Hardware Acceleration of Novel Chaos-based Image Encryption for IoT Applications

Andrew Boutros*†, Salma Hesham*‡, Barbara Georgey* and Mohamed A. Abd El Ghany*§
*Electronics Department, German University in Cairo, Cairo, Egypt
†Electrical and Computer Engineering Department, University of Toronto, Ontario, Canada
‡Application-Specific Multi-Core Architectures (MCA) Group, Ruhr-University Bochum, Germany
§Integrated Electronic Systems Lab, TU Darmstadt, Germany
E-mails: andrew.boutros@mail.utoronto.ca, {salma.hesham, mohamed.abdel-ghany}@guc.edu.eg

***Abstract*—In this paper, we propose a new chaos-based image encryption algorithm that combines Arnold's Cat and cascaded discrete Duffing equations maps for the confusion and diffusion stages of an image cryptosystem. The algorithm performs only one Arnold's Cat shuffle on the encryption side using two different keys instead of several shuffles used in conventional implementations. The security analysis of the proposed algorithm proves robustness when compared to state-of-the-art chaos-based encryption algorithms with less runtime and simpler operations. A complete accelerated hardware design of the proposed algorithm is implemented on Xilinx Zynq XC7Z020 FPGA board. For a 512 × 512 image, the hardware design achieves a maximum frequency of 135 MHz encrypting 256 fps which meets the real-time requirements of IoT applications.**

***Keywords*—Chaos, Image Encryption, FPGA, IoT**

I. Introduction

The wide invasion of internet applications and wireless communications, with increased number of connected devices over the Internet of Things (IoT), puts security of electronic data on top of the crucial demands for transmission reliability[1]. Specifically, the spread of IoT-based monitoring, surveillance and different imaging applications brings emphasized concerns for image cryptography. Chaos theory has emerged as a robust evolving path in this field [2], for its attractive properties such as periodicity, ergodicity, stochastic nature, and system unpredictable pseudo randomness with high sensitivity to small variations of initial conditions[3].

In that context, research efforts were brought towards several algorithmic proposals for chaotic image cryptography. They can be classified into three types: (1) coordinates transformation schemes, (2) values transformation schemes and (3) a mix of both types [3]. Coordinates transformations are called chaotic confusion, where pixels are scrambled by the use of permutation processes. Values transformations are called chaotic diffusion, where pixels are encrypted by changing their grayscale or RGB values. The applied strategies for the confusion and diffusion processes conform the core of any chaotic image encryption scheme. The work presented in this paper falls under the third category of mixed types. State-of-the-art strategies for mixed chaotic algorithms include Arnold's Cat map [4], [5], [6], genetic recombination [7] and colpitts chaotics [8] for the confusion stage, Chen's chaotic systems [4], [5], [9], Lorenz chaotic systems [10], hyperchaotic systems [7], Bernoulli shift map [6] and Duffing chaotic systems [8] for the diffusion stage as well as Clifford [11] and trigonometric-based chaos [12] for both stages. These algorithmic solutions compete on the level of security and robustness. Alongside, research efforts are exerted on the hardware acceleration of these compute-intensive algorithms to achieve high performance in terms of encrypted frames per second as well as efficient area utilization and power consumption [10], [11], [12], [13]. This is highly motivated by the increasing demands of real-time IoT imaging applications [1].

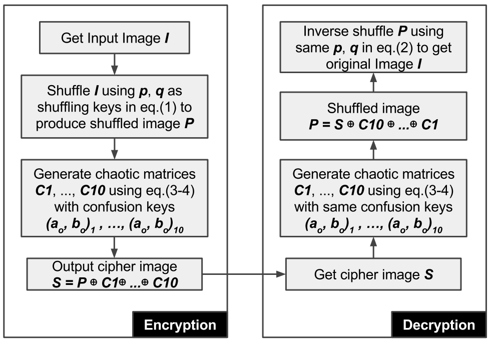

Get Input Image *I*
Shuffle *I* using *p*, *q* as shuffling keys in eq.(1) to produce shuffled image *P*
Generate chaotic matrices *C1*, ..., *C10* using eq.(3-4) with confusion keys $(a_o, b_o)_1, \ldots, (a_o, b_o)_{10}$
Output cipher image $S = P \oplus C1 \oplus \ldots \oplus C10$
Encryption
Inverse shuffle *P* using same *p*, *q* in eq.(2) to get original Image *I*
Shuffled image $P = S \oplus C10 \oplus \ldots \oplus C1$
Generate chaotic matrices *C1*, ..., *C10* using eq.(3-4) with same confusion keys $(a_o, b_o)_1, \ldots, (a_o, b_o)_{10}$
Get cipher image *S*
Decryption

Fig. 1. Proposed encryption and decryption scheme steps.

In this paper, we propose a new chaos-based image encryption scheme as shown in Fig. 1 and we implement a complete hardware accelerator for the proposed algorithm on an FPGA platform. Our contributions can be summarized as follows:

- We modify previous chaos image encryption algorithms to perform one Arnold's Cat shuffle using two different keys as well as multiple cascaded diffusion stages based on Discrete Duffing equations, each using different initial keys and simple XOR operations, instead of one stage of complex matrix manipulations in prior works.
- We present a full security analysis to prove that the proposed algorithm provides a significant runtime improvement without affecting the security robustness as the main concern.
- We implement a complete hardware accelerator for the proposed algorithm on a Xilinx Zynq FPGA. To the best of our knowledge, this is the first complete hardware implementation for a chaotic image cryptosystem that can achieve a throughput of 256 fps when tested for 512×512 images with an operating frequency up to 135 MHz.

The paper is organized as follows: Sec. II describes the proposed chaos-based algorithm which is then evaluated through a complete security analysis in Sec. III. Sec. IV presents the hardware architecture of the FPGA-based accelerator followed by the synthesis and timing results in Sec. V. Finally, Sec. VI concludes the work.

II. Proposed Encryption Scheme

The algorithm presented in this paper combines two chaotic systems, Arnold's Cat map [14] and Duffing equations [15], for the two stages of chaos-based image encryption. For the confusion stage, pixels are shuffled using a modified two-dimensional Arnold's Cat map. While, for the diffusion stage, pixels' values are manipulated through ten successive rounds of bit-wise XOR with chaotic maps generated using two-dimensional discrete Duffing equations. The mathematical foundations of the two algorithms were originally presented to study the mechanics of chaotic systems and were recently applied in encryption algorithms.

978-1-5386-4049-4/17/$31.00 ©2017 IEEE

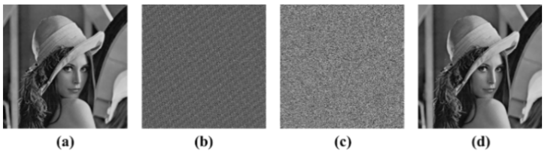


Fig. 2. Output images of each step of the algorithm (a) Original 512x512 grayscale image. (b) Shuffled image. (c) Cipher image. (d) Decrypted image.

A. Modified Arnold's Cat Confusion Scheme

Arnold's Cat map was originally presented to model the state space of a mechanical system as the surface of a torus with two variables x and y representing the longitude and latitude [14]. Starting from any point on a torus surface and moving with a fixed step in a helical shape, the system returns back to its starting state after N_T steps. Prior work in [16] used this property in image encryption and decryption. An $n \times n$ square image is shuffled N times using (1) on the encryption side such that x' and y' are the new pixel co-ordinates, x and y are its original co-ordinates, and p and q are non-zero positive integer values. On the decryption side, the image is shuffled again for $N_T - N$ times using the same p, q and N as shared keys to restore the original image.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} mod\ n \tag{1}$$

This work exploits a different property of the Arnold's Cat map: its invertibility. From (1), we derive an inverse Arnold's Cat map shown in (2). Then we use both maps to shuffle the input image for only one time on the encryption side and then inverse shuffle the image again on the decryption side using the same values of p and q as keys for the confusion stage. This modification significantly reduces the shuffling time from $N_T \times T_{shuffle}$ to only $2 \times T_{shuffle}$. We present an extensive security analysis in section III to prove that this modification did not affect the security robustness of the algorithm.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} mod\ n \tag{2}$$

B. Cascaded Discrete Duffing Maps Diffusion Scheme

Duffing map is a 2-D chaotic system, described by (3) and (4), that was also introduced to study chaotic dynamics [15]. It exhibits chaotic behavior where c and d are two constants with values 2.75 and 0.2 respectively, and initial values $a_0, b_0 \in [0, 1]$.

$$a_{i+1} = b_i \qquad 0 \le i < n \times n \tag{3}$$

$$b_{i+1} = -d\, a_i + c\, b_i - b_i^3 \qquad 0 \le i < n \times n \tag{4}$$

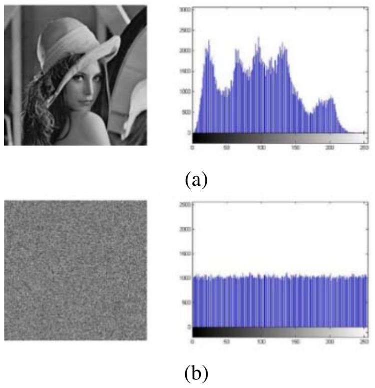
In the proposed algorithm, we use a cascade of ten confusion stages using ten different initial key sets $(a_0, b_0)_1, (a_0, b_0)_2, ..., (a_0, b_0)_{10}$. The Duffing equations are used on the encryption side to generate, from each initial key set, a sequence of numbers which forms a diffusive square matrix C of size $n \times n$. The entries in the matrix are the computed values of b_{i+1} from (4). The shuffled image is XORed in a cascaded manner with the ten generated chaotic matrices $C_1, C_2, ..., C_{10}$. On the decryption side, the same ten initial key values are used but in reverse order such that when the same matrices are XORed with the pixels' values of the cipher image, the shuffled image is restored. Security analysis is performed to determine the number of cascaded confusion stages required for maintaining the algorithm's security robustness.

III. Security Analysis

The proposed algorithm is implemented on Matlab R2016a to analyze its performance based on the standard security parameters. The analysis is performed on three grayscale test images of different sizes: Lenna (512×512), Lenna (256×256) and Cameraman (256×256). Fig. 2 shows the output images at each step of the implemented algorithm for the Lenna (512×512) test image.

A. Histogram Analysis

Histogram analysis is performed on the encrypted test images to visualize the difference in pixels' values distribution between the encrypted and the original image. Fig. 3 shows the results for the Lenna (512×512) test image. It shows a uniform distribution which differs significantly from that of the original image implying significant data distortion.



(a)

(b)

Fig. 3. Histogram analysis for (a) Original image. (b) Cipher image.

B. Correlation Coefficients

The correlation coefficients refer to the statistical relation between two adjacent pixels of an image, horizontally, vertically and diagonally. Original images are typically highly correlated (coefficients ≈ 1), while cipher images' pixels aim for dispersion (coefficients ≈ 0). Fig. 4 shows the correlation graphs for the same test image. For the original image, the correlation graph gives a linear shape unlike the scattered graph indicating poor pixels correlation in the cipher image.

C. Key Space Analysis

The proposed algorithm uses 22 different parameters as encryption keys which are: p and q in the confusion stage and the ten initial key sets $(a_0, b_0)_1, (a_0, b_0)_2, ..., (a_0, b_0)_{10}$ in the diffusion stage each of which is a 16-bit value. This results in an encryption key space of size 2^{352} proving robust resistance to brute-force attacks. Furthermore, decryption sensitivity is evaluated resulting in an output image that has 99.62% different pixels' values, in response to a single bit change in the cipher key.

D. Evaluated Security Parameters

The proposed algorithm is further evaluated in terms of its entropy value, mean square error, peak signal to noise ratio, percentage of changed pixels' values, and intensity of this change in the cipher image with respect to the original image as follows:

Entropy (E) indicates the randomness and unpredictability of the cipher image S and is calculated using (5), where $\Phi(x_i)$ is the probability of existence of a pixel of value x_i in S. For grayscale images, with maximum pixel value $P_{max} = 255$, the optimum value of Entropy is $E_{opt} = log_2(P_{max} + 1) = 8$.

$$E = \sum_{i=0}^{255} \Phi(x_i) \times log_2(\frac{1}{\Phi(x_i)}) \tag{5}$$

Mean Square Error (MSE) is the accumulative square error in pixel values between the cipher image S and the original image I as given in (6).

$$MSE = \frac{1}{n \times n} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [S(i,j) - I(i,j)]^2 \tag{6}$$

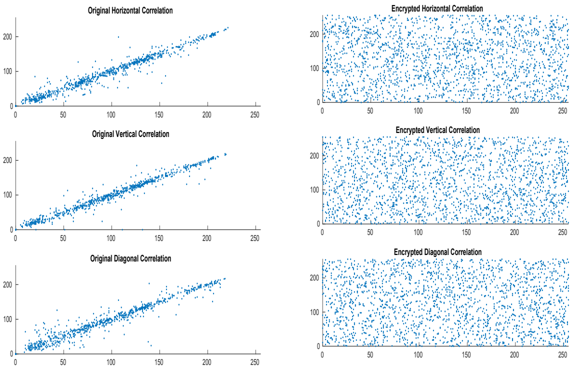
Peak Signal-to-Noise Ration (PSNR) is used to indicate the ability of the used encryption scheme to add noise and distortion to the original image as given in (7).

$$PSNR = 20 \times log_{10}(\frac{255}{MSE}) \tag{7}$$

Number of Pixel Change Rate (NPCR) is the percentage of pixels with changed values in cipher image S compared to the original image I. This parameter is determined using (8) and (9).

$$NPCR = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{D(i,j)}{n \times n} \tag{8}$$

$$D(i,j) = \begin{cases} 1, & S(i,j) \neq I(i,j) \\ 0, & S(i,j) = I(i,j) \end{cases} \tag{9}$$



(a) (b)

Fig. 4. Correlation graphs of random 2000 adjacent pixels in (a) Original Lenna image. (b) Cipher Lenna image.

Unified Average Changing Intensity (UACI) is the average change in all pixels' values of the cipher image S compared to the original image I as given in (10). It is used to indicate the sensitivity of the encryption scheme to changes in encryption keys when comparing the same image encrypted using different keys.

$$UACI = \frac{1}{n \times n} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \frac{|S(i,j) - I(i,j)|}{255} \tag{10}$$

The security analysis results for the proposed algorithm for the three test images are presented in Table I. In addition, Table II presents the correlation values of cipher images calculated for horizontal, vertical and diagonal directions. Both tables compare the obtained results to similar works proposing different chaos-based image encryption algorithms. For all test images, the proposed algorithm provides the highest entropy value. Specifically compared to [16] which uses the conventional Arnold's Cat scheme, our proposed algorithm is highly promoted by the obtained results. The results show a maintained security robustness in conjunction with reduced encryption time and simplified hardware implementation.

TABLE I
SECURITY PARAMETERS FOR DIFFERENT TEST IMAGES

Image	Lenna 512x512				
Ref.	[16]	[17]	[8]	Ours	
MSE	88.1073	N/A	N/A	88.7695	
PSNR	28.6807	N/A	N/A	28.6482	
NPCR	99.6094	99.6036	99.57	99.6074	
UACI	30.6605	33.0615	35.08	30.6068	
Entropy	7.997	7.997	7.996	7.999	
Image	Cameraman 256x256		Lenna 256x256		
Ref.	[18]	Ours	[7]	[9]	Ours
MSE	N/A	108.2	N/A	N/A	88.44
PSNR	N/A	27.78	N/A	N/A	28.66
NPCR	99.53	99.64	99.6	99.6	99.6
UACI	26.88	31.02	33.57	33.43	30.48
Entropy	7.571	7.997	7.997	7.997	7.997

TABLE II
CORRELATION COEFFICIENTS FOR LENNA CIPHER IMAGES

Image	Lenna 512×512 ($\times 10^{-3}$)			Lenna 256×256 ($\times 10^{-3}$)		
Ref.	[8]	[17]	Ours	[7]	[9]	Ours
Horiz.	1.4	72.2	0.8	5.5	2.7	7.2
Vert.	2.3	9.9	3.1	6.4	15.2	4.5
Diag.	0.2	20.1	1.4	7.2	7.1	3.3

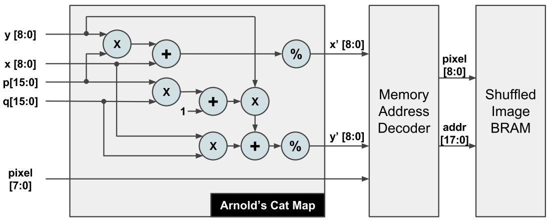


Fig. 5. Hardware architecture of the confusion stage.

IV. FPGA-BASED HARDWARE ACCELERATION

The second contribution of this paper is the acceleration of the proposed encryption scheme in a complete hardware solution suitable for real-time IoT imaging applications. This section gives a detailed description of the designed hardware architecture for each of the confusion and diffusion stages of the proposed algorithm.

A. Hardware Architecture of the Confusion Stage

The hardware architecture of the confusion stage based on the modified Arnold's Cat map is shown in Fig. 5. It takes as input the pixel value (8-bits) and its x and y coordinates ($log_2(n)$-bits, where n is the dimension of the input image), performs the location permutation and outputs the new pixel position. The first block implements the Arnold's transformation using the 16-bit keys p and q to compute the new pixel coordinates following (11) and (12) derived from (1). The subsequent blocks are responsible for saving the shuffled pixels by decoding the new coordinates into a memory address to store the pixel value into the appropriate location in an on-chip Block Random Access Memory (BRAM).

$$x' = [x + py] \ mod \ n \tag{11}$$

$$y' = [qx + (pq + 1)y] \ mod \ n \tag{12}$$

B. Hardware Architecture for the Diffusion Stage

The hardware architecture of the diffusion stage is illustrated in Fig. 6. Shuffled pixels stored in the BRAM are pushed sequentially into a pipeline of ten successive diffusion stages. At each stage, the input pixel is XORed with the outputs of the discrete Duffing equations blocks generated on-the-fly with different initial keys. The Duffing equations block is implemented using fixed point arithmetic such that its output is truncated into 8-bit Q6 format (i.e. 2 integer and 6 fractional bits) that is XORed with the 8-bit pixel. Fixed point precision was used to reduce the soft logic utilization especially that using high precision is not needed in this case. We also verified that the encrypted image using fixed point precision in hardware is identical to that produced by the floating point software implementation to assure that using reduced precision did not affect the security robustness of the algorithm.

V. SYNTHESIS AND SIMULATION RESULTS

A complete synthesizable hardware architecture is designed for the proposed algorithm using VHDL on a Xilinx Zynq XC7Z020 FPGA board. The designed architecture is used for the encryption of grayscale images. It is synthesized, placed and routed using Xilinx ISE 14.5.

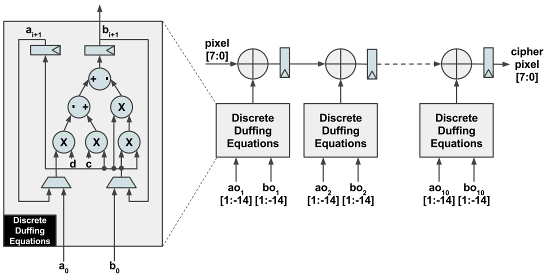


Fig. 6. Hardware architecture of the diffusion stage.

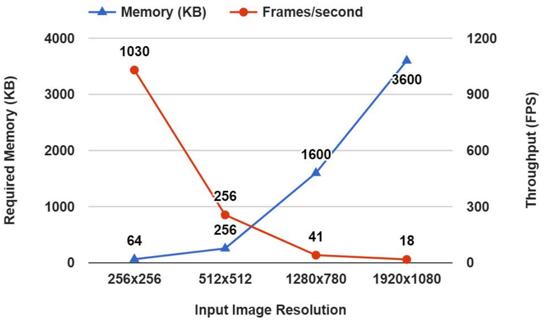


Fig. 7. Memory usage and encryption throughput for different resolutions.

A. Timing Results and Hardware Resources Utilization

The hardware runs at 135 MHz—encrypting one 512×512 frame in 3.9 ms with a throughput of 256 fps. A comparison between the software Matlab implementation runtime, on an Intel Core i7-4790 3.6 GHz CPU with 16 GB of RAM running Debian Linux release 8.8 in both floating and fixed point versions, and that of the hardware architecture is presented in Table III. It shows an acceleration of 7x in the confusion stage, 45x in the diffusion stage and an overall acceleration of about 26x compared to the fixed point software implementation. The high speed-up achieved in the diffusion stage comes as a result of hardware pipelining in contrast to the software sequential execution for generating the Duffing map and performing the successive XORing stages.

Table IV summarizes the resources utilization for the implemented design. The implemented design uses less than 1% of the available registers, 2% of the LUT resources and 46% of the BRAM capacity of the FPGA needed to store 512×512 pixels of the shuffled image, each of which is 8-bit wide, i.e. a total of 256KB. The design's resource utilization is limited by BRAM, similiar to many other high-throughput highly optimized FPGA-based designs [19]. Fig. 7 shows how memory utilization and number of encrypted frames per second scale with input image size. The proposed hardware design can offer a throughput up to 18 fps for 1080p resolution.

B. Comparison Versus State-of-the-Art Hardware Designs

The developed design is synthesized on similar board configurations at an attempt for fair comparison with state-of-the-art hardware solutions. Ansarmohammedi et al. [12] presented several HW/SW co-design configurations. In Table V, our work is compared to their fastest design. Our design achieves a 7.7x speedup with around half the resource utilization taking into consideration the 1.3 factor of different frame sizes. Compared to the Azzaz et al. [10], our design shows a slightly faster performance with less FPGA resources, however their implementation is a diffusion-only scheme. To the best of our knowledge, this work is the first chaotic image cryptosystem with throughput of 256 fps for 512×512 images and 18 fps for 1080p resolutions.

VI. Conclusion

This paper presented a new chaos-based image encryption algorithm combining a modified Arnold's Cat map scheme together with cascaded Duffing equations. The proposed algorithm maintains the required level of security robustness when analyzed and compared to state-of-the-art schemes. The algorithm is accelerated on a Xilinx Zynq FPGA board and the proposed hardware architecture shows up to 2x reduction in resource utilization and 7.7x speedup in encryption time compared to state-of-the-art designs. The implemented hardware runs at 135 MHz offering a throughput of 256 fps for 512×512 images and 18 fps for 1080p images meeting the real-time requirements of IoT applications.

TABLE III
Encryption Time in Software vs. Hardware Design

	SW (float)	SW (fixed)	HW Accelerator	Speedup
Confusion (ms)	12.7	12.7	1.9	7x
Diffusion (ms)	147.4	89.1	2	45x
Total (ms)	160.1	101.8	3.9	26x
Throughput (fps)	6.25	9.82	256	26x

TABLE IV
FPGA Resources Utilization for the Hardware Design

FPGA Resources	Used	Available	Utilization
Registers	827	106400	1%
LUTs	1001	53200	2%
BRAMs	64	140	46%
DSPs	60	220	27%

TABLE V
Comparison to Other FPGA-based Accelerators

Ref.	Logic elements	Memory	Enc. Time (ms)	FPS	Frame Size (B)
Altera Cyclone IV E FPGA					
[12]	2477	330 KB	23	43	195,075
Ours	1223	256 KB	3.9	256	262,144
Ratio	0.5	0.6	7.7	5.9	1.3
Virtex II Pro XC2VP30FF896 -7					
[10]	1375	288 KB	5.7	174	262,144
Ours	1119	288 KB	5.6	178	262,144
Ratio	0.81	1	0.98	0.98	1

References

[1] E. Kougianos *et al.*, "Design of a High-Performance System for Secure Image Communication in the Internet of Things," *IEEE Access*, vol. 4, pp. 1222–1242, 2016.

[2] L. Kocarev *et al.*, "From Chaotic Maps to Encryption Schemes," in *IEEE ISCAS*, 1998, pp. 514–517.

[3] J.-I. Guo *et al.*, "A New Chaotic Key-based Design for Image Encryption and Decryption," in *IEEE ISCAS*, 2000, pp. 49–52.

[4] Z.-H. Guan *et al.*, "Chaos-based Image Encryption Algorithm," *Physics Letters A*, vol. 346, no. 1, pp. 153–157, 2005.

[5] D. Xiao *et al.*, "Analysis and Improvement of a Chaos-based Image Encryption Algorithm," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2191–2199, 2009.

[6] R. Ye, "A Novel Chaos-based Image Encryption Scheme with an Efficient Permutation-Diffusion Mechanism," *Optics Communications*, vol. 284, no. 22, pp. 5290–5298, 2011.

[7] X. Wang *et al.*, "A Novel Image Encryption Algorithm Based on Genetic Recombination and Hyper-chaotic Systems," *Journal of Nonlinear Dynamics*, vol. 83, no. 1-2, pp. 333–346, 2016.

[8] Y. Abanda *et al.*, "Image Encryption by Chaos Mixing," *IET Image Processing*, vol. 10, no. 10, pp. 742–750, 2016.

[9] L. Kong *et al.*, "A New Image Encryption Algorithm Based on Chaos," in *35th IEEE Chinese Control Conference (CCC)*, 2016, pp. 4932–4937.

[10] M. Azzaz *et al.*, "Real-time Image Encryption Based on Chaotic Synchronized Embedded Cryptosystems," in *NEWCAS*, 2010, pp. 61–64.

[11] J. Giesl *et al.*, "Hardware Solution of Chaos-based Image Encryption," in *IEEE DDECS*, 2009, pp. 198–201.

[12] S. Ansarmohammadi *et al.*, "Fast and Area-efficient Implementation for Chaotic Image Encryption Algorithms," in *IEEE CADS*, 2015, pp. 1–4.

[13] Q. Wang *et al.*, "Theoretical Design and FPGA-based Implementation of Higher-dimensional Digital Chaotic Systems," *IEEE Trans. on Circuits and Systems*, vol. 63, no. 3, pp. 401–412, 2016.

[14] V. I. Arnold *et al.*, "Ergodic Problems of Classical Mechanics," 1968.

[15] G. Duffing, *Forced Oscillations with Variable Natural Frequency and their Technical Relevance*, 1918, no. 41-42.

[16] S. Naveenkumar *et al.*, "Chaos and Hill Cipher Based Image Encryption for Mammography Images," in *IEEE ICIIECS*, 2015, pp. 1–5.

[17] S. S. Alam *et al.*, "A Novel Image Encryption Algorithm using Hyper-chaos Key Sequences, Multi-step Group-based Binary Gray Conversion and Circular Bit Shifting Logic," in *IEEE ICSEMR*, 2014, pp. 1–9.

[18] M. G. Avasare *et al.*, "Image Encryption using Chaos Theory," in *IEEE ICCICT*, 2015, pp. 1–6.

[19] S. Yazdanshenas *et al.*, "Don't Forget the Memory: Automatic Block RAM Modelling, Optimization, and Architecture Exploration," in *ACM FPGA*, 2017, pp. 115–124.