Chapter

Fast Chaotic Encryption for Hyperspectral Images

Carlos Villaseñor, Javier Gomez-Avila, Nancy Arana-Daniel, Alma Y. Alanis and Carlos Lopez-Franco

Abstract

The information collected by hyperspectral images (HI) is essential in applications of remote sensing like object detection, geological process recognition, and identifying materials. However, HI information could be sensitive, and therefore, it should be protected. In this chapter, we show a parallel encryption algorithm specifically designed for HI. The algorithm uses multiple chaotic systems to produce a crossed multidimensional chaotic map for encrypting the image; the scheme takes advantage of the multidimensional nature of HI and is highly parallelizable, which leads to a time-efficient algorithm. We also show that the algorithm gets high-entropy ciphertext and is robust to ciphertext-only attacks.

Keywords: chaotic encryption, hyperspectral images, parallel computing

1. Introduction

Hyperspectral images (HI) or image spectrometry is a spectral sensing technique in which an object is photographed using several well-defined optical bands in the broad spectral range. This technique integrates imaging and spectroscopy to attain both spatial and spectral information from an object. It was originally developed on satellite and airborne platforms for remote sensing applications utilizing satellite imaging data of the earth and planets mostly; however, it has found application in diverse fields such as military defense, medical diagnosis, and agriculture [1].

HI are characterized by their spatial and spectral resolution. The spatial resolution measures the geometric relationship of the image pixels, and the spectral resolution determines the variations within image pixels as a function of wavelength. The HI has two spatial dimensions (m and n) and one spectral dimension (l). The hyperspectral data are represented in the form of a 3D hyperspectral data cube as it is shown in **Figure 1**. Each slice of the cube along the spectral dimension is called band or channel.

HI could be made up of hundreds of contiguous bands for each spatial position. Consequently, each pixel in a hyperspectral image contains a spectrum representing the light-absorbing and/or scattering properties of the spatial region represented by that pixel. The resulting spectrum acts like a signature, which can be used to classify or estimate composition of the material it represents.

However, the information in the HI could be sensitive, and thus, it should be protected. There already exist many algorithms for secure encryption [2], like advanced encryption standard (AES) [3–5] based on irreducible polynomials in

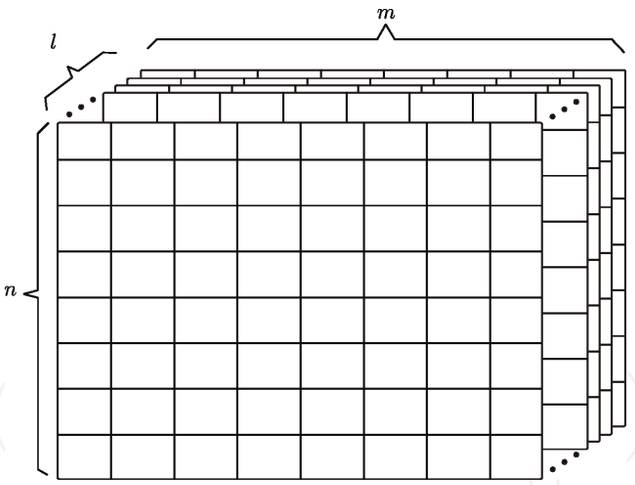

Figure 1.
Hyperspectral image dimensions.

Galois fields or the Rivest-Shamir-Adleman (RSA) algorithm [6] based on large prime number factorization. These algorithms work with raw data, and they offer a secure way to protect it. They work at the binary level of the data, and then they can deal with all kinds of data structures like images, videos, and documents.

The computational run-time of AES and RSA could be very high for a high volume of data. For this reason, in the last decades, the uses of chaos-based cryptography have become very popular [7]. The chaotic systems are deterministic dynamical systems which are sensitive to their initial conditions and parameters.

In this chapter, we present an extended revision of the parallel encryption algorithm based on chaotic systems for HI previously presented in [8]. This algorithm was specifically designed for HI, and it takes advantage of the spatial and spectral distribution to get fast encryption. The algorithm gets high-entropy ciphertext, and it is robust to ciphertext-only attacks.

The chapter is organized as follow: in Section 2, we briefly review the basics of HI and their applications. In Section 3, we explain the characteristics of the chaotic systems and why they are a suitable mathematical tool for encryption. In Section 4, we present our encryption scheme with full details of implementation (we include source code in Matlab and CUDA languages). In Section 5, we present new experimentation and results, and finally, in Section 6, we present our conclusions and future work.

2. Hyperspectral images

A digital image is a rectangular array of n rows and m columns where every element of the array, also called pixel, has an intensity value. New cameras allow us to have multiple measures in the same pixel of different wavelengths, where we represent this information by adding a new dimension l called channels.

The pixels are a discrete quantified measure of light; the intensity represented is an integer proportional to the number of photons detected. In a simple image, the intensity is represented with 2^8 precision, where 0 is no photons detected and 255 is the pixel saturated. Other systems use 2^{12}, 2^{14}, or 2^{16} levels of intensity [9].

Color images have three channels to represent visible light (400–800 nm of wavelength). This kind of camera uses filter for red, green, and blue light. When we have more of three channels, we called it multivariate images or multispectral images, for example, an image of 21 channels.

A common convention is to call hyperspectral images to images with more than 100 channels [9]. Every pixel of a HI has its spectral representation of that position;

classical spectrophotometers use a light source, a filter system to disperse the light into their respective wavelengths, and a detection system. The HI has commonly a spectral resolution of 10–20 nm.

HI have many applications such as oil detection [10], earth remote sensing [11], vegetation and water studies [12, 13], archaeology and art conservation [14–16], finding minerals and other materials [17], medical diagnosis [18–20], food quality and safety control [18, 21, 22], and crime scene analysis and forensic traces [23, 24].

3. Chaotic systems and chaotic encryption

Chaotic systems are dynamical systems that for a certain range of initial conditions show chaotic behavior. This chaotic behavior has the following features:

- Sensitive to initial conditions: the chaotic system is exponentially sensitive to small changes in the initial conditions. These small changes could produce a big difference in the system output.

- Bounded: the states with chaotic behavior have bounded limits.

- Deterministic: a chaotic system does not have randomness, so if we have the initial conditions and parameters, we can simulate the system.

- Aperiodic: there is no periodicity in the chaotic behavior.

The chaotic behavior is a deterministic phenomenon that seems random for a viewer that does not know the initial conditions and parameters of the system; but like any deterministic system, we can obtain the same time-series output by simulating the system with the same initial condition and parameters.

These features make the chaotic system a suitable platform for encryption. We have two principal ways of implementing chaotic encryption:

- Chaotic encryption by synchronization phenomenon: in the case of continuous signals, they could be encrypted by adding a chaotic continuous signal. For the decrypt process, we synchronize another chaotic system throughout the synchronization phenomenon and control theory to subtract the chaotic signal [25, 26].

- Simulating discrete chaotic system: it consists in generating a chaotic mask and mixing with a discrete plaintext with an involution operation. For decrypt, we generate the same mask, and by properties of the involutions, we recover the original signal. In this chapter, we concentrate in the second alternative.

Many chaotic systems have been used for different encryption problems like:

- Stream ciphers: Lorenz system has been used as a stream cipher in [27].

- Image encryption: for image encryption is common in using chaotic maps like Ikeda chaotic map [28], logistic map, tent map, quadratic map, and Bernoulli map [29], a survey is presented in [30].

- Video encryption: the chaotic maps also are used to encrypt video in [31–33].

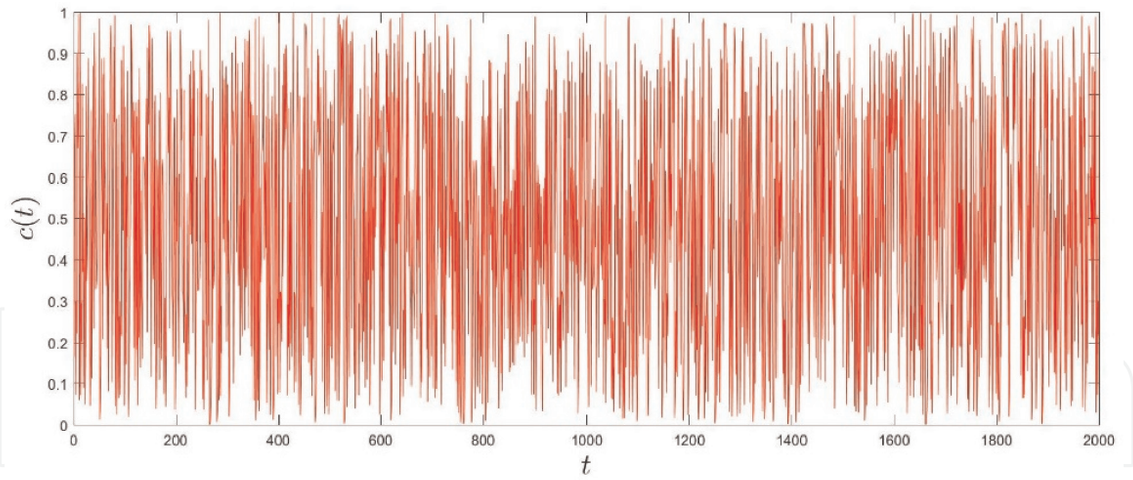

Figure 2.
Simulation of the PLM chaotic system.

For the proposed encryption scheme, we use the piecewise linear chaotic maps (PLM) described by Eq. (1), where $x_0 \in (0, 1)$ is the initial condition and $\mu \in (0, 0.5)$ is the parameter of the system. In **Figure 2**, we present a simulation of the PLM system.

$$x_{n+1} = \frac{x_n - \lfloor \frac{x_n}{\mu} \rfloor \mu}{\mu} \tag{1}$$

4. Chaotic systems for parallel encryption of hyperspectral images

In this section we will review the proposed chaotic encryption scheme for HI presented in [8]. The proposed scheme is shown in **Figure 3**. We propose to use four chaotic systems for encrypting HI. The first three chaotic systems are used to generate a chaotic mask of the same size of the HI, and the last one is used to generate a chaotic substitution box (S-box) according to the algorithm in [34].

The key is formed with initial conditions $\boldsymbol{x_i}$ and parameters $\boldsymbol{\mu_i}$ of the four PLM chaotic systems. It is important to notice that this scheme can be implemented with other discrete chaotic systems. These systems are simulated in parallel to create four different chaotic signals denoted by $\boldsymbol{c_1(t)}$, $\boldsymbol{c_2(t)}$, $\boldsymbol{c_3(t)}$, and $\boldsymbol{c_s(t)}$ that are saved as the key expansion.

In **Figure 4**, we show the source code of the parallel implementation of the chaotic system in the CUDA programming language, where the variable ite represents the maximum number of iterations of the chaotic system, x_vec is a vector with the initial condition parameters, mu_vec is a vector with the parameters of the PLM systems, and C is an array where the chaotic signals are stored.

The chaotic signal $c_s(t)$ is used to generate a chaotic S-box [34]. An S-box is a basic component for symmetric key ciphers because it is a reversible operation for confusion introduction. Confusion is a property of a secure cipher denied by Claude Shannon, and it means that each basic unit of the ciphertext should depend on several parts of the key hiding the relationship between them. The S-box is a permutation which is commonly implemented with a look-up table that maps from a byte to another byte.

In **Figure 5**, we present the Matlab code for the generation of the chaotic S-box and its inverse.

Using the previous codes, we can encrypt the HI by using Eq. (2), where every pixel in the direction (i, j, k) is encrypted in parallel by composing a chaotic signal with the $c_1(t), c_2(t), c_3(t)$ chaotic signals. We combine them by using the XOR

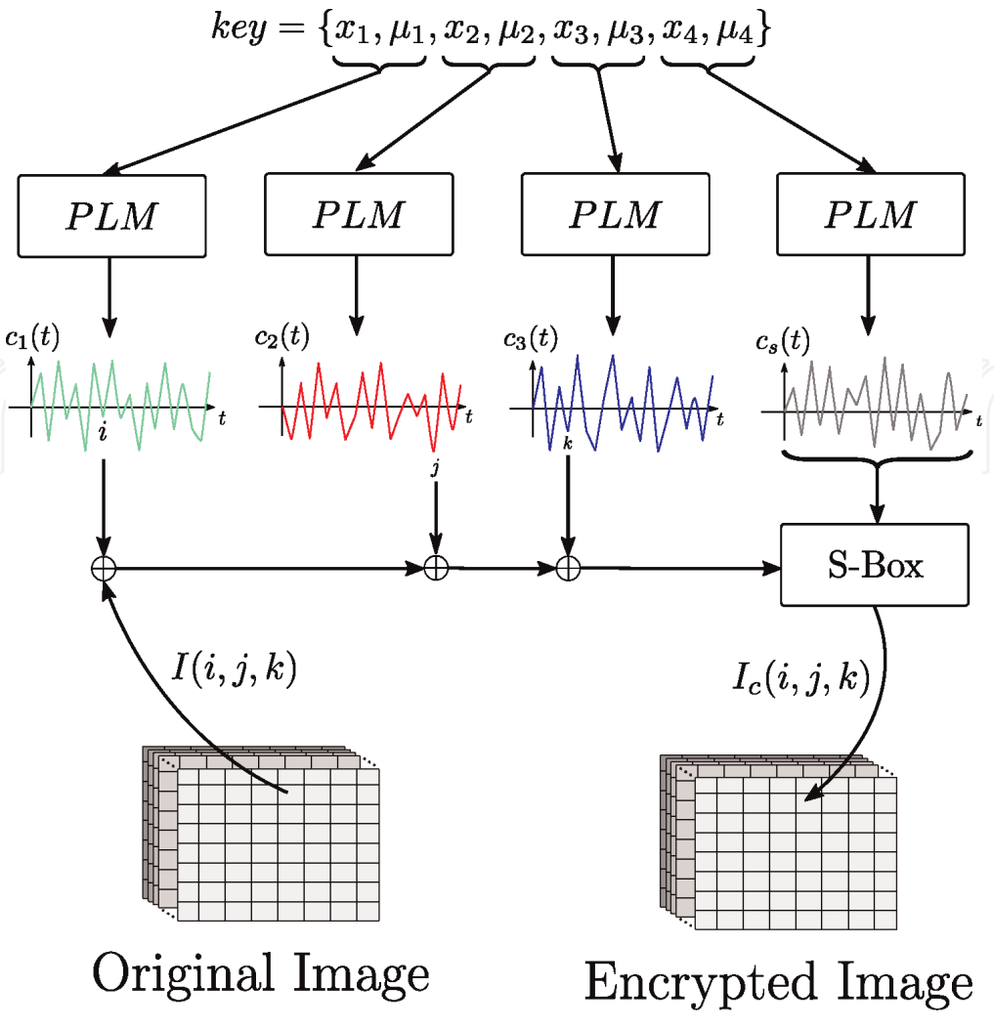

Figure 3.
Encryption scheme for HI.

```
#include "tmwtypes.h"

__device__ double PLM(double x, double mu)
{
        x = (x -floor)x / mu) * mu) / mu;
        return x;
}

__global__ void Chaos(uint8_T * C, const double* x_vec, const double* mu_vec, const
int ite)
{
        double temp_x = x_vec[threadIdx.x];
        for (int i = 0; i < ite; i++)
        {
                temp_x = PLM(temp_x, mu_vec[threadIdx.x]);
                C[ite*threadIdx.x + i] = (uint8_T)(255 * temp);
        }
}
```

Figure 4.
Parallel chaotic system implementation (Chaos.cu file).

operation which is an involution. Afterwards we use the chaotic S-box to include confusion in the signal:

$$I_c(i,j,k) = S_{box}[c_3(k) \bigoplus c_2(j) \bigoplus c_1(i) \bigoplus I(i,j,k)] \tag{2}$$

Eq. (2) is implemented in the CUDA language as it is shown in **Figure 6**.

To use the chaotic and encrypted function from Matlab, we compile from the terminal, in this case, using nvcc compiler with the lines in **Figure 7**:

```
function [S, invS] = Chaotic_SBox(x, mu)
    S = zeros(16*16,1);
    invS = zeros(16*16,1);
    Q = 0:255;
    max = 2^32;
    for i = 0:255
        x = PLM(x, mu);
        d = mod(floor(max*x), 256);
        j = mod(d,256 - i);
        S(i+1) = Q(j + 1);
        invS(Q(j+1) + 1) = i;
        Q(j+1) = Q(256-i);
    end
end
```

Figure 5.
Chaotic substitution box.

```
#include "tmwtypes.h"

__global__ void Encrypt(uint8_T * I, uint8_T * C, const int ite, uint8_T *S)
{
        // Alias
        int x = blockIdx.x;
        int y = threadIdx.y;
        int z = blockIdx.z;

        // Calculate image index
        int index = z * blockDim.y * gridDim.x + y *gridDim.x + x;

        // Encrypt
        I[index] = C[x] ^ C[ite + y] ^ C[2*ite + z] ^ I[index];
        I[index] = S[(int)I[index]];
}
```

Figure 6.
Parallel chaotic encryption (Encrypt.cu file).

```
nvcc -ptx Encrypt.cu -ccbin "C:\Program Files\Microsoft Visual Studio 12.0\VC\bin"
nvcc -ptx Chaos.cu -ccbin "C:\Program Files\Microsoft Visual Studio 12.0\VC\bin"
```

Figure 7.
Compilation of the CUDA files.

In **Figure 8**, we show the code in Matlab for encrypting HI in parallel. Notice that the first step in **Figure 8** is to load the HI in the variable I in the GPU memory (Igpu), and then we calculate the chaotic S-box in the variable S with the code in **Figure 5**; after that, we simulate the chaotic systems in the variable Cgpu with the code in **Figure 4**. Finally, we encrypt the HI with **Figure 6** gathering the result in the Ic variable.

The decryption scheme of the HI is almost symmetrical. This could be achieved with Eq. (3) which is implemented in **Figure 9**. The code in **Figure 9** is very close to the one in **Figure 6**, but notice that the inverse of the S-box is used and the XOR of the chaotic signal are in inverse order:

$$I(i,j,k) = c_1(i) \bigoplus c_2(j) \bigoplus c_3(k) \bigoplus S_{box}^{-1}[I_c(i,j,k)] \tag{3}$$

Using Eq. (2) in Eq. (3) we obtain the development in Eqs. (4)–(6) that shows how the S-box is canceled with the inverse S-box and how the chaotic signals are canceled because the XOR operation is an involution:

```
function Ic = Parallel_encryption(I, key)

    %% Size of hyperspectral image
    [i, j, k] = size(I);

    %% Load to GPU
    Igpu = gpuArray(I);

    %% Calculate Sustitution-Box
    [S, ~] = Chaotic_SBox(key(1), key(2));

    %% Parallel Simulation chaotic systems
    ite = max([i, j, k]);
    C = uint8(zeros(ite, 3));
    Cgpu = gpuArray(C);
    kernel = parallel.gpu.CUDAKernel('Chaos.ptx', 'Chaos.cu', 'Chaos');
    kernel.ThreadBlockSize = [3,1,1];
    Cgpu = feval(kernel, Cgpu, gpuArray([key(3); key(5); key(7)]),
                 gpuArray([key(4); key(6); key(8)]), ite);


    %% Parallel encryption
    kernel = parallel.gpu.CUDAKernel('Encrypt.ptx', 'Encrypt.cu', 'Encrypt');
    kernel.ThreadBlockSize = [1, j, 1];
    kernel.GridSize = [i, 1, k];
    result = feval(kernel, Igpu, Cgpu, ite, gpuArray(uint8(S)));

    %% Download to CPU
    Ic = gather(result);
end
```

Figure 8.
Encryption of the HI.

```
#include "tmwtypes.h"

__global__ void Decrypt(uint8_T * I, uint8_T * C, const int ite, uint8_T *invS)
{
        // Alias
        int x = blockIdx.x;
        int y = threadIdx.y;
        int z = blockIdx.z;

        // Calculate image index
        int index = z * blockDim.y * gridDim.x + y *gridDim.x + x;

        // Decrypt
        I[index] = invS[(int)I[index]];
        I[index] = C[2*ite + z] ^ C[ite + y] ^ C[x] ^ I[index];
}
```

Figure 9.
Parallel decryption function (Decrypt.cu).

$$I(i,j,k) = c_1(i) \bigoplus c_2(j) \bigoplus c_3(k) \bigoplus S_{box}^{-1}[S_{box}[c_3(k) \bigoplus c_2(j) \bigoplus c_1(i) \bigoplus I(i,j,k)]] \quad (4)$$

$$I(i,j,k) = c_1(i) \bigoplus c_2(j) \bigoplus c_3(k) \bigoplus c_3(k) \bigoplus c_2(j) \bigoplus c_1(i) \bigoplus I(i,j,k) \quad (5)$$

$$I(i,j,k) = I(i,j,k) \quad (6)$$

5. Experiments and results

In this section, we present the result of six new experiments. In **Table 1**, we report the references and the size of the HI.

Number	Name and reference	Size
1	San Francisco[1]	702 × 1000 × 148
2	Urban [35, 36]	307 × 307 × 210
3	Indian pine [37]	145 × 145 × 220
4	Jasper Ridge [35, 36]	100 × 100 × 224
5	University of Pavia[2]	610 × 340 × 103
6	Samson [35, 36]	95 × 95 × 156

[1]*From Ref. [38].*
[2]*From Ref. [39].*

Table 1.
HI data set.

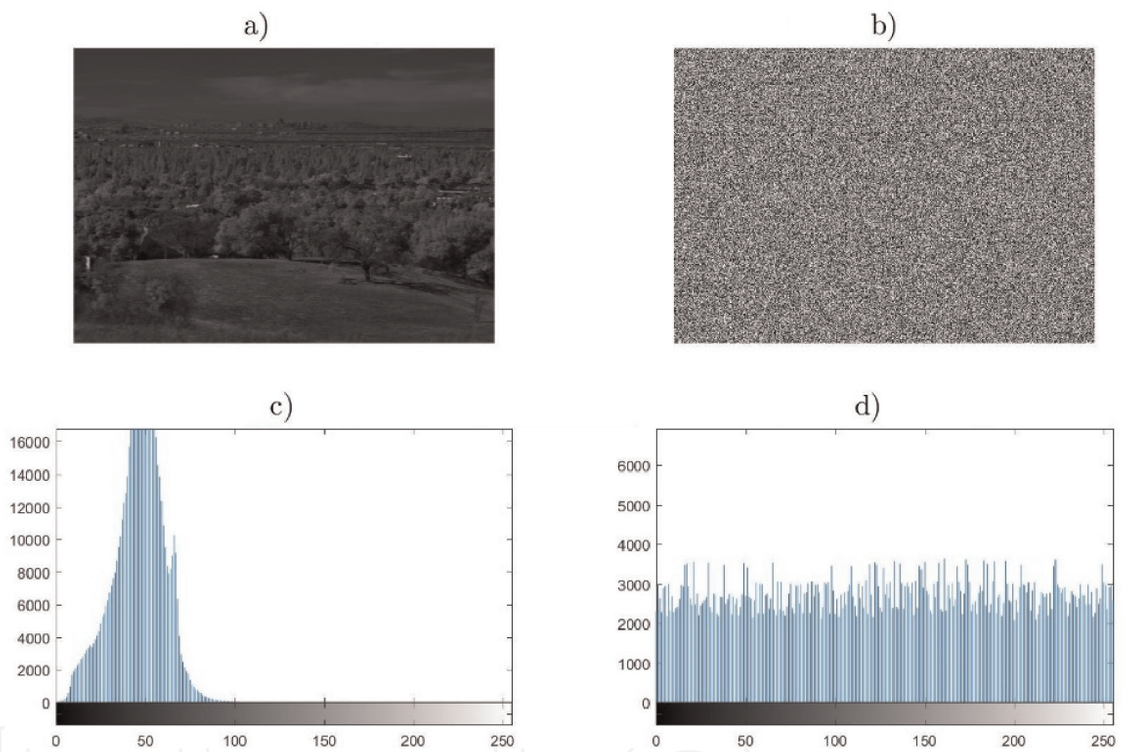

Figure 10.
Experiment 1: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

In the first experiment, we present the results of encrypting the "San Francisco" image. In **Figure 10**, we present a comparison of the ninety third channel of the original image and the cipher one. The image was loaded to the GPU in 0.4439 s and encrypted in 0.0094 s and then downloaded to the CPU memory in 0.21585 s.

The original image has an entropy of 6.1057; after the encryption, we get an entropy of 7.9985. In **Figure 11**, we show a comparison of the hyper-histograms of the original and encrypted image.

In the second experiment, we present the results of encrypting the "Urban" image. In **Figure 12**, we present a comparison of the sixtieth channel of the original image and the cipher one. The image was loaded to the GPU in 0.0118 s and encrypted in 0.0023 s and then downloaded to the CPU memory in 0.0435 s.

The original image has an entropy of 6.7452; after the encryption, we get an entropy of 7.9987. In **Figure 13**, we show a comparison of the hyper-histograms of the original and encrypted image.

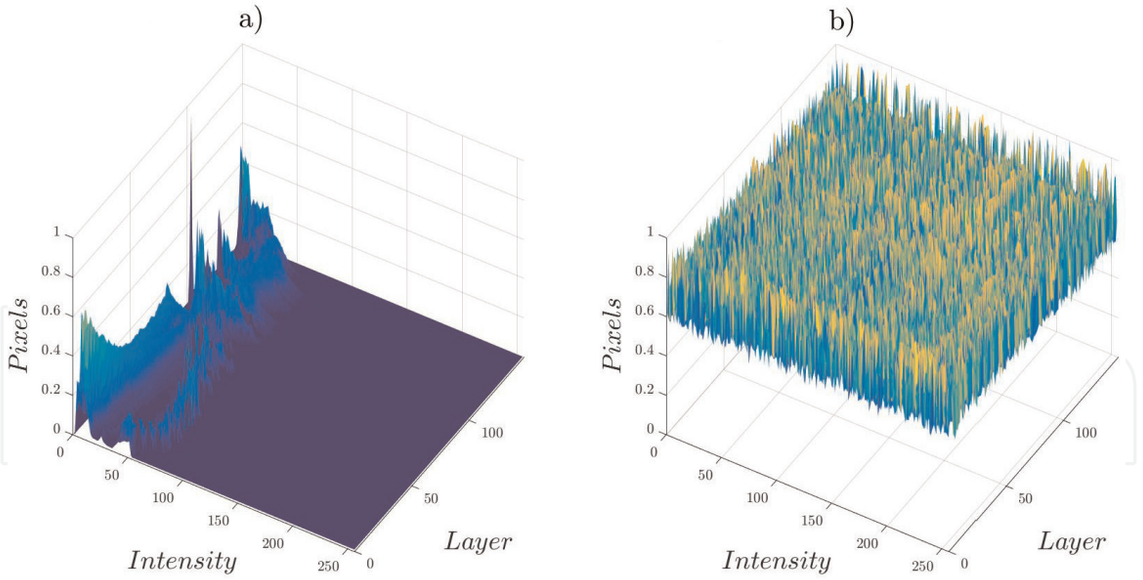

Figure 11.
Experiment 1: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

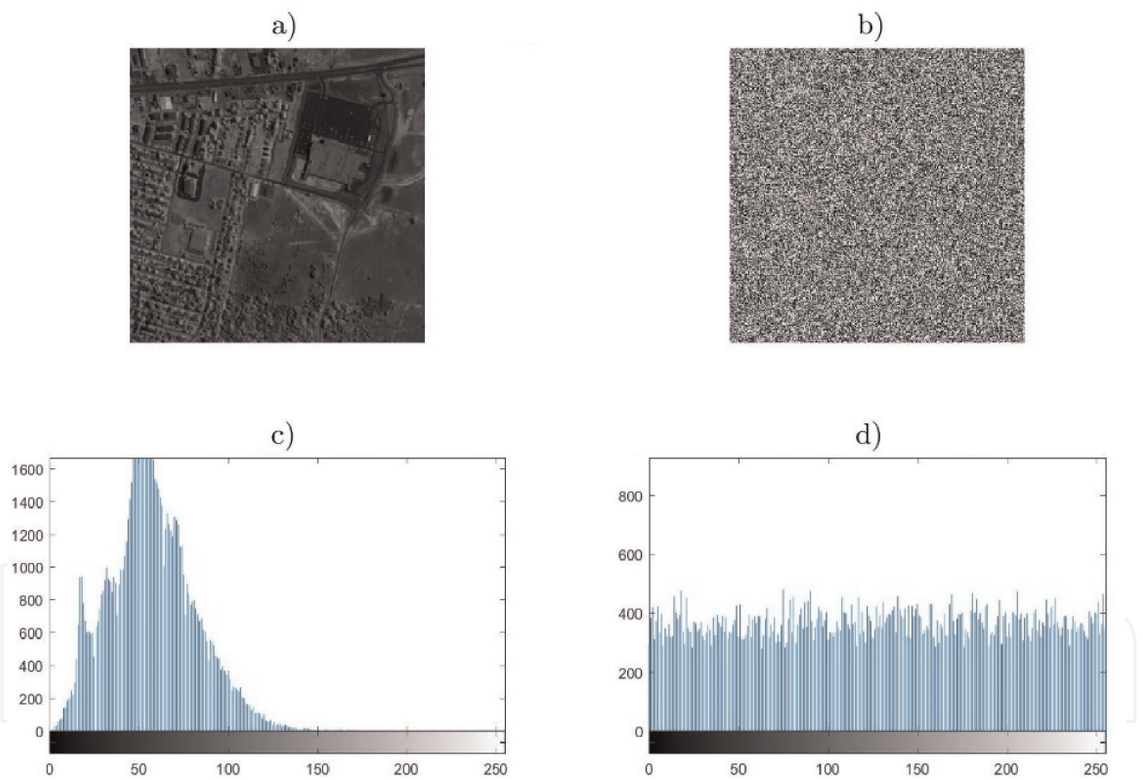

Figure 12.
Experiment 2: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

In experiment 3, we present the results of encrypting the "Indian pine" image. In **Figure 14**, we present a comparison of the forty-fifth channel of the original image and the cipher one. The image was loaded to the GPU in 0.0035 s and encrypted in 0.0010 s and then downloaded to the CPU memory in 0.0124 s.

The original image has an entropy of 6.3325; after the encryption, we get an entropy of 7.9985. In **Figure 15**, we show a comparison of the hyper-histograms of the original and encrypted image.

In experiment 4, we present the results of encrypting the "Jasper Ridge" image. In **Figure 16**, we present a comparison of the forty seventh channel of the original

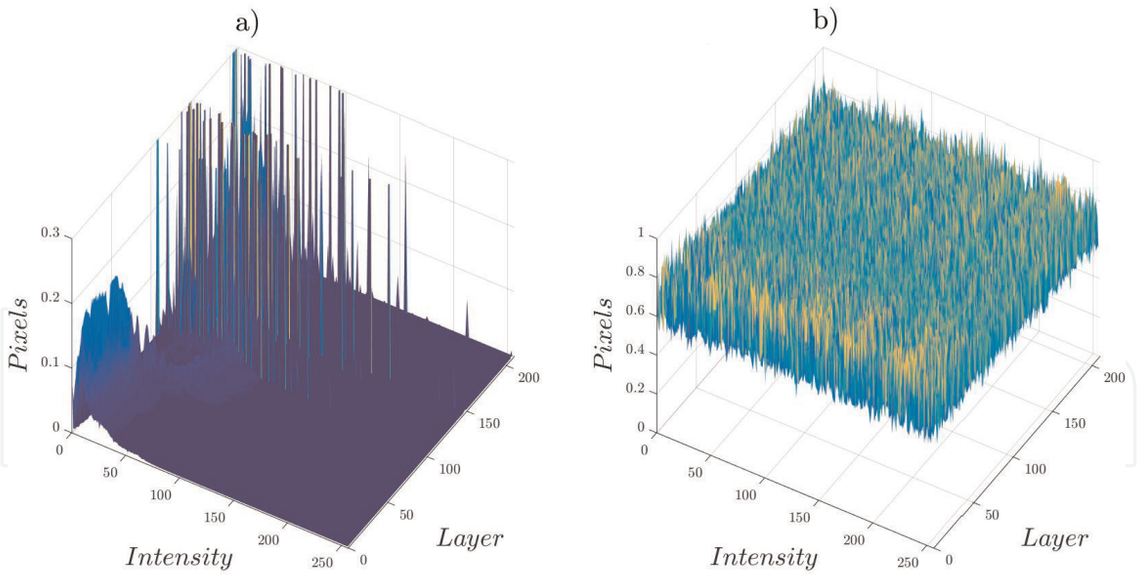

Figure 13.
Experiment 2: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

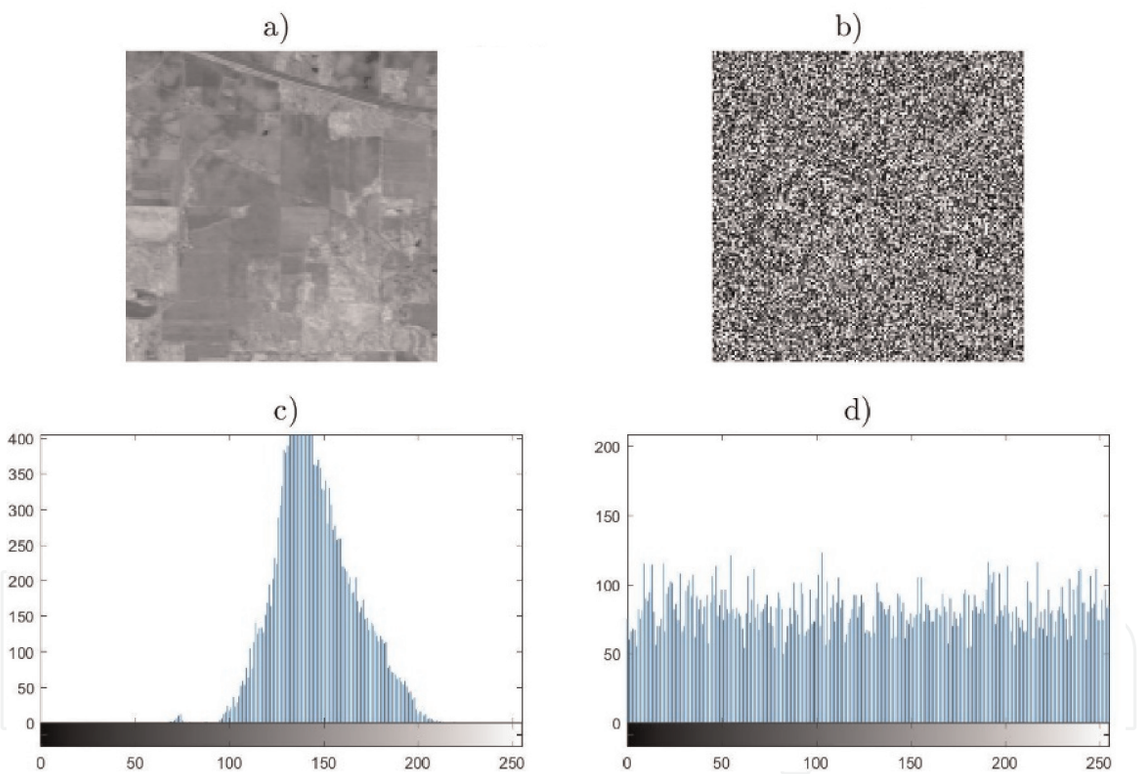

Figure 14.
Experiment 3: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

image and the cipher one. The image was loaded to the GPU in 0.0020 s and encrypted in 0.009 s and then downloaded to the CPU memory in 0.0055 s.

The original image has an entropy of 6.8918; after the encryption, we get an entropy of 7.9983. In **Figure 17**, we show a comparison of the hyper-histograms of the original and encrypted image.

In experiment 5, we present the results of encrypting the "University of Pavia" image. In **Figure 18**, we present a comparison of the 62nd channel of the original image and the cipher one. The image was loaded to the GPU in 0.0122 s and encrypted in 0.0026 s and then downloaded to the CPU memory in 0.0545 s.

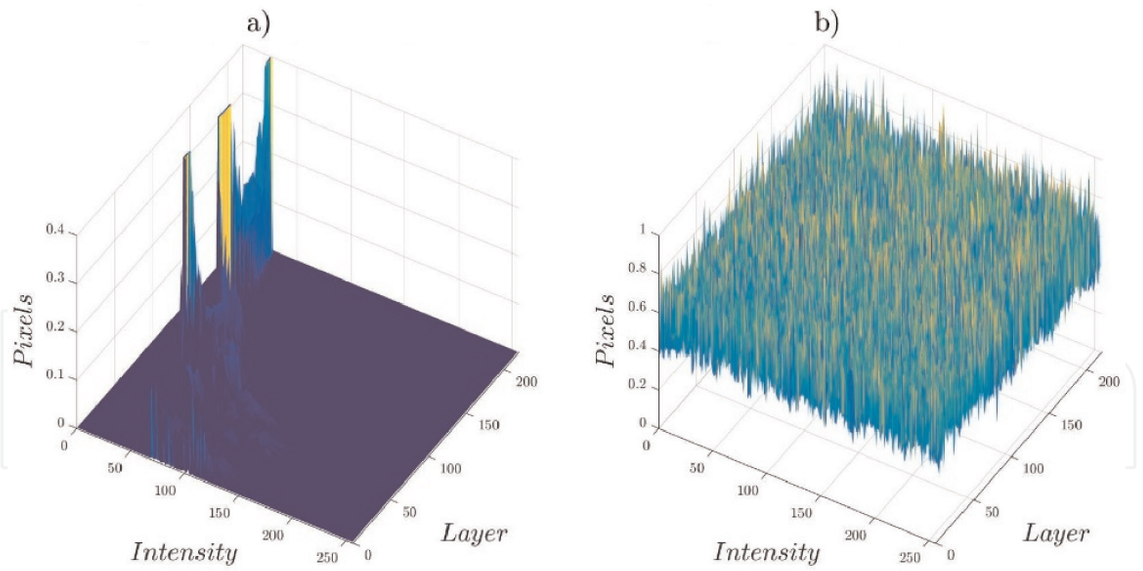

Figure 15.
Experiment 3: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

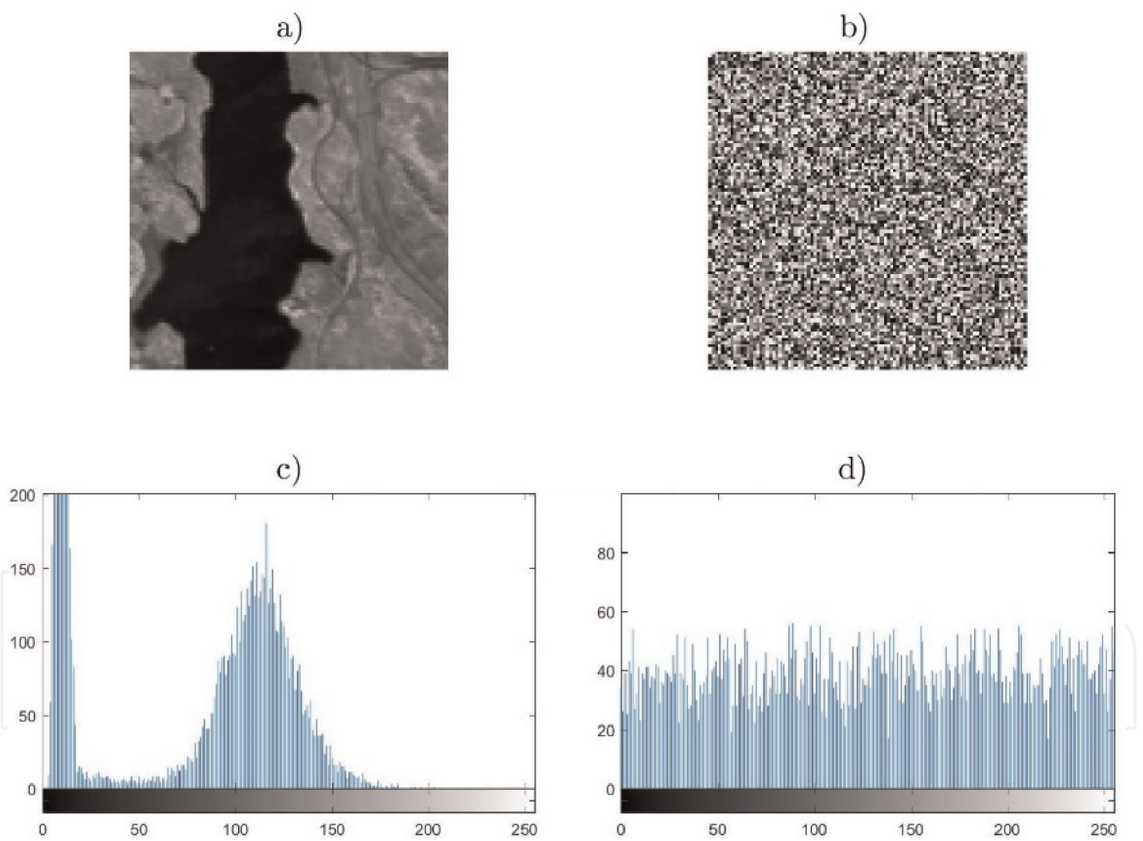

Figure 16.
Experiment 4: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

The original image has an entropy of 6.6827; after the encryption, we get an entropy of 7.9986. In **Figure 19**, we show a comparison of the hyper-histograms of the original and encrypted image.

In experiment 6, we present the results of encrypting the "Samson" image. In **Figure 20**, we present a comparison of the hundredth channel of the original image and the cipher one. The image was loaded to the GPU in 0.0013 s and encrypted in 0.0007 s and then downloaded to the CPU memory in 0.0036 s.

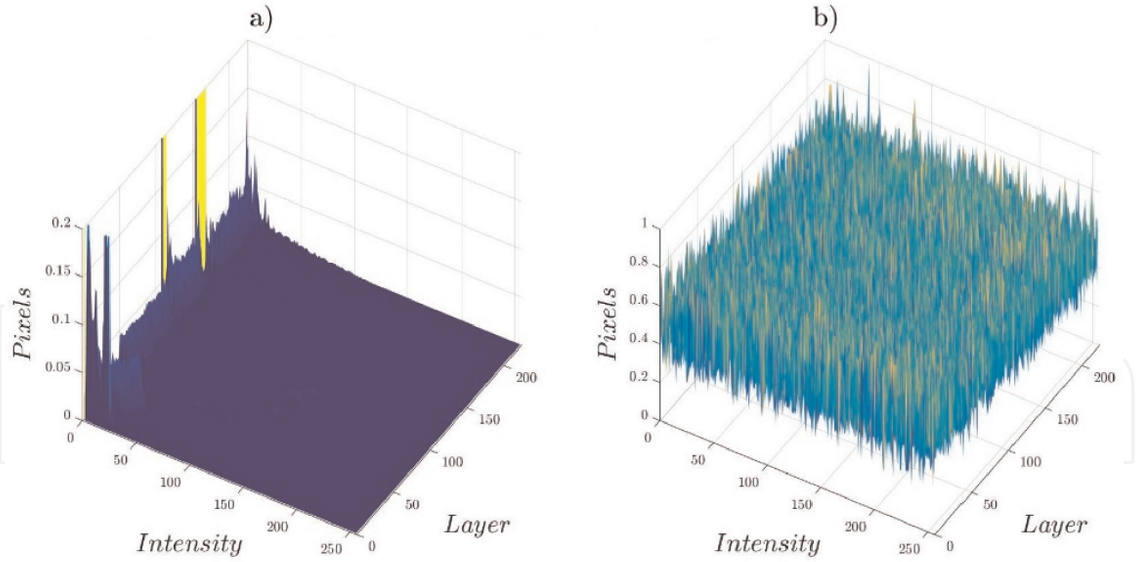

Figure 17.
Experiment 4: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

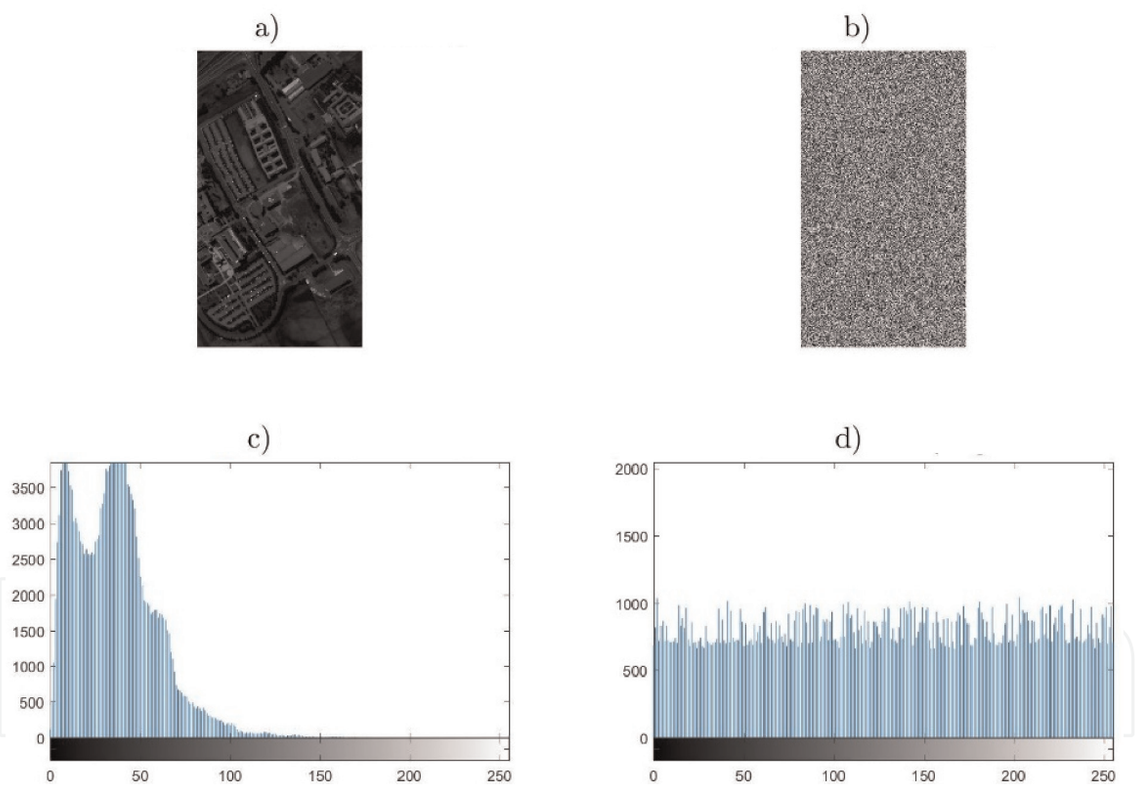

Figure 18.
Experiment 5: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

The original image has an entropy of 6.5833; after the encryption, we get an entropy of 7.9981. In **Figure 21**, we show a comparison of the hyper-histograms of the original and encrypted image.

In **Table 2**, we summarize the results of the encryption. Notice that the encryption process results in high entropy close to eight (the maximum for 2^8 representation). The entropy is a measure of uncertainty defined in Eq. (7), where s_i is a symbol or codification and $P(s_i)$ is the probability of s_i to appear. The number of bits for the representation, in this case eight, is depicted by b:

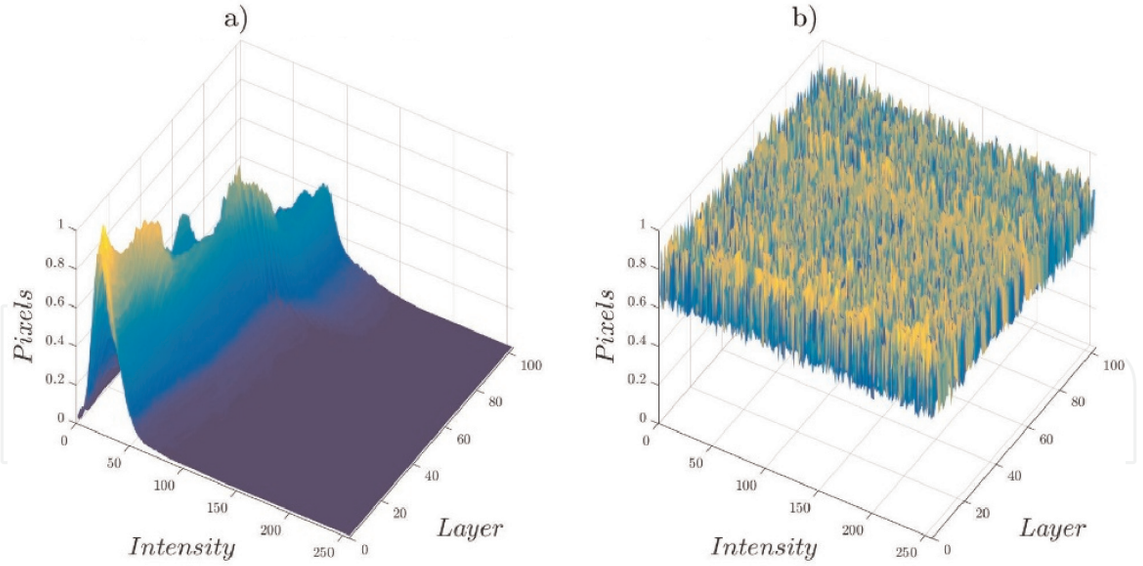

Figure 19.
Experiment 5: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

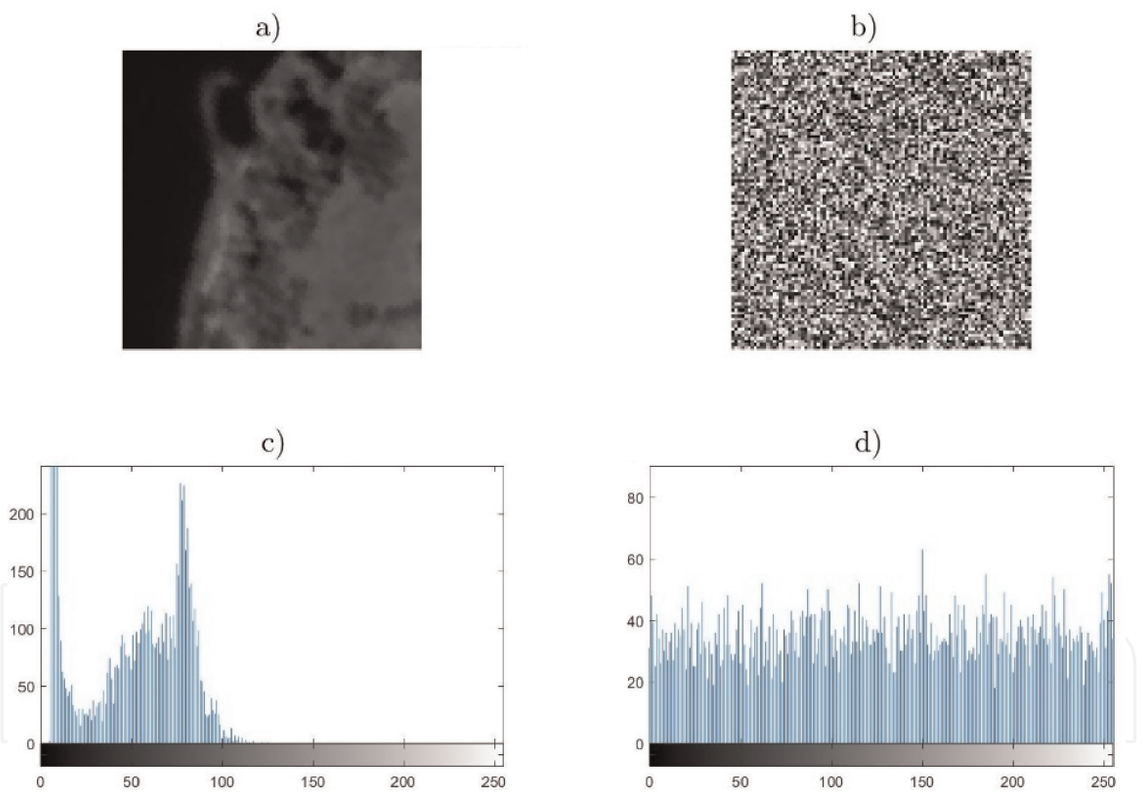

Figure 20.
Experiment 6: image comparison. a) Original image. b) Encrypted image. c) Original histogram. d) Encrypted histogram.

$$H(s) = -\sum_{i=0}^{2^b-1} P(s_i)\log_2(P(s_i)) \tag{7}$$

The proposed algorithm takes advantage of the multidimensionality nature of the HI. This leads to a high-performance algorithm that can encrypt images quickly. Notice that the load to GPU and download to CPU processes take most of the time. This could be avoided with a direct acquisition of the HI from the GPU, but this problem is beyond the scope of this chapter.

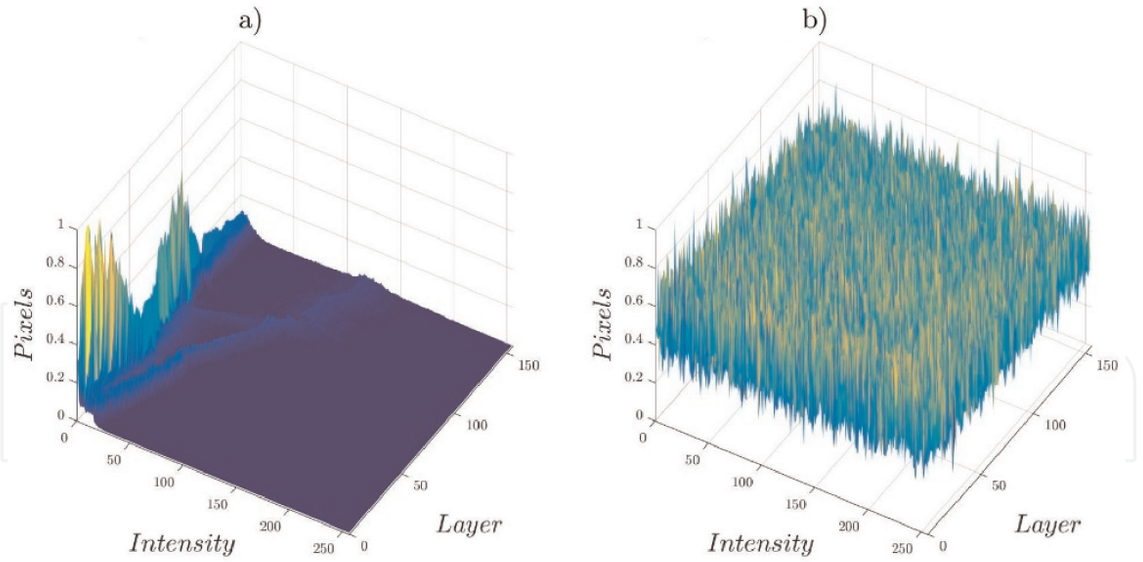

Figure 21.
Experiment 6: hyper-histogram comparison. a) Original Hyperhistogram. b) Encrypted Hyperhistogram.

Number	Name	Load (s)	Encrypt (s)	Download (s)	Total (s)	Original entropy	Encrypted entropy	Decrypt (s)
1	San Francisco	0.4439	0.0099	0.2158	0.6696	6.1057	7.9985	0.0095
2	Urban	0.0118	0.0023	0.0435	0.0576	6.7452	7.9987	0.0031
3	Indian pine	0.0035	0.0010	0.0124	0.0169	6.3325	7.9985	0.0012
4	Jasper Ridge	0.0020	0.0009	0.0055	0.0084	6.8918	7.9983	0.0009
5	University of Pavia	0.0122	0.0026	0.0545	0.0693	6.6827	7.9986	0.0023
6	Samson	0.0013	0.0007	0.0036	0.0056	6.5833	7.9981	0.0008

Table 2.
Encryption results.

6. Conclusions and future work

In this chapter, we have reviewed the proposed method in [8] with a full review of the source code. The proposed algorithm is a parallel encryption scheme specific for hyperspectral images. This scheme takes advantage of the multidimensionality nature of the HI, where every pixel of every channel is encrypted in parallel. These features lead to a high-performance algorithm capable of encrypting HI in a short time.

The algorithm uses four chaotic systems, the first three for generating a chaotic cube mask of the same size of the HI and the fourth one to generate a chaotic S-box to aggregate confusion in the ciphertext. As the experimentation has shown, the algorithm can generate high-entropy ciphertext. This algorithm is recommended for fast encryption in ciphertext-only attacks.

As future work, most of the computational time is wasted in the load to the memory of the HI image. A system to load directly to the GPU memory is desirable. Also, we recommend the readers to explore new ways of combining different chaotic signals for high-dimensional masks. Furthermore, an extension of this algorithm for working with a compatible compression algorithm is desired.

DOI: http://dx.doi.org/10.5772/intechopen.88980

Author details

Carlos Villaseñor, Javier Gomez-Avila, Nancy Arana-Daniel*, Alma Y. Alanis and Carlos Lopez-Franco
Centro Universitario de Ciencias Exactas e Ingenierías, Universidad de Guadalajara, Guadalajara, Mexico

*Address all correspondence to: nancyaranad@gmail.com

References

[1] Shippert P. Why use hyperspectral imagery? Photogrammetric Engineering and Remote Sensing. 2004;**70**(4):377-396

[2] Paar C, Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners. Berlin/Heidelberg, Germany: Springer Science & Business Media; 2009

[3] Biryukov A, Khovratovich D. Related-key cryptanalysis of the full AES-192 and AES-256. In: International Conference on the Theory and Application of Cryptology and Information Security. New York, USA: Springer; 2009. pp. 1-18

[4] Thulasimani L, Madheswaran M. A single chip design and implementation of aes-128/192/256 encryption algorithms. International Journal of Engineering, Science and Technology. 2010;**2**(5):1052-1059

[5] Moh'd A, Jararweh Y, Tawalbeh L. AES-512: 512-bit advanced encryption standard algorithm design and evaluation. In: 2011 7th International Conference on Information Assurance and Security (IAS). 2011

[6] Somani U, Lakhani K, Mundra M. Implementing digital signature with RSA encryption algorithm to enhance the data security of cloud in cloud computing. In: 2010 First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010). 2010

[7] Kocarev L. Chaos-based cryptography: A brief overview. IEEE Circuits and Systems Magazine. 2001;**1**(3):6-21

[8] Villaseñor C, Gutierrez-Frias EF, Arana-Daniel N, Alanis AY, Lopez-Franco C. Parallel crossed chaotic encryption for hyperspectral images. Applied Sciences. 2018;**8**:1-12

[9] Grahn H, Geladi P. Techniques and Applications of Hyperspectral Image Analysis. Hoboken, New Jersey, USA: John Wiley & Sons; 2007

[10] Alam MS, Sidike P. Trends in oil spill detection via hyperspectral imaging. 2012 7th International Conference on Electrical and Computer Engineering. 2012

[11] Goetz AF, Vane G, Solomon JE, Rock BN. Imaging spectrometry for earth remote sensing. Science. 1985; **228**(4704):1147-1153

[12] Govender M, Chetty K, Bulcock H. A review of hyperspectral remote sensing and its application in vegetation and water resource studies. Water Research Commission (WRC). 2007;**33**(2):145-152

[13] Adam E, Mutanga O, Rugege D. Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: A review. Wetlands Ecology and Management. 2010;**18**(3):281-296

[14] Liang H. Advances in multispectral and hyperspectral imaging for archaeology and art conservation. Applied Physics A. 2012;**106**(2):309-323

[15] Fischer C, Kakoulli I. Multispectral and hyperspectral imaging technologies in conservation: Current research and potential applications. Studies in Conservation. 2006;**51**(1):3-16

[16] Padoan R, Steemers T, Klein M, Aalderink B, De Bruin G. Quantitative hyperspectral imaging of historical documents: Technique and applications. In: Art Proceedings. 2008. pp. 25-30

[17] Clark RN, Swayze GA. Mapping minerals, amorphous materials, environmental materials, vegetation, water, ice and snow, and other materials: The USGS Tricorder algorithm. 1995

[18] Carrasco O, Gomez RB, Chainani A, Roper WE. Hyperspectral imaging

applied to medical diagnoses and food safety. In: Geo-Spatial and Temporal Image and Data Exploitation III. 2003

[19] Lu G, Fei B. Medical hyperspectral imaging: A review. Journal of Biomedical Optics. 2014;**19**(1):1-24

[20] Afromowitz MA, Callis JB, Heimbach DM, DeSoto LA, Norton MK. Multispectral imaging of burn wounds: A new clinical instrument for evaluating burn depth. IEEE Transactions on Biomedical Engineering. 1988;**35**(10): 842-850

[21] Gowen AA, O'Donnell CP, Cullen PJ, Downey G, Frias JM. Hyperspectral imaging—An emerging process analytical tool for food quality and safety control. Trends in Food Science and Technology. 2007;**18**(12): 590-598

[22] Feng Y-Z, Sun D-W. Application of hyperspectral imaging in food safety inspection and control: A review. Critical Reviews in Food Science and Nutrition. 2012;**52**(11):1039-1058

[23] Kuula J, Pölönen I, Puupponen H-H, Selander T, Reinikainen T, Kalenius T, et al. Using VIS/NIR and IR spectral cameras for detecting and separating crime scene details. In: Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense XI, International Society for Optics and Photonics. 2012

[24] Edelman G, Gaston E, Van Leeuwen T, Cullen P, Aalders M. Hyperspectral imaging for non-contact analysis of forensic traces. Forensic Science International. 2012; **233**(3):28-39

[25] Obregón-Pulido G, Gasca A, Solis-Perales G. Secure communications using different type of chaotic systems. Multimedia, Communication and Computing Application. 2015;**1**(2):1-13

[26] Pecora LM, Carroll TL. Synchronization in chaotic systems. Physical Review Letters. 1990; **64**(1990):8-21

[27] Zhang L, Zhang Y. Research on lorenz chaotic stream cipher. In: Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology 2005. 2005

[28] Jia X. Image encryption using the Ikeda map. In: 2010 International Conference on Intelligent Computing and Cognitive Informatics. 2010

[29] Sathishkumar G, Bhoopathy K, Sriraam N. Image encryption based on diffusion and multiple chaotic maps. International Journal of Network Security & Its Applications. 2011;**3**(2): 181-194

[30] Sankpal PR, Vijaya P. Image encryption using chaotic maps: A survey. In: 2014 Fifth International Conference on Signal and Image Processing; IEEE. 2014. pp. 102-107

[31] Su Z, Lian S, Zhang G, Jiang J. Chaos-based video encryption. In: Chaos-Based Cryptography. New York, USA: Springer; 2011. pp. 205-226

[32] Shang F, Sun K, Cai Y. An efficient MPEG video encryption scheme based on chaotic cipher. In: 2008 Congress on Image and Signal Processing; IEEE. 2008. pp. 12-16

[33] Lian S, Sun J, Wang Z, Dai Y. A fast video encryption scheme based-on chaos. In: ICARCV 2004 8th Control, Automation, Robotics and Vision Conference; IEEE. 2004. pp. 126-131

[34] Mroczkowski P. Generating pseudorandom S-boxes—A method of improving the security of cryptosystems based on block ciphers. Journal of Telecommunications and Information Technology. 2009;**2**:74-79

[35] Zhu F, Wang Y, Xiang S, Fan B, Pan C. Structured sparse method for hyperspectral unmixing. ISPRS Journal of Photogrammetry and Remote Sensing. 2014;**88**:101-118

[36] Zhu F, Wang Y, Xiang S, Fan B, Pan C. Spectral unmixing via data-guided sparsity. IEEE Transactions on Image Processing. 2014:5412-5427

[37] Baumgardner MF, Biehl LL, Landgrebe DA. 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. [Online]. 2015. Available from: https://purr.purdue.edu/publications/1947/1 [Accessed: 10 June 2019]

[38] Database 4 of imageVal Consulting. Available from: http://www.imageval.com

[39] Hyperspectral Remote Sensing Scenes. Available from: http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes