A Many-key Image Encryption Method Using the Lorenz System

Ronald Marsh and Scott Kerlin

Department of Computer Science
University of North Dakota
Grand Forks, ND 58203
rmarsh@cs.und.edu

Abstract

A many-key chaotic image encryption scheme based on the Lorenz system is presented. Using the time-delay Lorenz system, the proposed approach performed the pixel permutation on a bit-by-bit basis in the diagonal and antidiagonal directions, providing the necessary pixel confusion. Then, the Lorenz system is employed again to generate a sequence of random bytes that are used in the diffusion step. Numerical analysis is also presented showing that the system (with diffusion) should be resistant to any type of attack. We also show that, even though we have a very large key space, that without the diffusion step, the method could be susceptible to an attack.

Introduction

CubeSats [1] have generated significant interest due to their low cost and ease of launch vehicle integration. However, despite their small size (approximately 10 cm x 10 cm x 10 cm and a mass of 1.33 kg), CubeSats contain numerous systems and subsystems and can be designed to accomplish a wide variety of tasks. A student-run CubeSat program at the University of North Dakota (UND), called the Open Prototype for Educational Nanosats (OPEN) [2], is an interdisciplinary, CubeSat program whose goal is to provide students with the opportunity to get involved in all aspects and levels of a spacecraft development program. The OPEN program has a second mission and this is to create a CubeSat design that is very affordable (< $5000.00). To help keep costs down Cubesats commonly use amateur radio frequencies; frequencies that FCC regulations ban using any form of encryption with [3]. However, given the growing popularity of Cubesats and the increased launch opportunities [4, 5], we expect that the need for data confidentiality will outweigh the FCC's restrictions. Finally, as the UND Cubesat will be an Earth observing satellite we are interested in developing a very secure image encryption methodology.

Cryptography has a long history. Julius Caesar is credited with creating one of the earliest cryptographic systems to send military messages to his generals. Governments and military leaders have used cryptographic systems since. Focusing on image encryption, our domain of interest, a variety of different encryption methods have been proposed, including the Hartley transform and Logistic map [6], chaotic maps (chaos-based) [7], and a circular bit shift and XOR operations [8].

Perhaps the most active area right now is that of the chaos-based algorithms. These algorithms exhibit some exceptionally good properties in many aspects regarding encryption, such as complexity, non-periodicity, speed, and sensitivity to the initial conditions. These properties make this approach ideal for images as the more traditional encryption algorithms such as DES, IDEA and RSA as it is rather difficult to efficiently shuffle and diffuse data by these traditional means of encryption. As noted, the chaos-based algorithms are also highly sensitive to the initial conditions, meaning that even a slight change to the key will result in a significant change to the encrypted image. All of these characteristics lead to efficient methods for image encryption. Which is why we selected this approach.

Chaotic systems

In 1999, Professor G. Chen first presented a three-dimensional (3D) chaotic system [9], which improved on the small key space and weak security known to exist in the one-dimensional (1D) chaotic systems [10].

The system Chen presented is described by equation 1:

$$x' = a(y - x)$$
$$y' = (c - a)x - xz + cy \quad (1)$$
$$z' = xy - bz$$

where *a*, *b* and *c* are parameters. Choosing a=35, b=3, c ∈ [20, 28.4], the system enters a chaotic domain.

The system we are using is a variation of Chen's and is a Lorenz system based on the work of Huang, Ye, and Wong [11]. This system is described by equation 2:

$$x' = m(y - x)$$
$$y' = rx - y - xz \quad (2)$$
$$z' = xy - bz$$

where *m*, *b* and *r* are parameters. Choosing m=10, r=28, b =8/3, the system enters a chaotic domain. Thus, given initial values x_o, y_o, and z_o, the system will quickly diverge and generate values vastly different from a system given only slightly different values for x_o, y_o, or z_o [11].

A Many-key Approach

The approach to confusion we are taking is an extension to the approach taken by Jackson, Kerlin, and Straub [12] wherein they employed a chaotic block-based cypher which used diagonal and anti-diagonal shuffling of an image's pixels using a Lorenz System. The core process of their approach is quite simple in the design, but highly effective as a 1-way function. First the input data is read into a *n x n* matrix (with padding or subdividing as needed/desired). Then a 4-value key is used to seed a Lorenz System, the output of which is used to shuffle the data matrix against first its diagonals and then its anti-diagonals. The first critical point, with respect to this shuffling, is that the *n x n* matrix is treated as a circulant matrix, in which all diagonals and anti-diagonals will have *n* members. This permits the resulting diagonal and anti-diagonal matrices to be *n x n*, thus preserving the dimensional uniformity of the matrix throughout the translations while still providing significant entropy. The second critical point, with respect to the shuffling, is the use of a Lorenz System. Lorenz Systems have the property of being highly divergent for similar seeds, this makes them highly chaotic and an excellent 1-way function for the purposes of encryption. Jackson, Kerlin and Straub [12] utilized a Lorenz System requiring three inputs, so their keys end up being four values: three values are used as inputs (or seeds: x_o, y_o, and z_o) to the Lorenz System and the fourth value is used for number of iterations through the Lorenz System before output is taken (i.e. it controls how divergent/chaotic the values are). Using the output from the Lorenz System, each datum in the *n x n* matrix is provided with a value. The datum are then sorted (by Lorenz value *x*) along their circulant diagonals. This process of sorting is then repeated, but now the datum are sorted (by Lorenz value *y*) along their circulant anti-diagonals. Post shuffling, a block-based diffusion is performed in order to increase the entropy of the

output. This diffusion process uses modular arithmetic as a 1-way function for creating a different multiplicative for use on each block of the data matrix.

In addition to diagonal and anti-diagonal shuffling of an image's pixels one will also want to introduce some level of pixel diffusion into an encryption algorithm. There are two reasons to do this. The first is that the diffusion process can render the discretized chaotic map non-invertible. The second is that the diffusion process can significantly change the statistical properties of the image (e.g. the histogram). For example, the top row of figure 1 shows an image, its histogram, and the diagonal and anti-diagonal shuffled image. The bottom row of figure 1 shows the shuffled image's histogram. As one can see the two histograms are identical. Therefore, for a secure encryption scheme, a diffusion mechanism is desired.

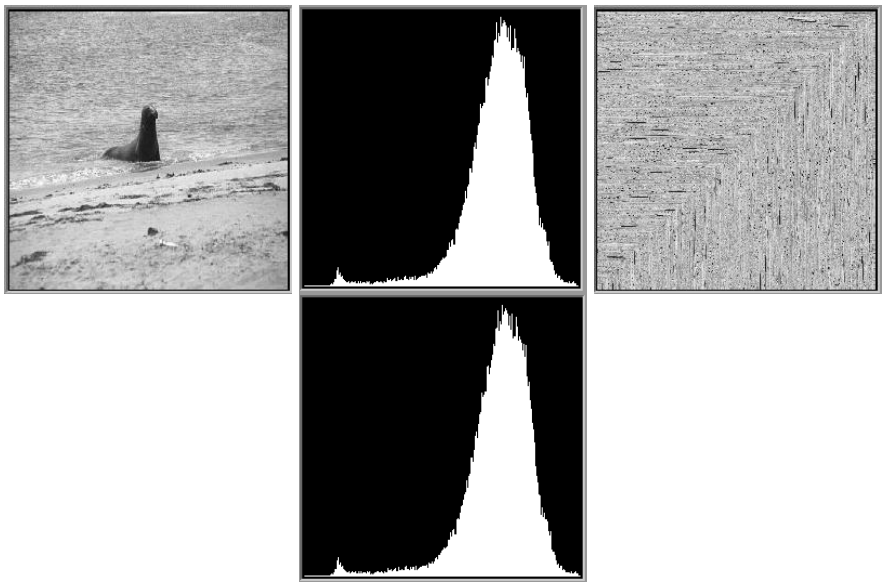

Figure 1: Seal image.

To improve upon the security of an encryption scheme, one can expand upon the idea to use more keys (parameters). Gao et al. [13] increased the number of parameters to six and, as in [14], two Logistic maps were employed to enlarge the key space.

For this work we use multiple keys and multiple Logistic maps. We use eight sets of $n \times n$ circulant matrices, one for each of the eight bits (we are assuming a grayscale 8-bit per pixel image). Therefore, we are using eight sets of four keys. The first four keys are three inputs to the Lorenz System and the number of iterations before output is taken. These values are used to perform the diagonal and anti-diagonal shuffling of the image. However, unlike in the work of Jackson, Kerlin, and Straub [12], we are now performing the shuffling on a bit by bit basis; however, as we will show, a diffusion step is still necessary. In a sense, we are treating the image as a 3D dataset, similar to what Koduru and Chandrasekaran did [15].

The top row of figure 2 shows an image, its histogram, and the diagonal and anti-diagonal shuffled image. The bottom row of figure 1 shows the shuffled image's histogram. If we were to expand this encryption scheme to 24 bit color images, we could employ up to 24 sets of *n x n* circulant matrices and 24 sets of four keys.

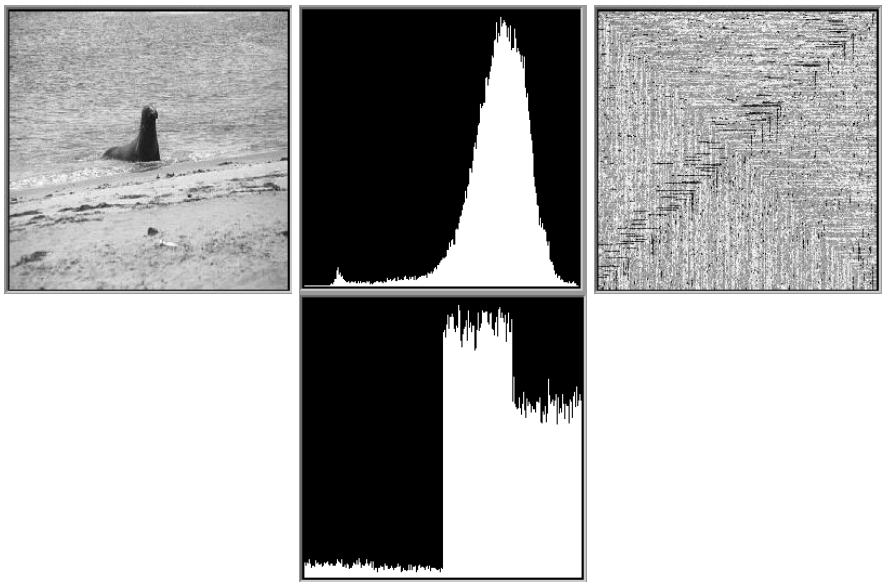

Figure 2: Seal image with bit shuffling.

The approach works well when applied to binary images, images with pixel values of 0 or 255, as well. As with figures 1 and 2, figure 3 shows a binary image, its histogram, the diagonal and anti-diagonal shuffled image, and the shuffled image's histogram.

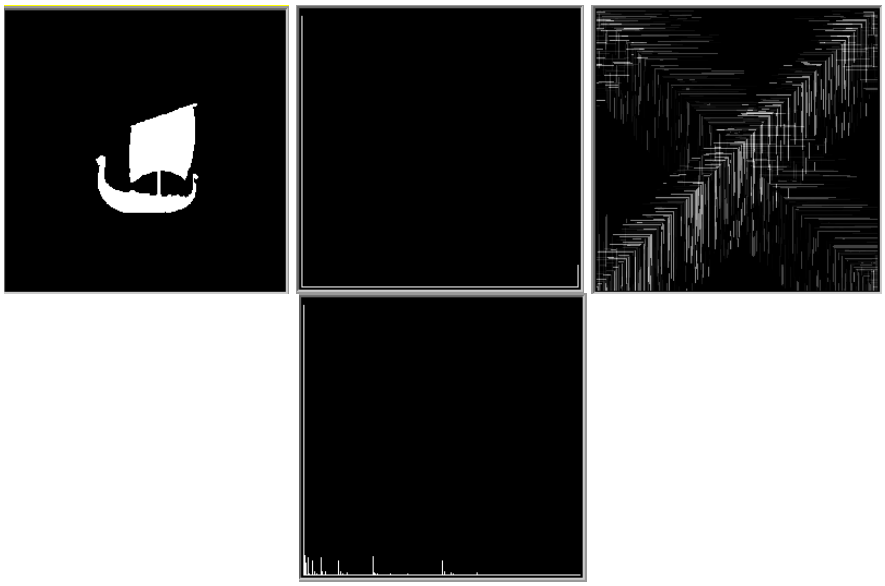

Figure 3: Binary ship image with bit shuffling.

Of course, we can still employ a separate diffusion algorithm. And that component may require another key(s), or that key(s) could be extracted from the existing sets. We chose a rather simple diffusion process, with its inverse, shown in equation 3.

$$\begin{aligned} I_k &= [[C_k + S_k] \bmod N] \oplus C_{k-1} \\ C_k &= [[I_{k-1} \oplus I_k] + N - S_k] \bmod N \end{aligned} \tag{3}$$

Where C_k are the input pixels with C_0 set to 128, N is 256, I_k are the resulting diffused pixels with I_0 set to 128, and S_k is a random value generated by a Lorenz function. The specific function used to generate S is shown in equation 4.

$$S_k = [[(x + y + z) * 10000] \bmod 254] + 1 \tag{4}$$

Where x, y, and z are values generated by the Lorenz function shown in equation 2 above using four new keys.

In the following examples, we applied the diffusion process after the diagonal and anti-diagonal shuffling of the bits. As with the previous figures, figure 4 shows the seal image, its histogram, the diagonal and anti-diagonal shuffled image, and the shuffled image's histogram when the extra diffusion step is applied. Figure 5 shows the binary ship image, its histogram, the diagonal and anti-diagonal shuffled image, and the shuffled image's histogram when the extra diffusion step is applied.

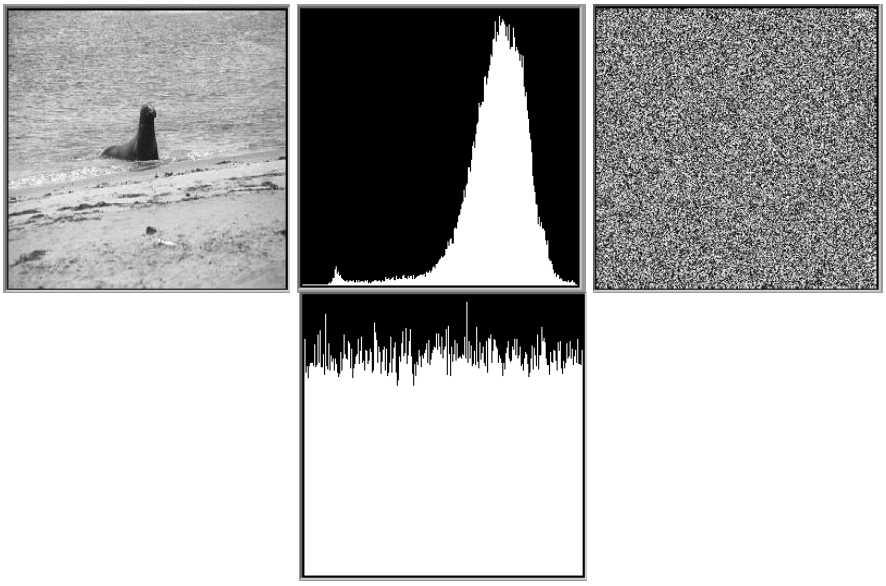

Figure 4: Seal image with bit shuffling and diffusion.

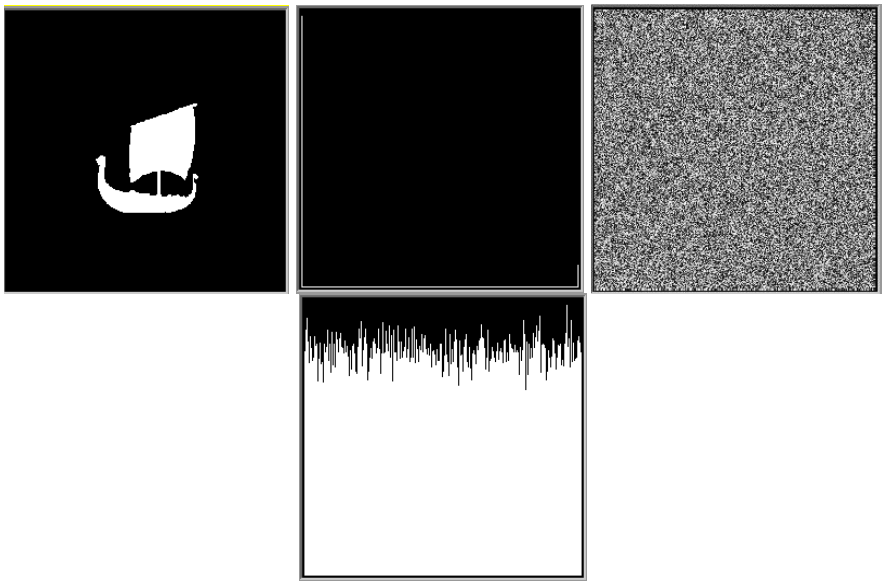

Figure 5: Binary ship image with bit shuffling and diffusion.

Key Analysis

A good encryption algorithm must satisfy several requirements. These include a large key space, sensitivity to the cipher keys, and no correlation between two adjacent pixels.

Key space – In the proposed approach (with diffusion), for a grayscale image, the key space is determined by 9 * 4 * N. 9 sets of 4-valued keys where each value is described by N digits. For example, if we set N to be 8, our overall key size would then be 288*4 digits. If we now assume that we can store each digit (0-9) with 4 bits, we would have an overall key size of 1152 bits. However, as Huang, Ye, and Wong have shown, when using a circulant operation, one can achieve key sensitivity out to the 15^{th} digit [11]. This suggests another way of allocating/storing keys. With this approach one would use floating point values for the first three keys of each set and an integer (byte) value for the forth key of the set. This allocation would create a key set determined by equation 5.

$$9 * (3 * Nf + Nb) \quad (5)$$

Where Nf is the number of bits required to store the floating point value (32 or 64) and Nb is the number of bits required to store the byte value (8).

Assuming C/C++ float types, we would have a key space requiring 9 * (3 * 32 + 8), or 936 bits. For 24-bit color images, this would become 2808 bits.

Sensitivity analysis – Both Huang, Ye, and Wong [11] and Chen, Mao, and Chui [7] have addressed the sensitivity analysis of these types of encryption schemes. Therefore, we will not expand further on that work.

Histogram analysis – The histograms shown in figures 2, 3, 4, and 5 clearly indicate that it would be difficult to employ a statistical attack as the pixel values are evenly distributed (when the added diffusion step is included) or, at least, very different from the original image (when the added diffusion step is not included).

Pixel correlation analysis – We used equation 6 [16] with 1500 randomly selected pixels (500 for each direction) to evaluate the adjacent pixel correlation.

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\sqrt{D(x)D(y)}} \tag{6}$$

Where:

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

$$D(x) = \left(\frac{1}{N}\right)\sum_{i=1}^{N}(x_i - E(x))^2$$

$$E(x) = \left(\frac{1}{N}\right)\sum_{i=1}^{N}x_i$$

And where x and y are the values of the two selected adjacent pixels in the image and N is total number of pixels (500) selected from the image for the calculation. Tables 1-3 show the results for the original image set, the encrypted image set with bit shuffling only, and the encrypted image set with bit shuffling and diffusion. As shown, the values are nearly zero indicating little correlation.

Image	Seal	Cenote	Tetons	Huts	Ship
Horizontal	0.13113	0.20000	0.13169	0.05017	0.12516
Vertical	0.12061	0.03095	0.58492	0.46717	0.21124
Diagonal	0.02523	0.04194	-0.01916	0.00105	-0.00836

Table 1: Correlation coefficients for image set (original).

Image	Seal	Cenote	Tetons	Huts	Ship
Horizontal	-0.02404	0.08899	0.13796	0.13142	0.10398
Vertical	0.06112	-0.02773	0.22178	0.17631	-0.00049
Diagonal	0.00626	-0.09521	-0.00015	-0.00474	-0.04606

Table 2: Correlation coefficients for image set (encrypted).

Image	Seal	Cenote	Tetons	Huts	Ship
Horizontal	0.02504	-0.00812	-0.03923	0.01609	-0.04842
Vertical	0.03934	0.03643	0.06635	-0.00540	0.01244
Diagonal	-0.10148	0.06879	0.00000	0.03436	0.06052

Table 3: Correlation coefficients for image set (encrypted).

Differential attack – We used two methods to evaluate the influence of changing a single pixel by a single bit in the original image on the encrypted. The first method used was the number of pixels change rate (NPCR), the second was the unified averaged changed intensity (UACI) [17]. The NPCR measures the percentage of different pixels between two images. In this case a two-dimensional array *D* is defined to have the same size as the two images C_1 & C_2. Where C_1 is the original unencrypted image and C_2 is a copy of C_1 with one pixel (in our case [128, 128]) changed by 1 bit. We then apply equation 7.

$$NPCR = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M} D(j,i)}{M * N} * 100\% \tag{7}$$

Where:

$$D(j,i) = \begin{cases} 0 \rightarrow C_1(j,i) \neq C_2(j,i) \\ 1 \rightarrow C_1(j,i) = C_2(j,i) \end{cases}$$

And M and N are the size of the encrypted image.

UACI measures the average intensity of differences between two images. UACI is defined by equation 8.

$$UACI = \left[\sum_{j=1}^{N}\sum_{i=1}^{M} \frac{|C_1(j,i) - C_2(j,i)|}{M * N * 255} \right] * 100\% \tag{8}$$

We calculated the NPCR and UACI for the image set by using the many-key encryption scheme, shown in tables 4 and 5. NPCR was over 99% for all cases showing that the encryption scheme is very sensitive to small changes in the unencrypted image. However, when the diffusion step was not included, the results were unsatisfactory. UACI was in the low to mid 30% range again showing that the encryption scheme is very sensitive to small changes in the unencrypted image. However, when the diffusion step was not included, the results were again unsatisfactory.

Image	Seal	Senote	Tetons	Huts	Ship
Value	0.009155	0.003052	0.003052	0.003052	0.001526
Value	99.48425	99.67041	99.646	99.58038	99.03107

Table 4: NPCR coefficients.

Image	Seal	Senote	Tetons	Huts	Ship
Value	0.000377	0.000018	0.000018	0.000018	0.000006
Value	32.93685	32.87933	33.25742	33.67321	35.56585

Table 5: UACI coefficients.

Entropy analysis – Entropy analysis is another tool that can be used to measure the strength of an encryption system. It is defined by equation 8 [18].

$$H(s) = \sum_{i=1}^{N} p(s_i) \log(\frac{1}{p(s_i)}) \quad (8)$$

Where $p(s_i)$ is the probability of occurrence of s_i. Note that for any ideal random sequence H(s) is 8. Table 6 shows the results for the image set with bit shuffling only on the top row and bit shuffling with diffusion on the bottom row. All images benefit from the added diffusion step.

Image	Seal	Cenote	Tetons	Huts	Ship
Value	7.252669	7.481663	7.852982	7.950469	3.010717
Value	7.996749	7.997534	7.997175	7.997135	7.997581

Table 6: Entropy coefficients.

Conclusion

Proposed is a many-key approach using the Lorenz System that incorporates bit by bit diagonal and anti-diagonal shuffling of the pixels in an image, employing a large ley space of 936 bits for grayscale images and 2808 bits for 24-bit color images. However, as demonstrated, a large key space is not enough. If one's desire is to provide maximum security, one must also ensure that some form of diffusion is used. The many-key approach described, when combined even with a rather simple diffusion step, does appear to provide a very secure image encryption mechanism.

References

1. R. Deepak and R. Twiggs, "Thinking out of the box: Space science beyond the CubeSat," *Journal of Small Satellites*, 1(1), pages 3–7, 2012.
2. J. Straub. "An open prototype for educational NanoSats: Increasing national space engineering productivity via a low-cost platform," 2nd National Academy of Inventors Conference. Tampa, FL. February, 2013.
3. Code of Federal Regulations, Title 47 – Telecommunication Volume: 5 Date: 2014-10-01 Original Date: 2014-10-01Title: Section 97.113 - Prohibited transmissions. https://www.gpo.gov/fdsys/pkg/CFR-2014-title47-vol5/xml/CFR-2014-title47-vol5-sec97-113.xml
4. Skrobot, G. and Coelho, R. "ELaNa–Educational Launch of Nanosatellite: Providing Routine Ride Share Opportunities," 26th Annual AIAA/USU Conference on Small Satellites, Logan, UT, United States. 2012.

5. European Space Agency, "Call for Proposals: Fly Your Satellite!," http://www.esa.int/Education/Call_for_Proposals_Fly_Your_Satellite
6. N. Singh and A. Sinha, "Optical image encryption using Harley transform and logistic map," *Optics Communications*, Vol. 282, no. 6, pages 1104-1109, 2009
7. G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solutions and Fractals,* Vol. 21, Issue 3, pages 749-761, 2004.
8. S-J Xu, X-B Chen, R Zhang, Y-X Yang, and Y-C Guo, "An improved chaotic cryptosystem based on circular bit shift and XOR operations," *Physics Letters A,* Vol. 376, Issues 10–11, pages 1003–1010. 2012.
9. G. Chen and T. Ueta, "Yet another chaotic attractor," *International Journal of Bifurcation and Chaos*, Volume 9, Issue 7, pages 1465-1466, 1999.
10. L. Kocarev, "Chaos-based cryptography: A brief overview," *IEEE Circuits and Systems Magazine*, Vol. 1, Issue 3, pages 6-21, 2001.
11. X. Huang, G. Ye, and K. Wong, "Chaotic image encryption algorithm based on circulant operation," *Abstract and Applied Analysis*, Volume 2013, Article ID 384067, 8 pages Hindawi Publishing Corporation, 2013.
12. S. Jackson, S. Kerlin, and J. Straub, "Implementing and Testing a Novel Chaotic Cryptosystem for Use in Small Satellites," 22nd ACM Conference on Computer and Communications Security (CCS'15), Denver, CO. 2015.
13. H. Gao, Y. Zhang, S. Liang, and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solutions and Fractals*, Vol. 29, No. 2, pages 393-399, 2006.
14. N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image and Vision Computing*, Vol. 24, Issue 9, pages 926–934, 2006.
15. S. C. Koduru and V. Chandrasekaran, "Integrated Confusion-Diffusion Mechanisms for Chaos Based Image Encryption," IEEE 8th International Conference on Computer and Information Technology Workshops, Sydney, QLD, pages 260-263, 2008.
16. H. Gao, Y. Zhang, S. Liang, and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solitons & Fractals*, Vol. 29, Issue 2, pages 393–399, 2006.
17. Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI Randomness Tests for Image Encryption," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications* (JSAT), April Edition, 2011.
18. W. Li and Y. Yuan, "A leak and its remedy in JPEG image encryption," *International Journal of Computer Mathematics*," Vol. 84, No. 9, pages 1367-1378, 2007.