} ← EXTENDEDEUCLIDEAN(24, P_c)
  PCParams ← QUADRATICMAP(A, x_0^q, P_q, 3 · nRounds)
  for k ← 1 **to** nRounds **do**
    Generators ← PCParams[k]
    PermShifts ← PCParams[nRounds + k]
    CMParams ← PCParams[2 · nRounds + k]
  end for
  PermParams ← FINDCLOSESTGENERATORS(Generators, P, nRounds)
  return {P, i_2, i_4, i_6, i_{24}, PermParams, PermShifts, CMParams}
end procedure

Theorem 1. Let $P > 2$ be a prime number. Then $\mathbb{Z}_P^*$ is a cyclic group with $\varphi(P-1)$ generators.

Theorem 2 (Fermat Little Theorem)**.** If $P > 2$ be a prime number and $a \in \mathbb{Z}_P^*$, then $a^{P-1} \equiv 1 \ mod \ P$.

The proofs of theorems 1 and 2 can be found in [5].

By Fermat little theorem, if $a \in \mathbb{Z}_P^*$ then $ord(a)/(P-1)$. Hence, the orders of $\mathbb{Z}_P^*$ elements are the prime numbers that divide $P-1$. We have the following theorems.

Theorem 3. If $a \in \mathbb{Z}_P^*$, then a is a generator of $\mathbb{Z}_P^*$ if and only if $a^{\frac{P-1}{q}} \neq 1 \ mod \ P$ for any prime number $q/(P-1)$, $q > 1$.

Theorem 4. *If P is a prime number, then* $\mathbb{Z}_P^*$ *has* $\varphi(P-1)$ *generators.*

The proofs of theorems 3 and 4 are in [13].

Result 1. If P is a prime number such that $Q = \frac{P-1}{2}$ is also a prime number, then the order of each element in $\mathbb{Z}_P^*$ is either $1, Q$ or $P-1$. In this case, the multiplicative group $\mathbb{Z}_p^*$ has $\varphi(P-1) = Q-1$ generators.

Result 2. If P is a prime number such that $Q = \frac{P-1}{2}$ is also a prime number, and $g \in \mathbb{Z}_P^*$ is a generator of $\mathbb{Z}_P^*$, then the generators of $\mathbb{Z}_P^*$ are the elements of the set:

$$\left\{ g^k : 1 \leq k \leq P-1 \ \& \ gcd(k, P-1) = 1 \right\}$$

Based on Result 1, $a \in Z_P^*$ is not a generator of Z_P^* if either $a^2 \equiv 1 \ mod(P)$ or $a^Q \equiv 1 \ mod(P)$. Otherwise, $a^{P-1} \equiv 1 \ mod(P)$ and a is a generator of Z_P^*. A function `IsGenerator` receives an integer x and the prime number P, and tests whether x is a generator of $\mathbb{Z}_P^*$. If $x^2 \equiv 1 \ mod(P)$ or $x^Q \equiv 1 \ mod(P)$, then x is not a generator and the function `IsGenerator` returns `False`. Otherwise x is a generator and

Algorithm 7 Testing whether element x is a generator of the finite field $\mathbb{Z}_P^*$

```
procedure IsGenerator(x, P)
    isGen ← True
    Q ← (P−1)/2
    xPowm ← x^2 mod(P)
    if xPowm = 1 then
        isGen ← False
        return
    end if
    for k ← 3 to Q do
        xPowm ← x * xPowm mod(P)
    end for
    if xPowm = 1 then
        isGen ← False
        return
    end if
    return IsGen
end procedure
```

the function `IsGenerator` returns `True`. A pseudo-code of the function `IsGenerator` is described in Algorithm 7.

The function `getSysParams` finds *nRounds* generators of $\mathbb{Z}_P^*$ by calling a function `FindClosestGenerators`. This function receives a vector `Generators` of length `NumGenerators`, the prime number P and an integer `NumGenerators`. It returns a vector `PermParams` of length *NumGenerators* whose components are Generators of $\mathbb{Z}_P^*$. A pseudo-code of the function `FindClosestGenerators` is given in Algorithm 8.

Algorithm 8 Finding the closest generators of the finite field $\mathbb{Z}_P^*$ to the components of vector `Generators`

```
procedure FindClosestGenerators(Generators, P, NumGenerators)
    PermParams ← Array(1, NumGenerators)
    for k ← 1 to NumGenerators do
        g ← Generators[k]
        while not IsGenerator(g, P) do
            g ← g + 1
        end while
        PermParams[k] ← g
    end for
    return PermParams
end procedure
```

If g is a generator of $\mathbb{Z}_P^*$, then the elements of $\mathbb{Z}_P^* = \{g, g^2, g^3, \ldots, g^{P-2}, g^{P-1}(\equiv 1 \ mod \ P)\}$. A function `getPermutation` receives a generator *Gen* and the prime number P, returns the permutation $\{Gen, Gen^2, \ldots, Gen^{P-1}(\equiv 1 \ mod \ P)\}$. Its pseudo-code is given in Algorithm 9.

After obtaining the vector `PermParams` of *nRounds* generators of Z_P^*, the function `InitializeEnc` calls a function `GeneratePermutations`. It receives the vector `PermParams`, a vector of permutations' shifts `PermShifts`, the prime number P and the required permutation length `ImgDim`. It iterates over the *nRounds* generators of $\mathbb{Z}_P^*$ and in each round k. It calls the function `getPermutation` to receive the corresponding k^{th} permutation P_k. Then, the function `GeneratePermutations` extracts the kth permutation $Perm_k$ from P_k by dropping the components with values bigger than *ImgDim*. The last

Algorithm 9 Finding the permutation that is generated by generator *Gen* of the multiplicative group $\mathbb{Z}_P^*$

```
procedure GETPERMUTATION(Gen, P)
    Perm ← Array(1, P − 1)
    Perm[1] ← Gen
    for k = 2 to P − 1 do
        Perm[k] ← (Gen · Perm[k − 1]) mod P
    end for
    return Perm
end procedure
```

element of the generated permutation $Perm_k$ is 1. To avoid the use of this information to design any kind of attack, a circular shift is carried out on $Perm_k$ by an amount of $PermShift[k]$ *mod ImgDim*. This way, the positions of permutation elements remain unknown. Algorithm 10 illustrates The pseudo-code of `GeneratePermutations`.

Algorithm 10 Finding the *nRounds* permutations each of length *ImgDim* generated by the components of *PermParams*

```
procedure GENERATEPERMUTATIONS(PermParams, PermShifts, nRounds, ImgDim, P)
    Perms ← Array(nRounds, ImgDim)
    for k ← 1 to nRounds do
        Gen ← PermParams[k]
        Perm ← GETPERMUTATION(Gen, P)
        j ← 1
        for m ← 1 to P − 1 do
            if Perm[m] ≤ ImgDim then
                Perms[k, j] ← Perm[m]
                j ← j + 1
            end if
        end for
    end for
    for k ← 1 to nRounds do
        PermShifts[k] ← PermShifts[k] mod ImgDim
        Perms[k, :] ← CIRCULARSHIFTS(Perms[k, :], PermShifts[k])
    end for
    return Perms
end procedure
```

If s is a permutation defined on a set S, then the inverse permutation of s is denoted by s^{-1} satisfies

$$s^{-1}(s(k)) = k, \forall k \in S.$$

To decrypt the cipher image, the receiver has to generate the inverse permutaions of those generated by the sender. A function `GetInversePermutations` receives parameters `PermParams, PermShifts, nRounds, ImgDim` and the prime number `P`. It returns the inverse permutations. The pseudo code of the function `GetInversePermutations` is given in Algorithm 11.

2.1.3 Generating chaotic sequences using integer Chebyshev map

The Chebyshev chaotic map is of the form

$$c_{n+1} = \cos\left(N \cdot \cos^{-1}(c_n)\right), n \geq 1,\ N \geq 2. \tag{1}$$

Algorithm 11 Finding the *nRounds* inverse permutations

```
procedure GENERATEINVERSEPERMUTATIONS(PermParams, PermShifts, nRounds, ImgDim, P)
    Perms ← GENERATEPERMUTATIONS(PermParams, PermShifts, nRounds, ImgDim, P)
    for k ← 1 to nRounds do
        for j ← 1 to ImgDim do
            invPerms[k, Perms[j]] ← j
        end for
    end for
    return invPerms
end procedure
```

If $-1 \leq c_0 \leq 1$, the Chebyshev map generates a peudo-random sequence of real numbers in $[-1,1]$. In this section, we introduce a new integer version of Chebyshev chaotic map over a finite multiplicative group $\mathbb{Z}^*_{P_c}$. This new integer map is based on the Maclaurin series expansions of the functions $\cos(x)$ and $\cos^{-1}(x)$, where $P_c \equiv 1 \ mod\ 4$. The Maclaurin's expansions of $\cos(x)$ and $\cos^{-1}(x)$ are given by:

$$\cos(\mathrm{x}) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \ldots, \text{ and} \tag{2}$$

$$\cos^{-1}(x) = \frac{\pi}{2} - x - \frac{x^3}{6} - \ldots; -1 \leq x \leq 1. \tag{3}$$

Approximating with the first three terms of the expansions give:

$$\cos(x) \approx 1 - \frac{x^2}{2} + \frac{x^4}{4!} \text{ and } \cos^{-1}(x) \approx \frac{\pi}{2} - x - \frac{x^3}{6}$$

Over $\mathbb{Z}^*_{P_c}$, the functions $\cos(x)$ and $\cos^{-1}(x)$ are approximated as follows:

$$\cos(\mathrm{x}) \approx \left(P_c + 1 - (2^{-1} mod\ P_c)x^2 + (24^{-1} mod\ P_c)x^4\right)\ mod\ P_c = S(P_c, x) \tag{4}$$

$$\text{and}\cos^{-1}(\mathrm{y}) \approx \left((4^{-1} mod P_c)(P_c - 1) - y - (6^{-1}\ mod\ P_c)y^3\right)\ mod\ P_c = T(P_c, x), \tag{5}$$

where, the extended Euclidean algorithm is be used for computing the multiplicative inverses of the elements $2, 4, 6$ and 24 in $\mathbb{Z}^*_{P_c}$. Also, since $\cos(x)$ is periodic with a period of 2π and $\pi/2$ is the quarter of the period, it is assumed that $(P_c - 1)/4$ is the corresponding point to $\pi/2$. If $z = N \cdot T(P_c, x)$, then $\cos\left(N \cos^{-1}(x)\right)$ can be approximated as

$$\cos\left(N \cos^{-1}(x)\right) \approx S(P_c, z).$$

Starting from an initial condition x_0^c, x_{n+1}^c can be obtained from x_n^c by using:

$$x_{n+1}^c = S(P_c, T(P_c, x_n^c)).$$

Starting from x_0^c with $N = N_c$, a function `genIntChebSeq` receives four parameters N_c, P_c, x_0^c and *LenS*. It generates a sequence S of length *LenS*. A pseudo code of `genIntChebSeq` is given in Algorithm 12.

The function `InitializeEnc` receives the vector `CMParams` from the function `getSysParams`. The elements of `CMParams` are to be the initial states of the chaotic sequences. A function `GetIntChebSeqs` receives parameters `CMParams`, N_c, P_c, *nRounds* and *ImgDim* from the function `InitializeEnc`. It then calls the function `genIntChebSeq` to generate a set of *nRounds* chaotic sequences. A pseudo-code for the function `GetIntChebSeqs` is described by Algorithm 13.

Algorithm 12 Generating a pseudo-random sequence of integers modulo P_c using the integer version of Chebyshev chaotic map

procedure GENINTCHEBSEQ(N_c, P_c, x_0^c, LenS)
  $S \leftarrow \text{zeros}(1, \text{LenS})$
  $S[1] \leftarrow x_0^c$
  for $k \leftarrow 2$ **to** LenS **do**
    $y_k \leftarrow \left(i4 \cdot (P_c - 1) - S[k-1] - i6 \cdot (S[k-1])^3\right) \bmod P_c$
    $S[k] \leftarrow \left(P_c + 1 - i2 \cdot N_c^2 \cdot y_k^2 + i24 \cdot N_c^4 \cdot (S[k-1])^4\right) \bmod P_c$
  end for
  return S
end procedure

Algorithm 13 Generating *nRounds* chaotic sequences each of length *ImgDim* using the Chebyshev chaotic map.

procedure GETINTCHEBSEQS(CMParams, N_c, P_c, nRounds, ImgDim)
  ChaotSeq $\leftarrow$ Array(nRounds, ImgDim)
  for $k \leftarrow 1$ **to** nRounds **do**
    ChaotSeq[k, :] $\leftarrow$ GENINTCHEBSEQ(N_c, P_c, CMParams[k], ImgDim)
  end for
  for $r \leftarrow 1$ to nRounds **do**
    for $c \leftarrow 1$ to ImgDim **do**
      ChaoticSeq[r, c] $\leftarrow$ ChaoticSeq[r, c] mod 256
    end for
  end for
  return ChaotSeq
end procedure

Also, the function `InitializeDec` receives the vector `CMParams` from the function `getSysParams`. It calls a function `GenerateInverseChaoticSeqs` to generate the additive inverse chaotic sequences modulo 256. The function `GenerateInverseChaoticSeqs` also receives the parameters `CMParams`, N_c, P_c, *nRounds* and *ImgDim*. It then calls the function `GetIntChebSeq` and subtracts the resulting sequences from 256. Algorithm 14 describes the pseudo-code of the function `GenerateInverseChaoticSeqs`.

Algorithm 14 Generating *nRounds* inverse chaotic sequences each of length *ImgDim* using the Chebyshev chaotic map.

procedure GENERATEINVERSECHAOTICSEQS(CMParams, N_c, P_c, nRounds, ImgDim)
  invChaoticSeqs $\leftarrow$ Array(nRounds, ImgDim)
  ChaotSeqs $\leftarrow$ GETINTCHEBSEQS(CMParams, N_c, P_c, nRounds, ImgDim)
  for $r \leftarrow 1$ to nRounds **do**
    for $c \leftarrow 1$ to ImgDim **do**
      invChaoticSeqs[r, c] $\leftarrow$ (256 − ChaoticSeqs[r, c]) mod 256
    end for
  end for
  return invChaoticSeqs
end procedure

The proposed method for image encryption and decryption is implemented via the algorithms that are detailed in this section. Series of simulations that illustrate the performance of the proposed method are given in the next section.

3 Performance of the proposed encryption algorithm

We consider for the simulations five gray-scale standard images, namely, Lena, peppers, baboon, Barbara and the camera man images. Each of the five images has a size of 512×512. However, for purpose of comparisons, these test images of sizes 256×256 are also used.

The plain, encrypted and decipher images are explained in Figure 2. The plain images are shown in the first column of Figure 2, while the encrypted and decipher images are shown in the second and third columns, respectively.

To measure the similarity between the plain and decipher images, the MSE, PSNR, and SSIM between the plain and decipher images are computed using the following formulas:

$$\mathrm{MSE} = \frac{1}{\mathrm{M} \times \mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{M}} (\mathrm{P(i,j)} - \mathrm{D(i,j)})^2 \tag{6}$$

where M and N are the image dimensions.

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\mathrm{I}_{\mathrm{Max}}^2}{\mathrm{MSE}}\right) \tag{7}$$

where I_{max} is the maximum possible value of the image (255).

If $\mu_x, \mu_y, \sigma_x, \sigma_y$, and σ_{xy} denoted the means of x and y, the variances of x and y and the covariance of x and y, respectively, then the SSIM is defined by

$$\mathrm{SSIM} = \left(\frac{2\mu_\mathrm{x}\mu_\mathrm{y} + \mathrm{c}_1}{\mu_\mathrm{x}^2 + \mu_\mathrm{y}^2 + \mathrm{c}_1}\right)\left(\frac{2\sigma_\mathrm{xy} + \mathrm{c}_2}{\sigma_\mathrm{x}^2\sigma_\mathrm{y}^2 + \mathrm{c}_2}\right) \tag{8}$$

where c_1 and c_2 are constants.

The *MSE*, *PSNR*, and *SSIM* values for the test images are illustrated in Table 1. The *MSE* values in Table 1 show that the decipher images are perfectly identical (0 MSE) to the plain images.

Table 1: The similarity between the plain and deciphered images.

Image	MSE	PSNR	SSIM
Lena	0.00	∞	1.00
Peppers	0.00	∞	1.00
Baboon	0.00	∞	1.00
Barbara	0.00	∞	1.00
Cameraman	0.00	∞	1.00

The performance analyses of the proposed method are discussed in the four subsequent sections, including the key space analysis, statistical analysis, sensitivity analysis and the performance of the method under attacks.

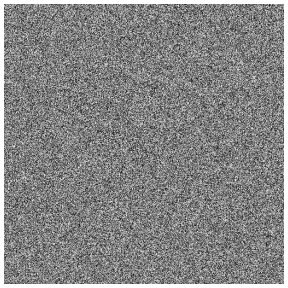
3.1 The key space analysis

The proposed image encryption method uses two chaotic maps with a total of six parameters. The parameters of the quadratic map are x_0^q, A and P_q taken from the multiplicative group $\mathbb{Z}_{P_q}^*$ and the parameters of the integer version of Chebyshev chaotic map are x_0^c, N_c and P_c taken from the multiplicative group $\mathbb{Z}_{P_c}^*$.

If P_q has a length b_q-bits and P_c is of length b_c-bits, then the key space of the algorithm is consisting of $2^{3b_q-1} \times 2^{3b_c} = 2^{3(b_q+b_c)-1}$ binary keys of lengths $3(b_q+b_c)-1$-bits. Given that the lengths of x_0^q and



(a) Lena plain image.



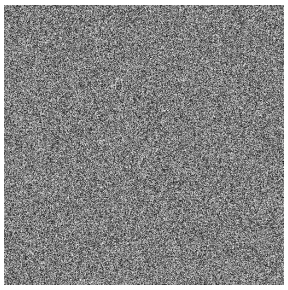
(b) Lena cipher image.



(c) Lena decipher image.



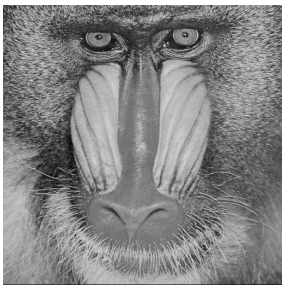
(d) Peppers plain image.



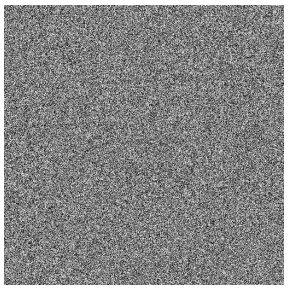
(e) Peppers cipher image.



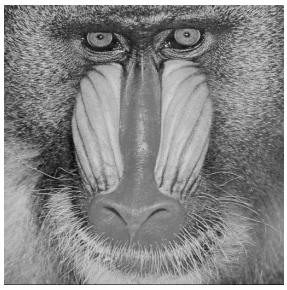
(f) Peppers decipher image.



(g) Baboon plain image.



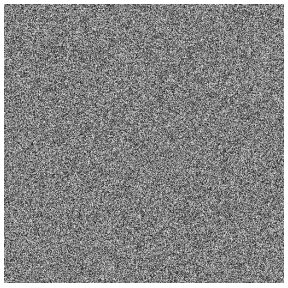
(h) Baboon cipher image.



(i) Baboon decipher image.



(j) Barbara plain image.



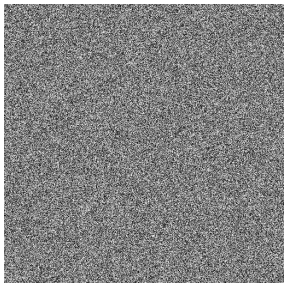
(k) Barbara cipher image.



(l) Barbara decipher image.



(m) Cameraman plain image.



(n) Cameraman cipher image.



(o) Cameraman decipher image.

Figure 2: Encrypted and deciphered 512×512 images of Lena, peppers, baboon, Barbara and Cameraman.

P_q are b_q-bits and the length of A is $b_q - 1$-bits, giving a total length of $3b_q - 1$-bits. Also, the lengths of x_0^c, N_c or P_c is b_c-bits, so the total length is $3b_c$-bits.

The proposed algorithm is safe against the brute force attach if $3(b_q + b_c) - 1 \geq 128$ which requires that at least $b_q = 32$ and $b_c = 16$ or vice versa. In this case, there are at least 2^{143} possible binary keys. This key space is much bigger than the key space for the standard AES-128.

3.2 Statistical analysis

To avoid statistical attacks, the cipher images must have different statistical properties than the plain images. In this section, three statistical analyses types are considered, namely the correlation analysis, information entropy analysis, and the histogram analysis.

3.2.1 Correlation analysis

If x and y are two adjacent pixels in an image, then the expected value $E(x)$, variance $V(x)$ and covariance $cov(x,y)$ are given by the formulas

$$\begin{aligned} E(x) &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ V(x) &= \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))^2 \\ cov(x,y) &= \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))\,(y_i - E(y)) \end{aligned}$$

where N is the number of adjacent pixel pairs from the image.

The the correlation between pixel x and pixel y are given by

$$\gamma_{xy} = \frac{\mathrm{cov(x,y)}}{\sqrt{\mathrm{V(x)}}\sqrt{\mathrm{V(y)}}}$$

Table 2 illustrates the horizontal, diagonal and vertical pixel correlations of the five cipher images. It compares the results obtained by the proposed method to the results in references [16], [30], [23] and [12].

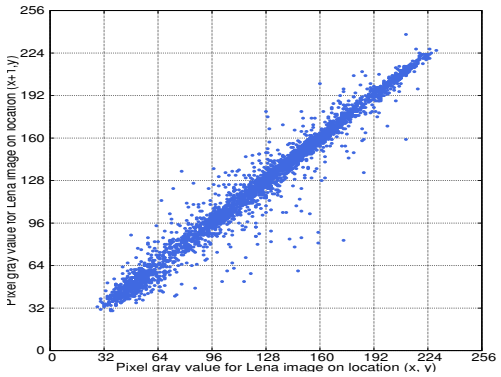
The pixel correlations of the plain and cipher images are illustrated in Figure 3, where the first and second columns show vertical correlations between the gray values at position (x,y) of the plain and cipher images, respectively.

3.2.2 Entropy analysis

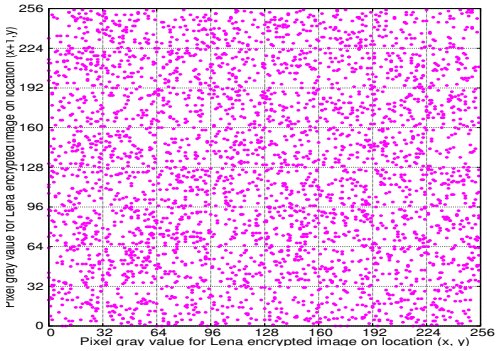
The information entropy is an important randomness measure. It is extensively used to measure the randomness of pixel values of cipher image. The 8-bit pixels of a cipher gray image are uniformly distributed if its entropy value is very close to 8. If s_i is the frequency of the pixel with value i, $i = 0,\ldots,255$ and $P(s_i)$ denotes the probability of s_i, then the entropy is computed as

$$H(s) = \sum_{i=0}^{2^N-1} P(s_i) log_2 \frac{1}{P(s_i)} = -\sum_{i=0}^{2^N-1} P(s_i) log_2(P(s_i))$$

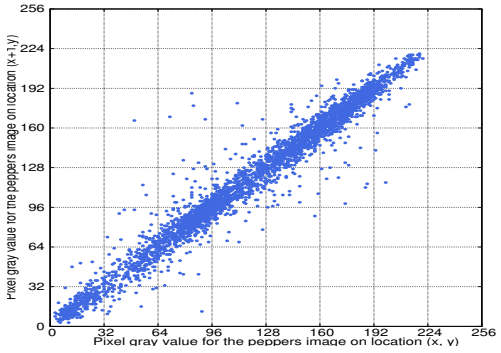
The entropy values of the five plain and cipher images are computed and compared to entropy values found in [16], [30], [23] and [12]. The results are illustrated in Table 3.



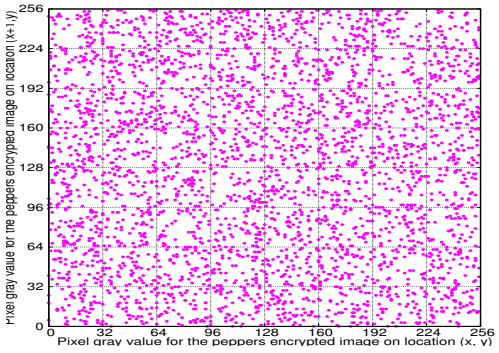
(a) Lena plain image.



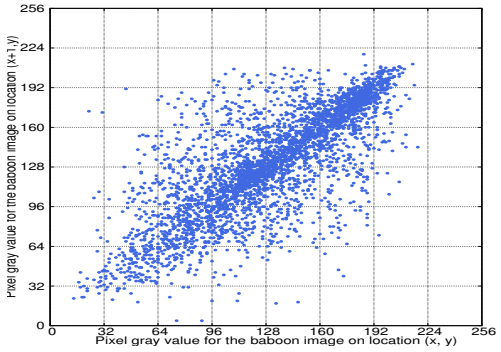
(b) Lena cipher image.



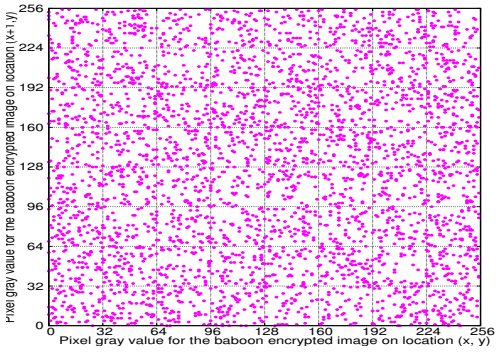
(c) Peppers plain image.



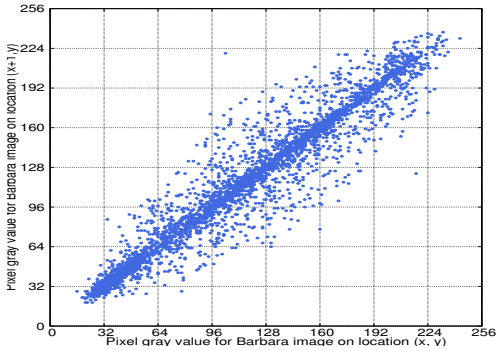
(d) Peppers cipher image.



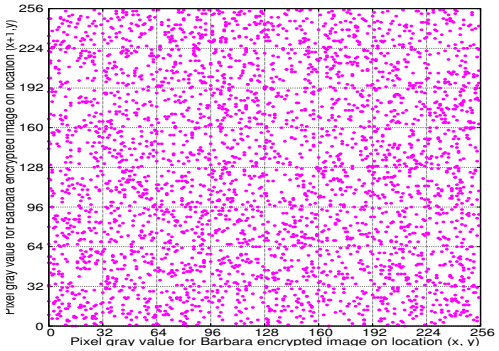
(e) Baboon plain image.



(f) Baboon cipher image.



(g) Barbara plain image.



(h) Barbara cipher image.

Figure 3: Vertical pixel correlations

Table 2: Correlation comparison of different encryption algorithms.

Test Image	Direction	Plain Image	Ciphered Image [16] 256×256	[30] 512×512	[23] 512×512	[12] 512×512	Proposed 256×256	Proposed 512×512
Lena	Horizontal	0.97193	0.0106	0.0045	-0.0685	0.0025	0.00043	0.00079
	Diagonal	0.96853	0.0009	-0.0058	0.0059	-0.0037	0.00183	0.00056
	Vertical	0.98503	-0.0012	0.0018	0.0857	-0.0016	0.00019	-0.00015
Pepper	Horizontal	0.97669	-0.0024	*****	*****	*****	0.00092	-0.00015
	Diagonal	0.96385	-0.005	*****	*****	*****	0.00280	-0.00062
	Vertical	0.97913	0.0142	*****	*****	*****	0.00033	0.00001
Baboon	Horizontal	0.86654	0.023	*****	*****	0.0005	-0.00033	0.00041
	Diagonal	0.72619	-0.0168	*****	*****	0.0013	-0.00069	-0.00083
	Vertical	0.75873	0.0054	*****	*****	-0.00040	-0.00079	0.00092
Barbara	Horizontal	0.89539	0.0040	-0.0018	*****	0.0073	-0.00087	-0.00068
	Diagonal	0.88304	0.0006	-0.0129	*****	-0.0019	-0.00050	0.00093
	Vertical	0.95887	0.00024	0.0091	*****	0.0006	-0.00047	-0.00025
Cameraman	Horizontal	0.9871	0.0113	*****	*****	*****	-0.00069	0.00002
	Diagonal	0.9787	0.0034	*****	*****	*****	-0.00374	0.00159
	Vertical	0.9919	0.0169	*****	*****	****	-0.00024	0.00025

Table 3: Entropy comparisons among different algorithms.

Test Image	Plain Image	Ciphered Image [16] 256×256	[30] 512×512	[23] 512×512	[12] 512×512	Proposed 256×256	Proposed 512×512
Lena	7.4451	7.9972	7.9992	7.9992	7.9992	7.9974	7.9993
Pepper	7.5925	7.9971	*****	*****	*****	7.9973	7.9993
Baboon	7.3583	7.9967	*****	*****	7.9994	7.9975	7.9994
Barbara	7.6321	7.9969	7.9993	*****	7.9994	7.9972	7.9993
Cameraman	7.0057	7.9972	*****	*****	*****	7.9974	7.9993

3.2.3 Histogram analysis

The image histogram shows the distribution of pixels' gray values. The image histograms of the plain and cipher images are illustrated in Figure 4, where the first and second columns shows the distribution of gray values of the plain and cipher images, respectively.

To show that the Gray density frequencies of the cipher images follows the uniform distribution, chi-square test is used. Within this test, the null hypothesis

$$H_0 : \text{the cipher image pixel frequencies are uniformly distributed}$$

is tested against the alternative hypothesis

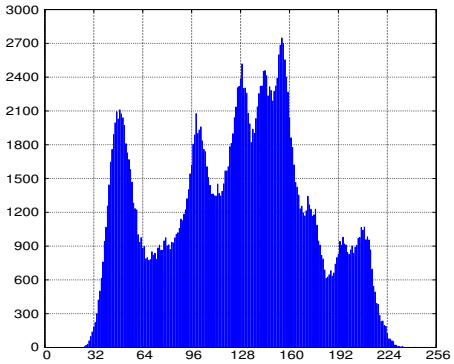
$$H_1 : \text{the cipher image pixel frequencies are not uniformly distributed.}$$

The expected pixel value of a 512×512 image is given by

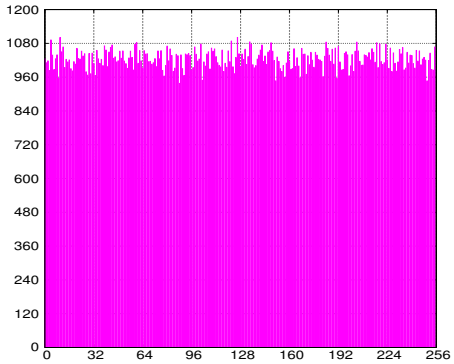
$$E_i = 512^2/256 = 1024, i = 0,\ldots,255.$$

Under 95% confidence interval, the null hypothesis H_0 is accepted if

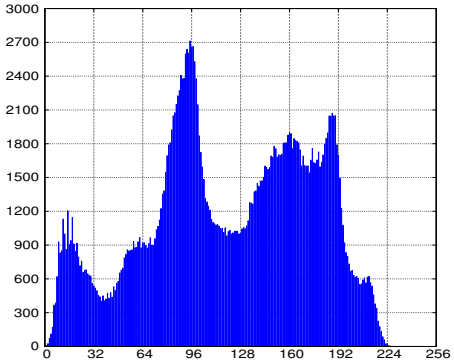
$$\chi^2_{\text{computed}} = \sum_{\text{i}=0}^{255} \frac{(\text{O}_\text{i} - \text{E}_\text{i})^2}{\text{E}_\text{i}} < \chi^2_{\text{inv}}(255, 0.95) \approx 293.24783508,$$



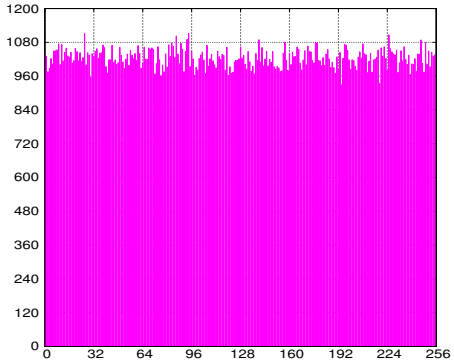
(a) Lena plain image.



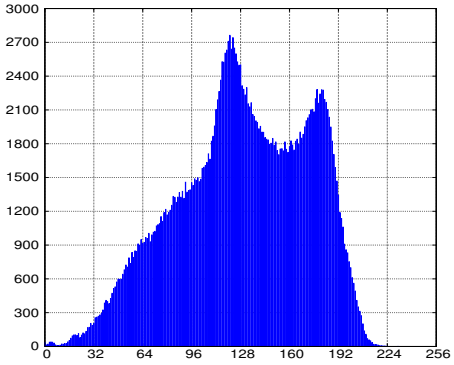
(b) Lena cipher image.



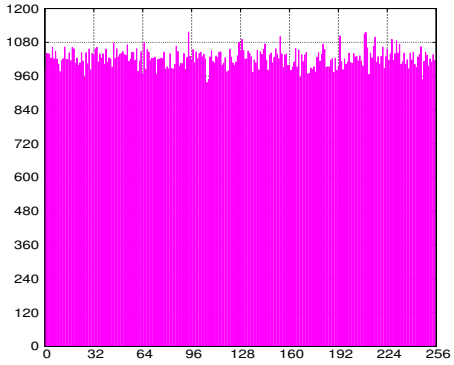
(c) Peppers plain image.



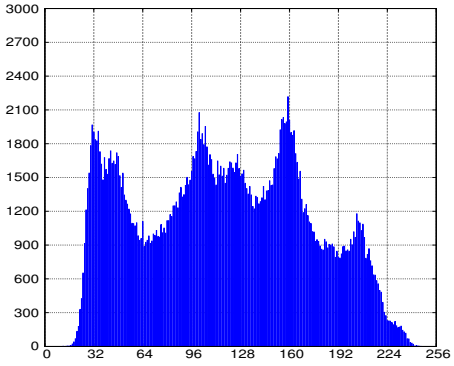
(d) Peppers cipher image.



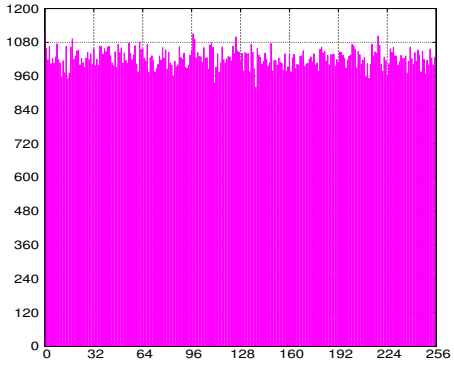
(e) Baboon plain image.



(f) Baboon cipher image.



(g) Barbara plain image.



(h) Barbara cipher image.

Figure 4: Image histograms of the plain and cipher images

where O_i is the frequency of the Gray density i in the cipher image.

The chi-square values for the five cipher images are illustrated in Table 4

Table 4: χ^2 test values for the ciphered images.

Image	$\chi^2_{inv}(255,0.95)$	$\chi^2_{computed}$	Decision
Lena	293.2478351	260.777344	Accept H_0
Peppers	293.2478351	265.285156	Accept H_0
Baboon	293.2478351	233.835938	Accept H_0
Barbara	293.2478351	248.267578	Accept H_0
Cameraman	293.2478351	227.710938	Accept H_0

3.3 Sensitivity analysis

Sensitivity analysis measures the amount of change in the ciphertext when a slight change to the encryption key or plaintext is made. There are two standard measures for the sensitivity of key or plaintext, namely the Number of Pixels Change Rate (NPCR) and the Unified Average Changing Intensity (UACI). The mathematical forms of the NPCR and UACI are

$$\begin{aligned}
\text{NPCR} &= \frac{1}{M\times N}\sum_{i=1}^{N}\sum_{j=1}^{M}|\text{Sign}(C_1(i,j)-C_2(i,j))|\times 100\% \\
\text{UACI} &= \frac{1}{M\times N}\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{|C_1(i,j)-C_2(i,j)|}{255}\times 100\% \\
\text{Sign}(x) &= \begin{cases} 1, x>0; \\ 0, x=0; \\ -1, x<0. \end{cases}
\end{aligned}$$

3.3.1 Key-sensitivity analysis

As introduced earlier, the encryption key consists of 6 parameters P_q, x_0^q, A, P_c, x_0^c, and N_c. To perform the key sensitivity analysis, one of the six parameters is changed while keeping the other parameters fixed. Then, the NPCR and UACI are computed for the two cipher images. Table 5 illustrates the computed values of the NPCR and UACI corresponding to changes in the key parameters. The number between the rounded brackets in the Parameter column points to the difference between the two values of the parameter. For example the first value of the parameter A was 209 and changed to 211. Both 209 and 211 are quadratic non-residues modulo P_q, but not 210.

3.3.2 Differential Attack

To show that the proposed method is resistant to differential attack, one pixel of the plain image P is modified giving an image $\tilde{P}$. Then the two images P and $\tilde{P}$ are encrypted and deciphered with the same key giving two deciphered images D and $\tilde{D}$. The NPCR and the UACI are computed for both cipher images P and $\tilde{P}$ and the two decipher images D and $\tilde{D}$, respectively. This experiment is repeated 100 times while changing one pixel at a random location. Finally, the average values of the NPCR and UACI over the 100 experiments are calculated and illustrated in Table 6.

Table 5: Key sensitivity analysis.

Parameter	Sensitivity measure(%)	Image				
		Lena	Peppers	Baboon	Barbara	Cameraman
P_q (8)	NPCR	99.5846	99.6197	99.5857	99.6033	99.6075
	UACI	33.4880	33.4351	33.3834	33.4031	33.4463
x_0^q (1)	NPCR	99.5941	99.6098	99.5834	99.6483	99.6101
	UACI	33.4253	33.4544	33.5077	33.4719	33.4303
A (2)	NPCR	99.6086	99.5983	99.6017	99.6216	99.5899
	UACI	33.3858	33.4501	33.3964	33.4327	33.4774
x_0^c (1)	NPCR	99.6197	99.6048	99.5926	99.6090	99.6086
	UACI	33.4405	33.4338	33.3965	33.4288	33.4846
P_c (4)	NPCR	99.6078	99.6078	99.6078	99.6078	99.6078
	UACI	33.4629	33.4737	33.5099	33.4448	33.5408
N_c (1)	NPCR	99.5930	99.5930	99.5930	99.5930	99.5930
	UACI	33.4867	33.5250	33.4813	33.4929	33.5241

Table 6: The NPCR and UACI values obtained by one pixel change.

Image	[16]		[30]		Proposed	
	NPCR	UACI	NPCR	UACI	NPCR	UACI
Lena	0.9962	0.3346	0.9962	0.3349	0.9960	0.3338
Peppers	0.9960	0.3348	******	******	0.9961	0.3354
Baboon	0.9961	0.3349	******	******	0.9960	0.3342
Barbara	0.9960	0.3349	0.9949	0.3333	0.9959	0.3347
Cameraman	0.9961	0.3349	******	******	0.9961	0.3345

3.4 Performance of the proposed method under attacks

In this section, we show the resistance of the proposed method to a salt & pepper noise or an occlusion crop attack.

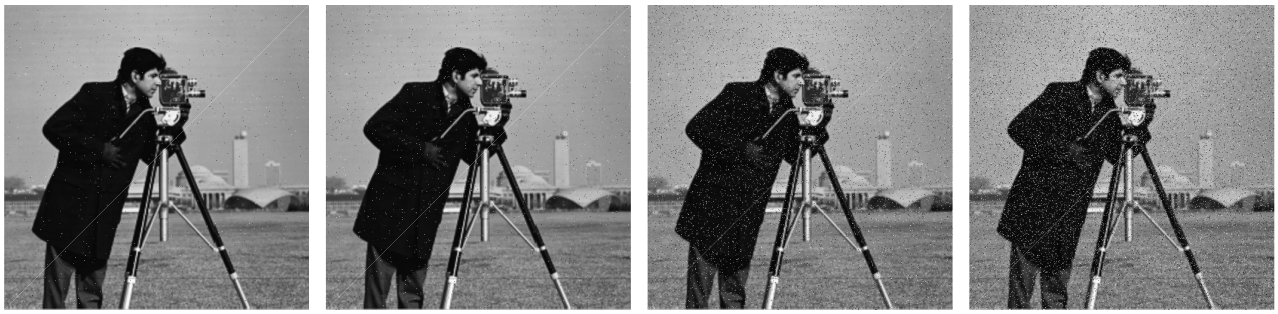
3.4.1 Salt and pepper attack

Salt & pepper noise attack aims to disturb the decryption process by adding a noise into the cipher image in a form of salt & pepper. Different powers of noise are tested to evaluate the resistance of the proposed method, including $0.001, 0.005, 0.01, 0.05$ and 0.1.

Table 7 illustrates the MSE and PSNR measurements for the different noises at powers obtained by the proposed method and existing recent methods from the literature.



(a) Lena: noise density of 0.005. (b) Lena: noise density of 0.01. (c) Lena: noise density of 0.05. (d) Lena: noise density of 0.1.



(e) Cameraman: noise density of 0.005. (f) Cameraman: noise density of 0.01. (g) Cameraman: noise density of 0.05. (h) Cameraman: noise density of 0.1.

Figure 5: decipher images under salt& pepper noise attack fro different power of noise (0.005, 0.01,0.05 0.1)

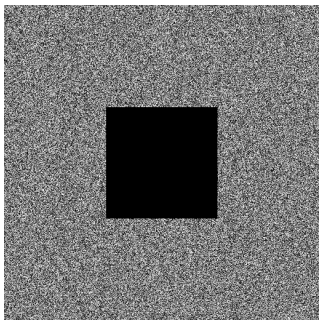
Table 7: Comparison of the proposed Methods with existing ones under Salt & Pepper noise

		Lena					Cameraman				
Methods	Measure	0.001	0.005	0.01	0.05	0.1	0.001	0.005	0.01	0.05	0.1
Ref. [28]	PSNR	38.4	30.5	28.3	21.4	***	38.3	31.5	28.8	21.5	***
Ref. [9]	PSNR	39.1	31.8	28.3	21.3	***	***	***	***	***	***
	MSE	***	***	***	444.1	909.3	***	***	***	***	***
Ref. [12]	PSNR	***	***	***	21.7	18.7	***	***	***	***	***
Proposed	MSE	7.9	38.7	77.5	387.9	775.8	9.4	46.3	93.0	464.9	930.2
Method	PSNR	39.2	32.2	29.2	22.2	19.2	38.4	31.5	28.5	21.5	18.4

3.4.2 Crop attack

The crop attack eliminates a part of the cipher image in order to disturb the process of decryption. This attack can cause the lost of the whole image. Figure 6 illustrates different types of crop attacks for both Lena and Cameraman images. We consider here four types of crop occlusion $1/8, 1/4, 1/2$ and $3/4$. Figure 6 illustrates the deciphered images of Lana and the cameraman under the different levels of crop occlusions.

Table 8 illustrates the MSE and PSNR measurements obtained by the proposed method and the algorithms introduced in [15], [18] and [21] at crop ratios $1/8, 1/4, 1/2$ and $3/4$.



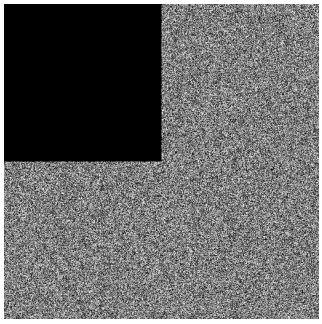
(a) Pixel crop occlusion of $\frac{1}{8}$.



(b) Lena: crop occlusion of $\frac{1}{8}$.



(c) Cameraman: crop occlusion of $\frac{1}{8}$



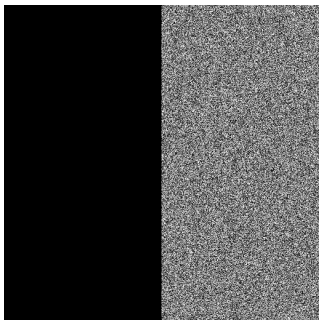
(d) Pixel crop occlusion of $\frac{1}{4}$.



(e) Lena: crop occlusion of $\frac{1}{4}$.



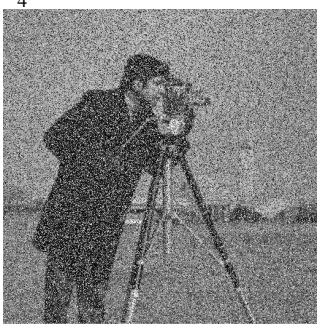
(f) Cameraman: crop occlusion of $\frac{1}{4}$.



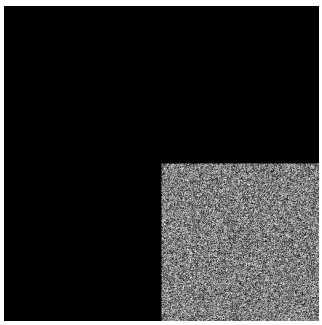
(g) Pixel crop occlusion of $\frac{1}{2}$.



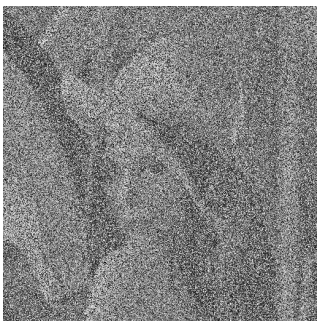
(h) Lena: crop occlusion of $\frac{1}{2}$.



(i) Cameraman: crop occlusion of $\frac{1}{2}$.



(j) Pixel crop occlusion of $\frac{3}{4}$.



(k) Lena: crop occlusion of $\frac{3}{4}$.



(l) Cameraman: crop occlusion of $\frac{3}{4}$.

Figure 6: Resistance to the crop attack for different occlusion

Table 8: Performance of the proposed method under crop attacks and comparisons against methods from literature.

		Lena				Cameraman			
Methods	Measure	1/8	1/4	1/2	3/4	1/8	1/4	1/2	3/4
Ref. [18]	MSE	**2960**	*****	*****	*****	*****	*****	*****	*****
Ref. [21]	PSNR	*****	*****	*****	*****	*****	**15.421**	**11.256**	**7.052**
Ref. [15]	PSNR	*****	**13.88**	**11.27**	**9.6**	*****	*****	*****	*****
Proposed	MSE	**959.77**	1952.90	3889.93	5817.76	1167.14	2341.09	4663.22	6980.44
Method	PSNR	18.30	**15.22**	**12.23**	**10.48**	17.45	**14.44**	**11.44**	**9.69**

4 A proposed enhancement method based on 2D median filter

4.1 Similarity between Crop and salt & pepper noise attacks

By looking at Figure 6, the crop attack looks like a salt & pepper noise attack. To confirm this intuition, we investigate the relationship between the crop and salt & pepper noise attacks at occlusion $1/4$ and $1/2$. We aim to find the powers of salt & pepper noise corresponding these crop occlusion that they have approximately the same PSNR between the deciphered and plain images. For this reason, we conduct a series of simulations to compare the performance of the proposed algorithm under crop attack and its corresponding one under salt & pepper noise attack.

Figure 7 shows the deciphered Lena and cameraman images under crop attacks with occlusions $1/4$ and $1/2$ and their corresponding under salt and pepper attacks with powers 0.252 and 0.499. We observe that we cannot detect visually any differences between the decipher images under crop attack and the one under salt & pepper attack.

Table 9 shows the MSE, PSNR and SSIM measurements for Lena and Cameraman images under salt & pepper and crop attacks at noise powers $1/4$ and $1/2$.

Table 9: Comparison of the proposed Methods with methods from literature under crop attack.

	Lena			Cameraman		
Methods	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Occlusion Crop 1/4	1952.90	15.22	0.1454	2341.09	14.44	0.1263
Salt & Pepper with power 0.252	1957.12	15.21	0.1445	2344.13	14.43	0.1263
Occlusion Crop 1/2	3889.83	12.23	0.07316	4663.22	11.44	0.0640
Salt & Pepper with power 0.499	3869.82	12.25	0.07327	4664.04	11.46	0.0639

Finally, we compare the distribution (Histogram) of decipher images under crop attack and its correspondent under salt & pepper attack for both images Lena and Cameraman. For both decipher images, figures 8 shows the similarity between these distributions, and the differences between them are very small. Given the different results described above, we conclude that within the proposed method the crop attack has the same behaviour as the salt & pepper attack.

4.2 Enhancement of the proposed method

We observe from figures 6 and 7 that the quality of the deciphered image is seriously affected by high levels of attacks. Several image processing techniques are used to enhance the image quality. Since 1979, a 2D median filter has been mostly used to denoise an image corrupted by a salt & pepper noise



(a) Lena decipher image under crop occlusion $1/4$.



(b) Lena decipher image, salt & pepper of power 0.2515



(c) Cameraman cipher image, crop occlusion $1/4$.



(d) Cameraman deciphered Image, salt& pepper of power $1/4$.



(e) Lena decipher image under crop occlusion $1/2$.



(f) Lena decipher image under salt& pepper of power $1/2$

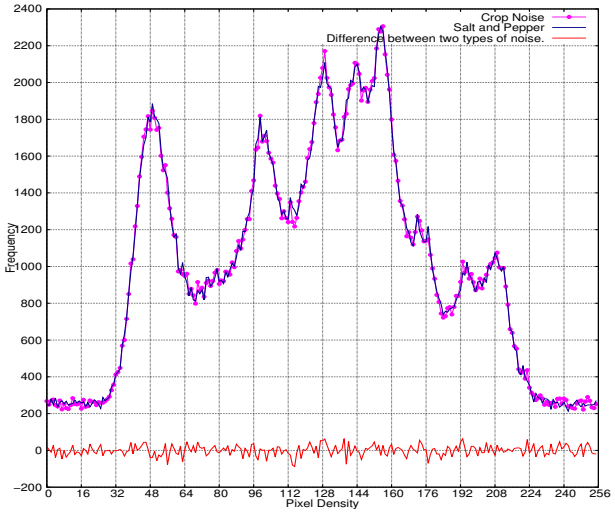


(g) Cameraman decipher image under crop occlusion $1/2$

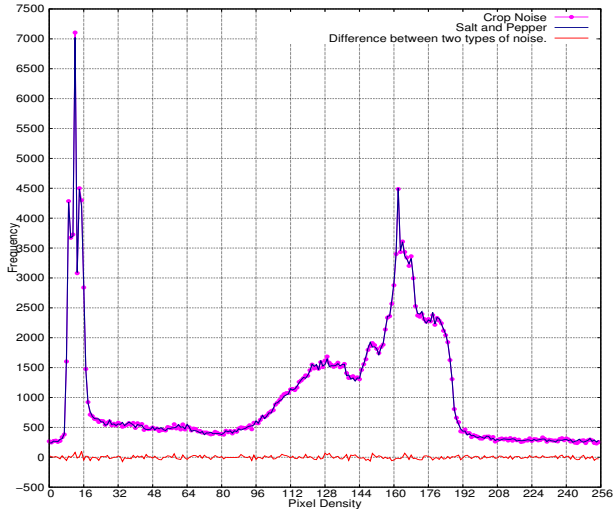


(h) Cameraman decipher image, salt& pepper of power $1/2$

Figure 7: Similarity between crop attack and salt & pepper attack



(a) Comparison of histogram of decipher image (Lena) under Crop and salt & pepper attacks



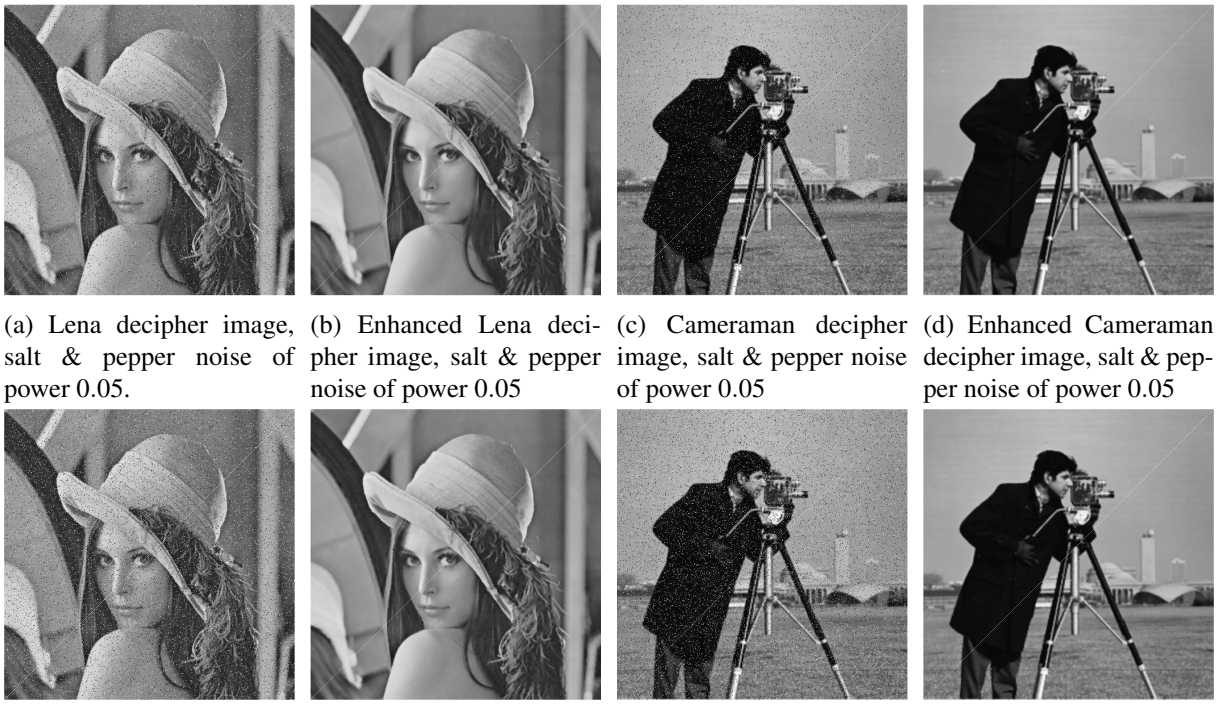
(b) Comparison of histogram of decipher image (Cameraman) under Crop and salt & pepper attacks

Figure 8: Histograms of decipher images under crop and salt & pepper attacks

[10]. It is also shown that the 2-D median filter is efficient to combat the presence of salt and pepper noise.

Given that the crop and salt & peppers noise attacks have the same behaviour, we apply 2D median filters to the deciphered images to enhance their qualities. We associate with each high level of attack a specific size of 2-D median filter.

After applying 2D median filters to the deciphered images, we observe from figures 9 and 10 significant improvements to quality of the deciphered images.



(a) Lena decipher image, salt & pepper noise of power 0.05.

(b) Enhanced Lena decipher image, salt & pepper noise of power 0.05

(c) Cameraman decipher image, salt & pepper noise of power 0.05

(d) Enhanced Cameraman decipher image, salt & pepper noise of power 0.05

(e) Lena decipher image, salt & pepper noise of power 0.1.

(f) Enhanced Lena decipher image, salt & pepper noise of power 0.1

(g) Cameraman decipher image, salt & pepper noise of power 0.1

(h) Enhanced Cameraman decipher image, salt & pepper noise of power 0.1

Figure 9: Enhancement of decipher image under salt & pepper attack by using median filter

Table 10 compares the performance of the proposed method with and without applying median filters.

5 Discussions and comparisons

This section discusses the performance analysis of the proposed method first, in the absence; second in the presence of crop and salt & peppers attacks. Then, it discusses the performance of the median filters when crop and salt & pepper attacks are present.

From Table 2, if we compare the pixel correlations obtained by the proposed methods to those results from the literature, we see that for both cases of 256×256 and 512×512 images, the results obtained by the proposed method are better than the methods found in [16], [30], [23] and [12] in most of the cases.

We see in Table 3 that all the entropy values of cipher images obtained by the proposed method are close to the optimal value 8. Comparing the entropy values obtained by the proposed method to existing ones in the literature ([16], [30], [23] and [12]), we see a comparable outcome.

The chi-square test results illustrated in Table 4, ensure that the pixel density distribution in the cipher images of Lena, peppers, baboon, Barbara, and cameraman are uniformly distributed in the space.



(a) Lena decipher image, crop occlusion attack $\frac{1}{8}$.

(b) Enhanced Lena decipher image, crop occlusion attack $\frac{1}{8}$.

(c) Cameraman decipher image, crop occlusion attack $\frac{1}{8}$.

(d) Enhanced Cameraman decipher image, crop occlusion attack $\frac{1}{8}$.

(e) Lena decipher image, crop occlusion attack $\frac{1}{4}$.

(f) Enhanced Lena decipher image, crop occlusion attack $\frac{1}{4}$.

(g) Cameraman decipher image, crop occlusion attack $\frac{1}{4}$.

(h) Enhanced Cameraman decipher image, crop occlusion attack $\frac{1}{4}$

(i) Lena decipher image, crop occlusion attack $\frac{1}{2}$.

(j) Enhanced Lena decipher image, crop occlusion attack $\frac{1}{2}$.

(k) Cameraman decipher image, crop occlusion attack $\frac{1}{2}$.

(l) Enhanced Cameraman decipher image, crop occlusion attack $\frac{1}{2}$

(m) Lena decipher image, crop occlusion attack $\frac{3}{4}$.

(n) Enhanced Lena decipher image, crop occlusion attack $\frac{3}{4}$.

(o) Cameraman decipher image, crop occlusion attack $\frac{3}{4}$.

(p) Enhanced Cameraman decipher image, crop occlusion attack $\frac{3}{4}$.

Figure 10: Resistance to the crop attack for different occlusion

Table 10: Comparison of the proposed method with existing ones under different levels of crop and salt & pepper noise attacks.

Attack	Noise Power	Proposed Method	Lena			Cameraman		
			MSE	PSNR	SSIM	MSE	PSNR	SSIM
Crop occlusion attack	1/8	Without enhancement	959.77	18.30	0.2689	1167.14	17.45	0.2307
		With enhancement	28.85	33.52	0.9087	20.08	35.10	0.9693
	1/4	Without enhancement	1952.90	15.22	0.1454	2341.09	14.44	0.1263
		With enhancement	68.23	29.79	0.8437	76.07	29.31	0.8764
	1/2	Without enhancement	3889.83	12.23	0.0731	4663.22	11.44	0.0640
		With enhancement	175.21	25.69	0.7560	266.86	23.86	0.7464
	3/4	Without enhancement	5817.76	10.48	0.0470	6980.44	9.69	0.0393
		With enhancement	744.44	19.41	0.5899	1510.59	16.33	0.5170
Salt & pepper noise attack	0.01	Prop. Method	77.45	29.24	0.8539	93.00	28.44	0.8278
		With enhancement	8.65	38.76	0.9570	5.73	40.55	0.9920
	0.05	Without enhancement	387.86	22.24	0.5061	464.90	21.45	0.4540
		With enhancement	21.66	34.77	0.9194	10.22	38.03	0.9819
	0.10	Proposed	775.84	19.23	0.3180	930.17	18.44	0.2764
		With enhancement	26.651	33.87	0.9122	15.756	36.15	0.9754
	0.25	Without enhancement	1938.02	15.25	0.1458	2326.08	14.46	0.1273
		With enhancement	65.99	29.93	0.8463	74.51	29.41	0.8767
	0.5	Without enhancement	3877.21	12.24	0.0731	4653.00	11.45	0.0638
		With enhancement	176.01	25.67	0.7559	269.43	23.83	0.7405

The NPCR and UACI measures in tables 5 and 6 are close to 99.6% and 33.4%, respectively. They show that the proposed method is sensitive to slight changes in either the encryption key or plaintext. Therefore, it is resistant against the differential attacks.

Now, we focus on the performance of the proposed method in the presence of crop or salt & pepper noise attacks.

We observe in Figure 5 that for a power of salt & pepper noise attacks below 0.01, the effect of attack is a negligible. In fact the quality of the deciphered images are very close to the original plain images. On the other hand, the effect of noise becomes clearly observable from the power of 0.05. In spite of this high level of attack, the proposed method ensures a good resistance to this attack, where both images Lena and cameraman are still readable. Table 7 shows that the proposed method has better performance in term of MSE compared to [12]. In term of PSNR, the proposed method has a slightly better performance compared to [28] and has the best performance compared to others techniques from noise power 0.05 for Lena image.

From Figure 6, we observe that the proposed method is strongly resilient to the crop attack when the occlusion is less than $1/4$. Starting from crop occlusion $1/2$, the resistance to crop attacks becomes difficult. Hence, the degradation in the quality of the deciphered image is observable. Despite the performance limitations of the proposed method for high crop occlusion values, the proposed method shows a better performance in terms of MSE, PSNR and SSIM compared to the proposed methods in the literature. This can be seen from Table 8 which shows that the proposed method has the lower MSE. As example for occlusion crop $1/8$, the MSE of the proposed method is 959.77 and in [18] is 2.96×10^3. In terms of the PSNR, the proposed method has better performance compared [15] in all different crop occlusion values and also better resistance for high crop occlusion values compared to [21].

Regarding the decryption algorithm against the crop attack, Table 9 shows the performance of the

proposed method under crop and its correspondent of salt & pepper attacks. We observe that for all used measures including MSE, PSNR and SSIM, the relative error between the performance of the proposed method under crop and under salt & pepper attacks doesn't exceed 7‰. As example the relative error for MSE under crop occlusion $1/4$ and for salt & pepper with power 0.252 is $Re = \frac{1952.90-1957.12}{1952.9} = -0.0022$, for PSNR $Re = \frac{15.22-15.21}{15.223} = 0.000657$ and for SSIM $Re = \frac{0.1454-0.1445}{0.1454} = 0.0062$. Therefore, these results confirm that the crop and salt & pepper attacks have similar behaviour. From Figure 10 and Table 10 we see that for large crop or salt & pepper noise powers, the noise affects the clarity of the deciphered image.

It was proven that the median filter is appropriate to improve the quality of image corrupted by salt and pepper noise. Since we showed that the crop and salt & pepper noise attacks have the same behaviour, the 2D median filter is applied to the deciphered images corrupted by either the crop or salt & pepper noise attack, as appears in Figure 10.

For low noise attack with power less than 0.01, tables 7 and 10, show that the performance of the proposed method without filter is still good as $MSE = 77.45$, $PSNR = 29.24$ and $SSIM = 0.8539$ if the power of noise attack is 0.01.

The necessity of using median filter to enhance the decipher image is well observed from the noise power 0.1 or above, and also for all cases of crop attacks starting from $1/8$ crop occlusion.

As an example, MSE yielded by the proposed method under crop attack with occlusion $1/2$ is 3877.21. After applying median filter to the deciphered image, the MSE is decreased to 176.01 and the PSNR increased from 12.24 (without enhancement) to 25.67 (with enhancement). For high crop attack $(3/4)$, Table 10 shows clearly the limitations of the proposed method in terms of the MSE, PSNR and mostly in term of SSIM, which are $5817.76, 10.48$ and 0.0470, respectively. These results are significantly improved when jointly the proposed method with median filter. In this case the MSE of the enhanced deciphered image is decreased to 744.44 and the PSNR and SSIM are increased to 19.41 and 0.5899, respectively.

6 Conclusions

This paper presented a new method for image encryption and decryption. This method involves new techniques for generating two integer chaotic maps and permutations over multiplicative finite groups. The first chaotic map is an integer version of the quadratic map, which is used to generate some system parameters. The second one is an integer version of Chebyshev chaotic map that is used to generate chaotic sequences of integers. The proposed image encryption algorithm consists of multiple rounds, where in each round a permutation is used to shuffle the image pixels and a chaotic sequence is added to change the pixels densities to the transformed image.

The proposed method is tested against many security measures including key space and key sensitivity for robustness of the encryption key, pixel correlations, image histogram, entropy and chi-square analyses methods for testing the robustness of the encryption algorithm based on the cipher image, and the NIST measures including the MSE, PSNR and SSIM for testing the quality of the deciphered image. Based on these measures, we found that the proposed method is slightly better than some recent methods from the literature. As example, the entropy for any ciphered image obtained by the proposed method is much closer to 8 than the other methods from the literature.

Also, the proposed method is tested against the crop and salt & pepper attacks, with crop ratios $1/8, 1/4, 1/2$ and $3/4$, and salt & pepper noises of powers $0.001, 0.005, 0.01, 0.05$ and 0.1. The MSE, PSNR and SSIM are used to measure the quality of the deciphered image compared to the plain image. Under these measures, the results obtained by the proposed method are compared to methods from the literature. The proposed method showed good resistance against crop and salt & pepper attacks and

better performance than methods from the literature. But, the quality of the deciphered image is affected by high levels of crop or salt & pepper noise attacks.

The paper also showed that crop and salt & pepper noise attacks have the same behaviours under the proposed decryption algorithm. Therefore, to enhance the quality of the deciphered image we applied a 2D median filter for both cases of attacks. The quality of such enhanced deciphered image is compared to the quality of the deciphered image (without enhancement) using the MSE, PSNR and SSIM measures. High levels of improvements were observed. As example, at crop occlusion of $1/2$ the PSNR was increased with a rate around 100% (12.23 to 25.69) and the SSIM was increased from 0.06 to 0.74 (1200%)

References

[1] M. Alawida, A. Samsudin, J. S. Teh, and R. S. Alkhawaldeh. A new hybrid digital chaotic system with applications in image encryption. *Signal Processing*, 160:45–58, July 2019.

[2] S. M. H. Alwahbani and E. B. M. Bashier. Speech scrambling based on chaotic maps and one time pad. In *Proc. of the 2013 International Conference on Computing, Electrical and Electronic Engineering (ICCEEE'13), Khartoum, Sudan*, pages 128–133. IEEE, August 2013.

[3] E. Bashier, G. Ahmed, H.-A. Othman, and R. Shappo. Hiding secret messages using artificial DNA sequences generated by integer chaotic maps. *International Journal of Computer Applications*, 70(15):1–5, May 2013.

[4] B. Furht, E. Muharemagic, and D. Socek. *Multimedia Encryption and Watermarking*. Springer US, 2005.

[5] S. Geoff. *Topics in Group Theory*. Springer London Imprint Springer, London, 2000.

[6] M. Ghebleh, A. Kanso, and D. Stevanović. A novel image encryption algorithm based on piecewise linear chaotic maps and least squares approximation. *Multimedia Tools and Applications*, 77(6):7305–7326, April 2017.

[7] L. Hartigan, L. Cussen, S. Meaney, and K. O'Donoghue. Patients' perception of privacy and confidentiality in the emergency department of a busy obstetric unit. *BMC Health Services Research*, 18(1):1–6, December 2018.

[8] K. Hill. Consent, confidentiality and record keeping for the recording and usage of medical images. *Journal of Visual Communication in Medicine*, 29(2):76–79, January 2006.

[9] T. Hu, Y. Liu, L.-H. Gong, and C.-J. Ouyang. An image encryption scheme combining chaos with cycle operation for DNA sequences. *Nonlinear Dynamics*, 87(1):51–66, August 2016.

[10] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, February 1979.

[11] L. M. Jawad and G. Sulong. A survey on emerging challenges in selective color image encryption techniques. *Indian Journal of Science and Technology*, 8(27):1–12, October 2015.

[12] A. Karawia. Image encryption based on fisher-yates shuffling and three dimensional chaotic economic map. *IET Image Processing*, 13(12):2086–2097, October 2019.

[13] J. Kraft and L. Washington. *An Introduction to Number Theory with Cryptography*. Taylor and Francis Ltd, January 2018.

[14] D. Lambić. A new discrete-space chaotic map based on the multiplication of integer numbers and its application in s-box design. *Nonlinear Dynamics*, 100(1):699–711, January 2020.

[15] J. Lang. Image encryption based on the reality-preserving multiple-parameter fractional fourier transform and chaos permutation. *Optics and Lasers in Engineering*, 50(7):929–937, July 2012.

[16] L. Liu, Y. Lei, and D. Wang. A fast chaotic image encryption scheme with simultaneous permutation-diffusion operation. *IEEE Access*, 8:27361–27374, February 2020.

[17] Y. Liu, J. Zhang, D. Han, P. Wu, Y. Sun, and Y. S. Moon. A multidimensional chaotic image encryption algorithm based on the region of interest. *Multimedia Tools and Applications*, 79(25-26):17669–17705, February 2020.

[18] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai. A secure image encryption algorithm based on rubiks cube principle. *Journal of Electrical and Computer Engineering*, 2012:1–13, March 2012.

[19] W. Lu, A. L. Varna, and M. Wu. Confidentiality-preserving image search: A comparative study between homomorphic encryption and distance-preserving randomization. *IEEE Access*, 2:125–141, February 2014.

[20] W. Puech, Z. Erkin, M. Barni, S. Rane, and R. L. Lagendijk. Emerging cryptographic challenges in image and video processing. In *Proc. of the 2012 19th IEEE International Conference on Image Processing, Orlando, FL, USA*, pages 2629–2632. IEEE, September 2012.

[21] N. Rawat, B. Kim, and R. Kumar. Fast digital image encryption based on compressive sensing using structurally random matrices and arnold transform technique. *Optik*, 127(4):2282–2286, February 2016.

[22] L. Shiguo. *Multimedia content encryption: techniques and applications*. CRC Press, Boca Raton, 2009.

[23] Z. Tang, Y. Yang, S. Xu, C. Yu, and X. Zhang. Image encryption with double spiral scans and chaotic maps. *Security and Communication Networks*, 2019:1–15, January 2019.

[24] C. Volos, I. Kyprianidis, and I. Stouboulos. Image encryption process based on chaotic synchronization phenomena. *Signal Processing*, 93(5):1328–1340, May 2013.

[25] X. Wu, H. Kan, and J. Kurths. A new color image encryption scheme based on DNA sequences and multiple improved 1d chaotic maps. *Applied Soft Computing*, 37:24–39, December 2015.

[26] L. Xu, Z. Li, J. Li, and W. Hua. A novel bit-level image encryption algorithm based on chaotic maps. *Optics and Lasers in Engineering*, 78:17–25, March 2016.

[27] R. Ye and H. Zhao. An efficient chaos-based image encryption scheme using affine modular maps. *International Journal of Computer Network and Information Security*, 4(7):41–50, July 2012.

[28] W. Yu, Y. Liu, L. Gong, M. Tian, and L. Tu. Double-image encryption based on spatiotemporal chaos and DNA operations. *Multimedia Tools and Applications*, 78(14):20037–20064, February 2019.

[29] R. Zahmoul, R. Ejbali, and M. Zaied. Image encryption based on new beta chaotic maps. *Optics and Lasers in Engineering*, 96:39–49, September 2017.

[30] Y.-Q. Zhang, J.-L. Hao, and X.-Y. Wang. An efficient image encryption scheme based on s-boxes and fractional-order differential logistic map. *IEEE Access*, 8:54175–54188, March 2020.

Author Biography



Eihab Bashier obtained his PhD in 2009 from the University of the Western Cape in South Africa. He is an associate professor of applied mathematics at the faculty of mathematical sciences and informatics, University of Khartoum, since 2013. Currently, he is at the Faculty of Education and Arts, Sohar University University, Oman. The research interests of Dr. Bashier are mainly in numerical methods for differential equations with applications to biology and in information and computer security with focus in cryptography. In 2011, Dr. Bashier won the African Union and the Third World Academy of Science (AU-TWAS) young scientists national award in basic sciences, technology and Innovation. Dr. Bashier is a reviewer for some international journals and an IEEE member.



Taoufik Ben Jabeur received the M.S. in Applied Mathematics and Ph.D. degrees in Computer Science (Applied Mathematics: Signal Processing) from University of Paris-Descartes, France, in 2005 and 2009, respectively. He was temporary lecturer (2008-2010) at the same institute. He joined the University of Reims, France, as a Postdoc (2010-2011). From May 2011 to May 2014, he has been with Qatar University as a Postdoctoral student. He was also a Research Scientist at Qatar Mobility Innovations Center (QMIC). Since 2019, He has been assistant professor at Dhofar University of applied mathematics. His research interests are: Wireless sensor networks system, multicarrier communications (OFDM) systems, Signal and Image processing, Passive RFID system, time–frequency analysis and Cryptography.