Received March 18, 2019, accepted April 6, 2019, date of publication April 12, 2019, date of current version April 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2910859

Implementation of Encryption Algorithm and Wireless Image Transmission System on FPGA

CHENG-HSIUNG YANG[1], HOU-CHENG WU[1], AND SHUN-FENG SU[2]
[1]Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan

Corresponding author: Cheng-Hsiung Yang (yangch@mail.ntust.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan (ROC), under Grant MOST 107-2221-E-011-091-.

ABSTRACT In this paper, we proposed to improve bit insertion based on the common chaotic encryption algorithm, which reduces the computational complexity of the chaotic encryption algorithm and is used on low computational systems. We changed the encryption steps from permutation to XOR and XOR to bit insertion, which is our proposed bitwise operation based on encryption. Next, we proposed a four-dimensional chaotic system using the discrete time signal of the chaotic system for the key generator and optimized the floating point operation as well as FPGA resources of the chaotic system. Then, we implemented the algorithm on our Nios II-based wireless image transmission system, where the system's processor clock is 50 MHz. Finally, we performed a security analysis on our encryption algorithm, and the results show that our proposed algorithm consumes less computing resources while maintaining sufficient security.

INDEX TERMS Chaotic system, XOR, bits insertion, image encryption, FPGA.

I. INTRODUCTION

We have to believe the chaotic characteristics are important and desire to apply chaotic encryption algorithms to mobile phones, smart home appliances and other devices. However, after studying the common encryption algorithm [1]–[6], we find that the algorithm needs to be upgraded in terms of security and computational requirements. In addition, some authors of chaotic encryption algorithms claim that chaotic encryption algorithms have faster speeds, and even claim that their algorithms are for low-performance systems. However, the proposed algorithm uses a Core 2 Duo processor to operate in a Matlab environment [44], or [45] uses a 50×50 grayscale image test result and shows an 128×128 encryption time. These are not real low-performance systems or commonly used file types.

Therefore, in order to solve the controversy of non-use of low-performance systems, we will build a 50 MHz FPGA-based Nios II system and implement our algorithm on this system. And we design our chaotic encryption algorithm based on the common file types, simplified operations, and enhanced differential attack resistance [22]–[29].

For the diversity of encryption algorithms, we consider the chaotic characteristics and hope to apply chaotic encryption algorithms to mobile phones, smart home appliances and other devices. However, after studying the traditional encryption algorithm, we find that the algorithm needs to be improved in terms of security and computational requirements.

The traditional chaotic encryption algorithms usually consist of two steps. The first step is to perform complex and time-consuming element permutation operation on the plaintext. The second step is to use the iterative value of the chaotic map as a fixed key to perform an XOR operation with the ciphertext generated in the first part. Obviously, this type of algorithm has security issues in the ciphertext generated in the first step, so the second step of the operation is needed to encrypt the ciphertext again. In addition, this type of algorithm is usually only applicable to the specific format of data types.

For this reason, the encryption algorithm proposed in this thesis is to improve these two steps and reverse the order of steps of the traditional chaotic encryptions. First, we designed

The associate editor coordinating the review of this manuscript and approving it for publication was Rasheed Hussain.

a four-dimensional chaotic system and a key generator that adaptively generates a key based on the iterative values of the chaotic system and the eigenvalues of the plaintext. The key generator is able to improve the security issues of the second step of the traditional chaotic encryption algorithms and to increase the amount of the text required for the differential attacks. Then, we used the Bits Insertion operation based on the logical operation and the bit operation to replace the element permutation operation in the first step of the traditional chaotic encryption algorithms, which greatly reduced the time consumed by the encryption operation, and consequently to improve the security [30]–[43].

This paper is organized as follows. Section II presents our designed a four-dimensional chaotic system, the difference of the same chaotic system in different numerical operations, our designed key generator, Bits Insertion scheme, and the flow of proposed chaotic encryption algorithm. Section III explains the resource optimization of the chaotic system circuit and how we implement the wireless image transmission encryption system on the FPGA. In Section IV, we analyze the security of proposed encryption algorithm. Finally, summarize the advantages of proposed algorithm in Section V.

II. ALGORITHM DESIGN

In this section, we present the effects of numerical operations on chaotic systems and apply these effects to our key generators. Then explain our proposed Bits Insertion encryption. The encryption and the decryption process of proposed algorithm shown through two flow charts are described as follows:

A. EFFECTS OF NUMERICAL OPERATIONS ON THE CHAOTIC SYSTEM

We designed a chaotic system based on the Lorenz attractor as Equation (1),

$$\begin{cases} \dot{x} = a\,(y - x) + yz + w \\ \dot{y} = bx + cy - xz - ew \\ \dot{z} = x^2 - dz \\ \dot{w} = x - w \end{cases} \tag{1}$$

where parameter values are $a = 20$, $b = 0.5$, $c = 6.8$, $d = 8$, and $e = 0.5$.

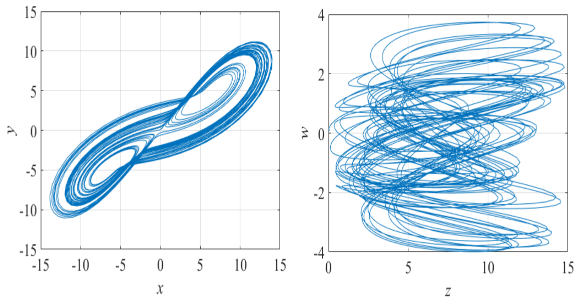

FIGURE 1. Two-dimensional phase portraits of proposed chaotic system.

We use ODE45 algorithm in MATLAB, set the initial values $x, y, z, w = [0.1, 0.1, 0.1, 0.1]$, time interval $t = [250, 300]$, and draw two-dimensional phase portraits as Fig. 1. It can be clearly found that this system has a butterfly-like orbit with Lorenz's system in the three dimensions x, y, z.

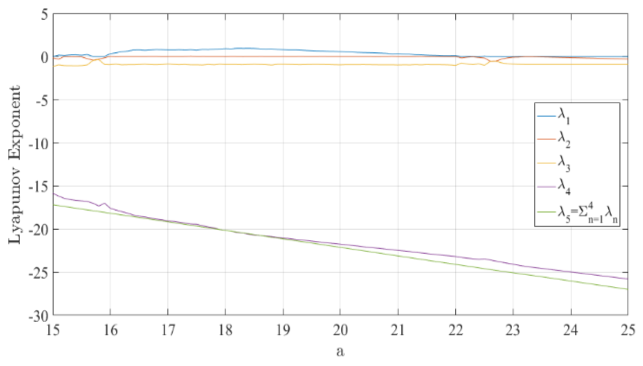

FIGURE 2. Lyapunov exponents of proposed chaotic system.

We continually analyze the Lyapunov exponent of the system. We refer to the method proposed by Wolf *et al.* [7] and extend the MATLAB program proposed by Vasiliy Govorukhin to calculate the Lyapunov exponent [8] to four dimensions. The time parameter t cannot be set to the infinity because it is limited by the computing system, although we have increased Vasiliy Govorukhin's original time parameter from 200 seconds to 300 seconds. We verified all the parameters in the proposed chaotic system, and the results show that the proposed system is chaotic. The result of a parameter is shown in Fig. 2.

In order to use the signal of the chaotic system for the key generator, we use the Euler method to obtain the discrete time signal of the chaotic system as Equation (2),

$$\begin{cases} x(n+1) = \{a\,[y(n) - x(n)] + y(n)z(n) + w(n)\}\,h + x(n) \\ y(n+1) = [bx(n) + cy(n) - x(n)z(n) - ew(n)]\,h + y(n) \\ z(n+1) = \left[x^2(n) - dz(n)\right] h + z(n) \\ w(n+1) = [x(n) - w(n)]\,h + w(n) \end{cases} \tag{2}$$

where step size $h = 0.01$.

However, we found that in the single-precision and the double-precision operations, there are more than 10^{-6} differences in less than 100 iterations, and the difference reaches more than one digit after 1000 iterations. This is because the binary has a round-off error, so the binary cannot strictly obey the commutative law or the distribution law in floating-point operations. One of the characteristics of the chaotic system is that it is sensitive to errors, which means that we only need to change the calculation order of Equation (2) to the form of Table 1, and we will get different results.

The Fig. 3 shows the difference in the x dimension calculated by the system using Equation (2) and Table 1. When the system is chaotic ($c = 6.8$), the difference is sharply enlarged after $t = 50$, and when the system is periodic ($c = 4.5$), the difference is always maintained below 10^{-12}.

This situation meant to us is to adopt the chaotic system to generate the encryption key, because even if the chaotic

TABLE 1. The calculation order of the chaotic system.

	$x(n+1)$	$y(n+1)$	$z(n+1)$	$w(n+1)$
(a)	$x_{11}=y(n)-x(n)$ $x_{12}=y(n)\times z(n)$ $x_{21}=a\times x_{11}$ $x_{22}=x_{12}+w(n)$ $x_3=x_{21}+x_{22}$	$y_{11}=b\times x(n)$ $y_{12}=c\times y(n)$ $y_{13}=x(n)\times z(n)$ $y_{14}=e\times w(n)$ $y_{21}=y_{11}+y_{12}$ $y_{22}=y_{13}+y_{14}$ $y_3=y_{21}-y_{22}$	$z_{21}=x(n)\times x(n)$ $z_{22}=d\times z(n)$ $z_3=z_{21}-z_{22}$	$w_3=x(n)-w(n)$
(b)	$x_4=x_3\times h$ $x(n+1)=x_4+x(n)$	$y_4=y_3\times h$ $y(n+1)=y_4+y(n)$	$z_4=z_3\times h$ $z(n+1)=z_4+z(n)$	$w_4=w_3\times h$ $w(n+1)=w_4+w(n)$

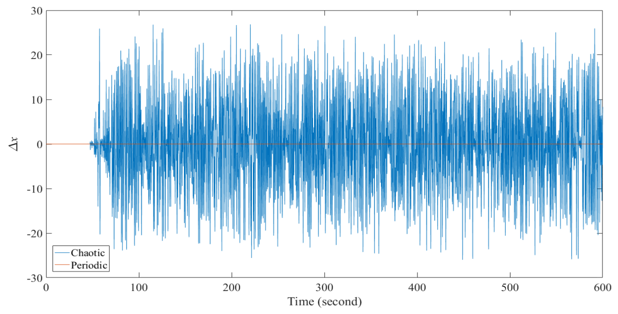

FIGURE 3. The difference in x dimension with Equation (2) and Table 1.

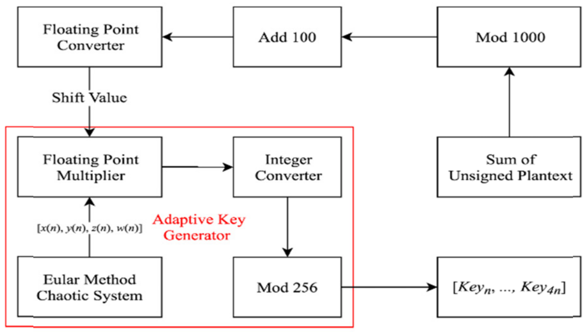

FIGURE 4. The chaotic key generator.

system adopted is public, as long as the precision and order of the calculation process are non-public, others will not get the same results.

B. OUR CHAOTIC KEY GENERATOR

Such above keys have some fatal flaws in the resistance of plaintext attacks. Even if the plaintext is repeatedly encrypted several times, the ciphertext will still have obvious features because of the file format feature of the plaintext and the fixed key, which is conducive to the implementation of the plaintext attack. Since the XOR algorithm of the fixed key cannot pass the NPCR test, most chaotic encryption algorithms will use pixel rearrangement for the secondary-encryption to improve it, but the paper in Ref. [9] shows that this method is still not safe.

Therefore, this paper proposes an adaptive key generator whose length and elements will change with different plaintext. The key generator is shown in Fig 4. It mainly consists of Euler method chaotic system and a floating-point multiplier. The generator can generate the key elements, which are equal to the dimensions of chaotic system in each step. When the key generated by the key generator used for the XOR operation, the effect of confusion and diffusion can be achieved at the same time.

The floating-point multiplier multiplies the iterative value of each chaotic system by a shift value calculated from the plaintext. Then, round the quotient to the integer by an integer converter. Finally, the remainder of the integer 256-modulo operation is the output as key elements. These processes are equivalent of treating the chaos system's iterative value as a vector, multiplying the vector by a multiple of the shift value, and project it onto the 256-plane. The larger the shift value range, the larger the key space.

The shift value is the sum of the elements of the plaintext as unsigned integers. Then we take the 1000-modulo operation and add 100, as in Equation (3),

$$\begin{aligned}&Shift\_Value\\&\quad=\left(\sum_{n=1}^{Plaintext\_Length} Unsigned\_Plaintext_n\right)\%1000+100\end{aligned}\tag{3}$$

where % is the modulo operator, and the result is stored in the single-precision floating-point format.

The 1000-modulo operation here is that because the single-precision floating-point number has only 7 significant digits of decimal digits, the iteration values of the chaotic system are up to ten digits. The floating-point multiplication operation may have 2 digits decimal error, so we conservatively limit the multiplier to about 10^3. In order to avoid excessive overflow operation, the randomness of the key will decrease. The addition of 100 intends to avoid a situation in which the key has a continuously 0 decreasing randomness when the shift value is 0 and the chaotic system is near the equilibrium point (0, 0, 0, 0).

TABLE 2. Average entropy of the consecutive keys.

Sample Size	256	512	768	1024	1280	1536	1792	2048	2304	2560
Average Entropy	7.124	7.566	7.710	7.783	7.824	7.852	7.879	7.890	7.902	7.913

Finally, we examine the minimum range of applicable XOR encryption for the keys generated by the adaptive key generator. That is, Entropy is calculated by randomly 256 times consecutive keys, until the average Entropy > 7.9 of the sample taken out, then the minimum effective key available for XOR encryption is obtained. We take the average of 1000 samples from each sampling interval and round down the results to the third decimal place as shown in Table 2. The result shows that the key generated by our designed key generator is suitable for XOR encryption of plaintext over 2.25KB.

C. BITS INSERTION

The common chaotic encryption algorithms are modeled on the concepts of AES and DES, and the encryption method can be classified into two steps of XOR operation and permutation, just like the two main methods of the encryption algorithm proposed by Shannon in Ref. [10]:

1) Diffusion: Making a subtle change in the plaintext produces a completely different ciphertext, such as permutation.
2) Confusion: Mask the relationship between plaintext and ciphertext, such as the XOR operation.

However, these chaotic encryption algorithms often use a large number of complex operation methods in the permutation.

TABLE 3. The key length and its expected security age.

Symmetric	ECC	RSA/DH/DSA	MIPS Years to Attack	Protection Lifetime
80	160	1024	10^{12}	Until 2010
112	224	2048	10^{24}	Until 2030
128	256	3072	10^{28}	Beyond 2031
192	384	7680	10^{47}	
256	512	15360	10^{66}	

Not only is it less efficient than AES and 3DES, but it also cannot increase security by increasing the length of the key due to the use of a fixed-length permutation key. Table 3 shows the key lengths and expected security ages of AES, RSA, ECC, and other algorithms [11].

In Section III.B, we propose some improvements to common XOR arithmetic steps for chaotic encryption algorithms. Using a key generator that is sensitive to the plaintext, the encryption of the XOR operation has the characteristics of confusion and diffusion at the same time. And in this section, we propose a re-encryption scheme with a variable-length key that also has obfuscation and diffusion characteristics. We call this the Bits Insertion encryption.

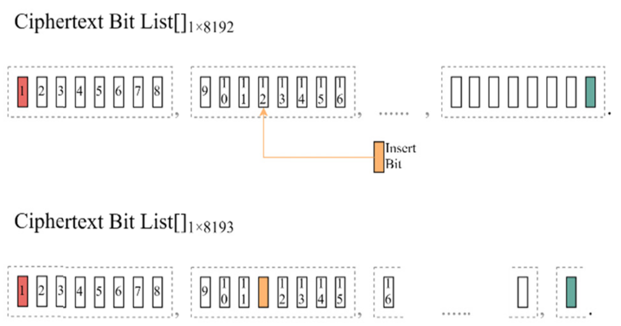
Ciphertext Bit List[]$_{1\times 8192}$
Insert Bit
Ciphertext Bit List[]$_{1\times 8193}$

FIGURE 5. Change of bits sequence by bits insertion at position 12.

In the Bits Insertion encryption, we treat the elements of the XOR encrypted ciphertext as an unsigned integer and take the value of the 1000-modulo operation as the ciphertext characteristic. In the Bits Insertion encryption, we treat the elements of the XOR encrypted ciphertext as an unsigned integer and take the value of the 1000-modulo operation as the ciphertext characteristic. And calculate the value *Num*, which between 1 and 10, shown as Equation (4),

$$\begin{aligned} Num = 1 + \frac{1}{2}[(&Ciphertext\_Length\ \% \\ &Ciphertext\_Characteristic)\ \%\ 10] \\ &+ \frac{1}{2}(Ciphertext\_Characteristic\ \%\ 10) \end{aligned} \tag{4}$$

where % is a modulo operator. This value multiplied by 8 is the number of bits to insert.

Next, we follow the steps 1 to 4 below to obtain the key used by the Bits Insertion encryption algorithm. The key is a sequence showing the position of the insertion bit in the ciphertext. However, the programming language efficient unit of operation is a byte, so in steps 5 to 7, we convert the key to a sequence of numbers representing the bytes' position, and a sequence of bit positions in the byte. This improves the efficiency of algorithmic operation:

Step 1. Change the original input adaptive key generator's shift value to the ciphertext characteristic, after the XOR encryption key is generated.

Step 2. Continue to operate the adaptive key generator $2 \times Num$ times to obtain a bit length of $8 \times Num$ elements as the relative position of the Bits Insertion sequence, abbreviated as *bIRP*[].

Step 3. Make ciphertext length in floating-point format divided by the sum of elements of *bIRP*[] to get the quotient in floating-point, abbreviated as *FpQ*.

Step 4. Have *bIRP*[] multiplied by *FpQ*, and then round down it to the integer to get the absolute position of the bit insertion sequence, abbreviated as *bIAP*[].

Step 5. Let *bIAP*[] divided by 8 in the integer format to get the integer quotient sequence, which is the position number of the byte used to insert bits in the ciphertext, abbreviated as PBI[].

Step 6. Take the remainder of *bIAP*[] in an integer format to get the number of Bits Insertion positions in the bytes in the ciphertext, abbreviated as *PbI*[].

Step 7. View the elements of *PbI*[]. If $PbI[i]$ is 0, change to 8 and subtract 1 from the corresponding $PBI[i]$ value.

Then we use these two sequences to encrypt the ciphertext with Bits Insertion. The Bits Insertion encryption is to consider the byte sequence in the ciphertext as a single string bit sequence. The key for inserting the bit encryption is the position in the order of 1 from the highest bit of the ciphertext in bits.

When the encryption of bit insertion, it counts backward from position 1 to the specified position and inserts a bit of value $1_{(b)}$. From the inserted position, the original bits are all shifted backward. The bit sequence changes when the specified position 12 is shown in Fig. 5. Each bit insertion thus causes confusion and diffusion of the ciphertext from the insertion position until every multiple of 8 times is inserted.

The ciphertext encrypted by the Bits Insertion will have more Num bytes than the original ciphertext. This can avoid doing differential attacks on the file header, thus speeding up the cracking of ciphertext. The number of inserted bits is a multiple of 8 because the data not reaching 8 bits will result in the failure to convert to a byte or add complement of $0_{(b)}$ to become a byte. These situations will result in the loss of data, the inability to decrypt, or the number of bits that are found to be inserted to reduce the difficulty of MIPS attack. Although the number of $1_{(b)}$ in the plain text is usually higher than $0_{(b)}$, we still insert $1_{(b)}$. This is because in most cases, $0_{(b)}$ will be slightly higher than $1_{(b)}$ in ciphertext encrypted by XOR operation. Therefore, we choose to insert $1_{(b)}$ to make the $0_{(b)}$ and $1_{(b)}$ numbers closer and have a higher Entropy, so we refer to the inserted $1_{(b)}$ as the noise bit.

In addition, selecting Num as in Equation (4) is due to the length of the ciphertext, the ciphertext eigenvalues will be changed after the Bits Insertion encryption, and such changes will be non-linear. At the same time, to avoid the encryption, in most cases, will consume more storage and transmission resources. We consider the following two points:

1) Since the unit size of the hard disk storage, called "cluster size", is set to 4 KB [12] in mainstream databases such as NTFS. Therefore, any size P bytes file in the hard disk with a cluster size of 4 KB actually occupies Q bytes, as in Equation (5).
2) During transmission, the file is cut into packet lengths and sent one by one. According to IEEE 802.3 specification [13], there are three types of packets: a basic frame with a length of 1500 bytes, a Q-tagged frame with a length of 1504 bytes, and an envelope frame with a length of 1982 bytes. Basic frame is the most common primary transmission packet.

After encrypting the ciphertext of R bytes with Bits Insertion, we obtain the other ciphertext of S bytes. If R and S conform to Equation (6), the ciphertext encrypted by bits will not require more storage and transmission resources. Therefore, we conservatively limit *Num* between 1 and 10 by taking 10-modulo operation. This makes Bits Insertion encryption not consume more storage or transmission resources in most cases.

$$Q = 4096 \times \left\lceil \frac{p}{4096} \right\rceil \tag{5}$$

$$\begin{cases} \left\lceil \frac{R}{4096} \right\rceil = \left\lceil \frac{S}{4096} \right\rceil \\ \left\lceil \frac{R}{1500} \right\rceil = \left\lceil \frac{S}{1500} \right\rceil \end{cases} \tag{6}$$

where $\lceil\ \rceil$ is ceiling function.

In the end, we must declare that the Bits Insertion encryption is not suitable for directly encrypting the plaintext. This is because the Bits Insertion encryption is used to add noise in the ciphertext, making differential attack difficultly. However, the plaintext formed by a fixed block is highly resistant to the confusion and proliferation caused by Bits Insertion encryption. After the Bits Insertion encryption, some blocks still have obvious features. The BMP file is a typical case.

D. FLOW OF ENCRYPTION ALGORITHM

The Fig. 6 shows the flow of the encryption algorithm, where Shift Value and Ciphertext1 Characteristic are the keys of XOR and Bits Insertion respectively.

III. FPGA IMPLEMENTAYION

In this section, we are proposing the resource optimization of the chaotic system circuit on the FPGA and explain the architecture of the Nios II-based wireless image transmission system we implemented. The hardware used in this paper is a pair of DE2-115 motherboards and Terasic RFS daughter cards. And the system shows as Fig 7.

A. RESOURCE OPTIMIZATION OF THE CHAOTIC SYSTEM CIRCUIT

We use the adder, substractor and multiplier in the IP core of Altera's floating-point arithmetic unit to implement the circuit. By setting these IP core to the fastest architecture, the latency of adder, substractor, and multiplier are 7 units, 7 units, and 5 units, respectively. The unit of the latency is the interval of two clock edge signals. Our key generator and its chaotic system circuit operate with a 50 MHz clock signal. With the setting of our condition, each of the adder or the substractor is constituted by 170 look-up tables (a.k.a. LUT) and 377 registers (a.k.a. REG). And each of the multiplier is constituted by embedded 9-bit elements (a.k.a. 7dsp_9bit), 111 LUTs, and 209 REGs.

It can be seen that these floating-point arithmetic units are very much resource intensive, so we propose a resource optimization method that is able to reuse~~s~~ arithmetic units.

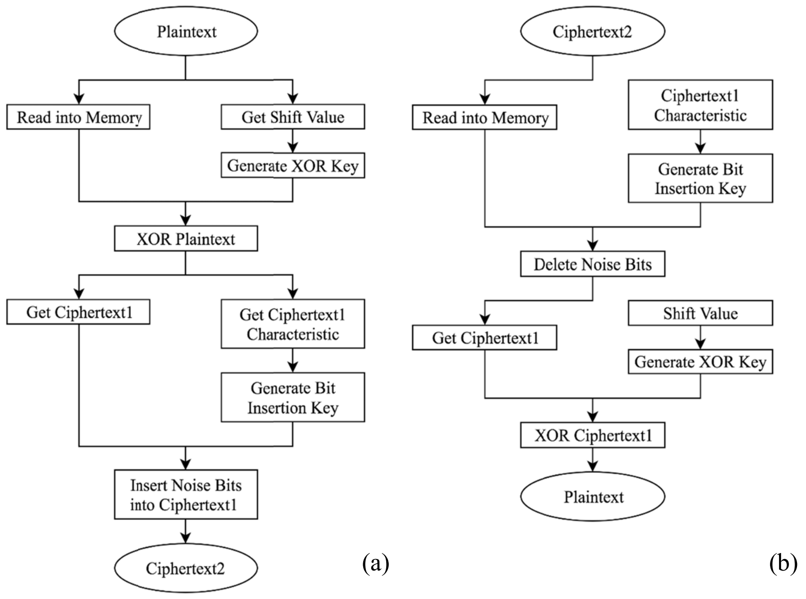

FIGURE 6. **Encryption and decryption flow charts. (a) is encryption flow and (b) is decryption flow.**

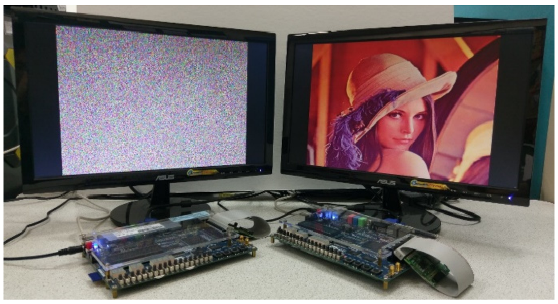

FIGURE 7. **Wireless image transmission system.**

We know that these three types of arithmetic units have the following three common characteristics:

1) There are two inputs and one output.
2) After any input signal changes, the output after the latency will change.
3) The sum of the latency of any two units must be greater than the latency of the third unit.

This means that we can reuse the arithmetic units from step of second front. Then we can compare the hardware resources used by the non-reuse with reusing unit architecture as shown in Table 4. The red numbers in the table indicate that the arithmetic unit is the reused one. It can be seen through the proposed method we made to save 1686 REGs, 2329 LUTs, and 49bits DSPs in this case.

B. ARCHITECTURE OF THE WIRELESS IMAGE TRANSMISSION SYSTEM

Our wireless image transmission system is based on Nios II system, a 32-bit RSIC soft-core embedded processor architecture designed specifically for FPGAs by Altera. We use Qsys to construct our Nios II system, where the core selection is the Nios II/f, and use the 50Mhz crystal clock on the DE2-115 board as the operating frequency of processor and SDRAM. According to Altera's manual [14], we understand that the on-board SDRAM of DE2-115 has two chips, each with 8M × 16bit × 4bank capacity, and can operate at 50 MHz, 100 MHz, and 133 MHz.

The image encryption wireless transmission system that we have finally completed includes a SD Card I/O interface, a VGA display interface, a wireless transmission interface, and an encryption algorithm. The implementation of these interfaces and algorithms will be introduced one by one in the following subsections.

1) SD CARD INTERFACE

The SD Card interface in our system is responsible for the sender's plaintext picture read, ciphertext picture write, and the plaintext picture that the receiver decrypts from the ciphertext. We use the IP core of the Altera University Program and apply the SD Card interface to our system according to the manual [15]. However, this IP core can only support the FAT16-baed portable driver, and only has the function of writing a file, and cannot create corresponding system data resulting in the file, which cannot be opened in the windows operating system. Therefore, when writing the file such as a BMP, we need to create a blank file in the SD card in advance through the windows operating system for the IP core to write. The file written by this method will have an additional 0-byte tuple when closing the file to use the alt_up_sd_card_fclose() function provided by the IP core, so we do not use this function.

2) VGA DISPLAY INTERFACE

The VGA display interface is used to instantly display the plaintext and ciphertext images at the sending end and

TABLE 4. Resource consumption with two method.

Method	Latency	Arithmetic Unit	Step						Total	Saved Units
			1	2	3	4	5	6		
Intuitive	38	Adder	-	1	2	2	-	2	8	
		Subtractor	2	1	1	1	-	-	4	-
		Multiplier	4	4	2	-	2	-	12	
Units reused	33	Adder	-	3	1	-	4		4	4
		Subtractor	1	-	2+1	-	-	-	3	1
		Multiplier	5	3	-	4	-		8	4

the receiving end. In this interface, we also use the IP core of the Altera University Program, which supports image and video display. According to the manual [16], we use 2MB of SRAM as the Pixel Buffer, and connect the DMA Controller, RGB Resampler, and Dual-clock FIFO in order to convert the 24-bit RGB data to the 30-bit VGA RGB data. The Dual-clock FIFO is responsible for the data transmission of the 50MHz processor and 25MHz VGA port. The VGA port operates at 25MHz due to the 640×480 resolution we use.

3) WIRELESS TRANSMISSION INTERFACE

The wireless transmission interface is responsible for the transmission of ciphertext data between the receiving end and the sending end. Here we chose to use Terasic RFS daughter card, which includes gyroscopes, temperature and humidity sensors, Bluetooth module, and ESP8266 wireless transmission module. According to the manual [17], [18], we use the RS-232 serial port as UART channel for transferring instructions and data between the Nios II system and the ESP8266.

In testing the wireless transmission module, we found the following three points:

a) The ESP8266 does not support 5G wireless network signals. Only 2.4G wireless network signals can be used.
b) ESP8266 comes with 50 KB memory as a buffer for receiving and transmitting, as well as store system configuration parameters and AP mode site-local data.
c) Although the transmission speed can reach 2048 KB packets every 20 ms, since the data is obtained by using sprintf() function, the packets at the sending end need to be cropped in accordance with the line-feed character to avoid the loss of data received by the receiving end.

Ultimately, our wireless interface receives user commands through the Terminal Console of the Nios II system. The interface will search for and connect to the specified Wi-Fi signal source after the Nios II system is started. After the connection, the interface will automatically obtain the IP address. The receiver will request to open the Port number of the receiving data, and the sending end will send the ciphertext to the receiving port after the encryption.

4) ENCRYPTION ALGORITHM

According to Section II, we will implement our encryption algorithm as four functions such as Get_Chaos_Key(), XOR_BMP(), BI_Encryption(), and BI_Decryption(). Among them, Get_Chaos_Key() will input the transcoded Shift Value to the Chaos_Key_Generator module on the FPGA, and obtain the key needed for encryption from the module. XOR_BMP() is a function used to perform XOR operation on BMP images. This function will skip the BMP file header and only encrypt the content so that we can use the picture viewer to view the ciphertext directly on the computer. BI_Encryption() and BI_Decryption() are the encryption function and decryption function of the Bits Insertion algorithm, respectively. The encryption function will display the two keys of the encrypted file to the user through Terminal Console. These two keys are the length of the plaintext and the characteristic value of ciphertext before the Bits Insertion encryption. The decryption function will ask the user for the two keys corresponding to the ciphertext received to decrypt. Source code posted on my Github: https://github.com/M10512001/DE2-115.

IV. SECURITY ANALYSIS

In this section, we will verify the security of our proposed encryption algorithm through histogram analysis, correlation analysis, differential attack analysis, and entropy analysis.

The test images we use here are Lena, Mandrill (a.k.a. Baboon), Airplane, and Peppers, taken from The Lenna Story [19] and The USC-SIPI Image Database [20], respectively. And these images are tested after convert file format from tiff to 24-bits BMP with Microsoft Paint.

A. HISTOGRAM ANALYSIS

The distribution of pixels is reflected in the RGB histogram, while the non-uniform distribution of pixels represents the characteristic plaintext. Therefore, the ciphertext produced by a secure encryption algorithm should be uniform and horizontal in the histogram.

The Fig. 8 and Fig. 9 are histogram tests of Lena and Airplane, respectively, and (a) and (b) are the RGB histograms of the plaintext and its chipertext encrypted by our proposed chaotic encryption. It can be seen that our proposed chaotic encryption is safe in histogram analysis.

B. CORRELATION ANALYSIS

The correlation coefficient represents the linear intensity and direction between the two variables. So, any two adjacent pixels in the horizontal, vertical, or diagonal directions of the plaintext image are often predictable, which means that the absolute value of the correlation coefficient of these

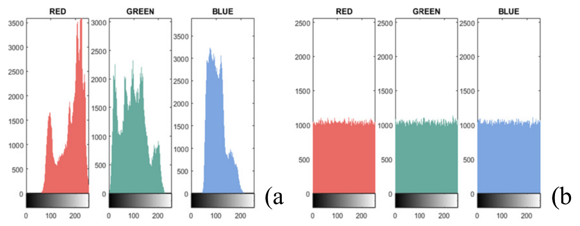

FIGURE 8. Histogram of Lena and its ciphertext encrypted by our proposed chaotic encryption.

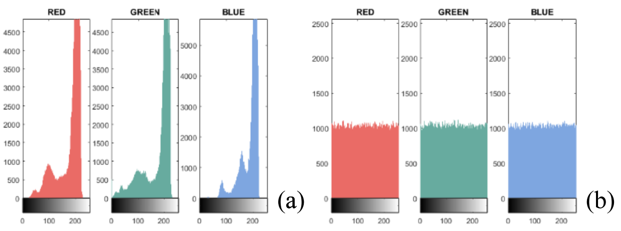

FIGURE 9. Histogram of Airplane and its ciphertext encrypted by our proposed chaotic encryption.

TABLE 5. Comparison of correlation coefficients on horizontal.

	Proposed	Ref. [4]	Ref. [5]	Ref. [6]
Lena	7.51×10^{-4}	4.049×10^{-4}	5.8×10^{-3}	1.587×10^{-3}
Mandrill	2.71×10^{-4}	1.2×10^{-3}	8.2×10^{-3}	1.531×10^{-2}
Airplane	1.198×10^{-3}	3.33×10^{-5}	2.3×10^{-3}	NaN
Peppers	9×10^{-5}	1.261×10^{-3}	4.5×10^{-3}	1.445×10^{-2}

two pixels will approach 1. However, almost of two adjacent pixels in the ciphertext are unpredictable, so the correlation coefficient of the ciphertext tends to be zero. And getting closer to 0 means that the security of the encryption algorithm used is higher.

We calculate the correlation coefficient through Equation (7),

$$corr\,(px, py) = \frac{cov\,(px, py)}{\sigma_{px}\sigma_{py}} \tag{7}$$

$$cov\,(px, py) = \frac{1}{n}\sum_{i=1}^{n}(px_i - \mu_{px})(py_i - \mu_{py}) \tag{8}$$

$$\sigma_{px} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(px_i - \mu_{px})^2} \tag{9}$$

$$\sigma_{py} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(py_i - \mu_{py})^2} \tag{10}$$

where $cov(px, py)$ is the covariation as in Equation (8), and σ_{px} and σ_{py} are standard deviations as shown in Equation (9) and Equation (10), μ_{px} and μ_{py} are the average of px and py, respectively.

We calculate the correlation coefficients in the three directions by choosing different relative positions of px_i and py_i.

The test results and comparisons of the ciphertext of Lena, Mandrill, Airplane, and Peppers encrypted using our proposed chaotic encryption algorithm and the studied encryption algorithms are shown in Table 5, Table 6, and Table 7.

TABLE 6. Comparison of correlation coefficients on vertical.

	Proposed	Ref. [4]	Ref. [5]	Ref. [6]
Lena	1.133×10^{-3}	1.033×10^{-3}	2.6×10^{-3}	1.470×10^{-2}
Mandrill	8.15×10^{-4}	7.427×10^{-4}	8×10^{-4}	1.732×10^{-3}
Airplane	1.5×10^{-4}	3.420×10^{-4}	3×10^{-4}	NaN
Peppers	1.92×10^{-3}	4.950×10^{-4}	1.5×10^{-3}	1.048×10^{-2}

TABLE 7. Comparison of correlation coefficients on diagonal.

	Proposed	Ref. [4]	Ref. [5]	Ref. [6]
Lena	1.253×10^{-3}	1.519×10^{-3}	2.4×10^{-3}	2.381×10^{-3}
Mandrill	1.41×10^{-4}	1.566×10^{-3}	8.3×10^{-3}	9.718×10^{-3}
Airplane	9.9×10^{-4}	2.6×10^{-3}	3.52×10^{-2}	NaN
Peppers	9.3×10^{-4}	1.6×10^{-3}	1.5×10^{-3}	4.223×10^{-3}

C. DIFFERENTIAL ATTACK ANALYSIS

Differential attack is a plaintext attack that derives the encryption key by analyzing the characteristics of the plaintext. This attack corresponds to the strength of the diffusion algorithm in the encryption algorithm, that is, the amount of change in the ciphertext caused by the small difference is in the plaintext. The security of the differential attack is mainly analyzed by the number of changing pixel rate (called as NPCR) and the unified averaged changed intensity (called as UACI). NPCR is to compare with the two ciphertext images in which there are a difference of one pixel between the two, and here, we would obtain a percentage value as Equation (11),

$$NCPR = \frac{\sum_{i}^{W}\sum_{j}^{H} D\,(i, j)}{W \times H} \times 100\% \tag{11}$$

$$D\,(i, j) = \begin{cases} 0, & \text{if } C_1\,(i, j) = C_2\,(i, j) \\ 1, & \text{if } C_1\,(i, j) \neq C_2\,(i, j) \end{cases} \tag{12}$$

where W and H share the common width and height of the ciphertext images, and i and j are the pixel positions. $D(i, j)$ is a comparison function, and Equation (12), C_1 and C_2 are two-dimensional matrices of pixel values of two ciphertext images serving for comparison.

The UACI is the average pixel distance of two ciphertext images in which there are a pixel difference between the two plaintexts, and its percentage value calculated as Equation (13),

$$UACI = \frac{1}{W \times H}\left(\sum_{i}^{W}\sum_{j}^{H}\frac{|C_1\,(i, j) - C_2\,(i, j)|}{255}\right) \times 100\% \tag{13}$$

where W, H, i, j, C_1, and C_2 are the same as in Equation (11) and Equation (12).

Then we tested 10 times the ciphertext images of Lena, Mandrill, Airplane, and Peppers encrypted by our proposed chaotic encryption algorithm with the rounding to fourth decimal place average of 10 samples, based on the safety standards of NPCR and UACI proposed by [21], and our proposed chaotic encryption passed all theoretically critical tests.

TABLE 8. Comparison of NPCR and UACI test values.

NPCR / UACI	Proposed	Ref. [4]	Ref. [5]	Ref. [6]
Lena	99.6112 / 33.3743	99.6060 / 33.4465	99.6123 / 33.4503	99.27 / 33.22
Mandrill	99.6146 / 33.3507	99.6074 / 33.4314	99.6113 / 33.4763	99.31 / 33.11
Airplane	99.6105 / 33.3523	99.6128 / 33.4253	99.6042 / 33.4618	NaN
Peppers	99.6115 / 33.3841	99.6024 / 33.4868	99.6161 / 33.4283	99.35 / 33.14

TABLE 9. Comparison of Global and Local information entropy.

Global / Local	Plaintext	Proposed	Ref[4]	Ref[5]	Ref[6]
Lena	7.271856 / 5.9494	7.999329 / 7.9023	7.999283 / NaN	7.9911 / NaN	7.999282 / 7.9021
Mandrill	7.644731 / 6.7184	7.999291 / 7.9027	7.999288 / NaN	7.9895 / NaN	7.999283 / 7.9019
Airplane	6.576839 / 5.2493	7.999288 / 7.9021	7.999249 / NaN	7.9893 / NaN	NaN
Peppers	7.297795 / 5.9721	7.999330 / 7.9027	7.999359 / NaN	7.9895 / NaN	7.999246 / 7.9024

Among them, $N^*_{\beta_n}$ and $(u^{*-}_{\beta_n}, u^{*+}_{\beta_n})$ are critical values of NPCR and UACI, respectively, and there are only three kind of β_n: $\beta_0 = 0.05$, $\beta_1 = 0.01$, and $\beta_2 = 0.001$. When our test values are lower than $N^*_{\beta_n}$ and $u^{*-}_{\beta_n}$, it means that our ciphertext has only β_n+1 probability as a completely random image, and the test values below the index of β_2 means the ciphertext image is not safe enough. And their values are $N^*_{0.05} = 99.5893\%$, $N^*_{0.01} = 99.5810\%$, $N^*_{0.001} = 99.5717\%$, $(u^{*+}_{0.05} = 33.5541\%, u^{*-}_{0.05} = 33.3730\%)$, $(u^{*+}_{0.01} = 33.5826\%, u^{*-}_{0.01} = 33.3445\%)$, and $(u^{*+}_{0.001} = 33.6156\%, u^{*-}_{0.001} = 33.3115\%)$, respectively.

However, these tests cannot completely test our ciphertext images. Take a 512 × 512 size 24bits-BMP image file as 786486 Bytes for example. The ciphertext images of Lena, Mandrill, Airplane, and Peppers encrypted by our proposed chaotic encryption algorithm exist extra 4 Bytes, 6 Bytes, 5 Bytes, and 6 Bytes, respectively. Finally, we compare our test results with the studied encryption algorithm, as shown in Table 8.

D. INFORMATION ENTROPY ANALYSIS

Information entropy is used to represent the randomness of data and is calculated as in Equation (14),

$$H(m) = -\sum_{i=1}^{L} P(m_i) \log_b P(m_i) \qquad (14)$$

where m represents the type of all existing elements in the data, the number of types is L. $P(m_i)$ means that the m_i type element accounts for the proportion in the data, that is, the probability that the element in the data is m_i. And b is the number of types of carriers used to represent the data element. For example, the bit has two types, 1 and 0, so its $b = 2$.

A 512 × 512 size 24-bits BMP image has 512 × 512 × 3 bytes representing the pixels, and the byte group of 8 bits has 256 representative types. Therefore, when we calculate the information entropy of a 512 × 512 size 24-bits BMP image, we use the parameter $L = 256$. Here b is usually used 2 or 2^8, this theoretically means each byte representing of the pixels requires 0 to 8 bits or 0 to 1 byte. When the value of $H(m)$ is closer to 8 ($b = 2$ and $L = 256$) or 1 ($b = 28$ and $L = 256$), it indicates that the data has no obvious characteristics, that is, the image data is highly random and incompressible.

However, global information entropy cannot be compared among images of different sizes or provide sufficiency to prove that any block on the ciphertext is sufficiently secure.

Therefore, we refer to the local information entropy test proposed by Ref. [22], to be that ~~is~~ randomly sample more than 30 blocks of 44 × 44 pixel size from the entire picture need calculating the average value of information entropies of those samples. This makes the test results of cipher texts of different sizes comparable in local information entropy.

When the average information entropy (called local information entropy) is greater than 7.9, it means that all parts of the image have a high degree of randomness, that is, the tested ciphertext has sufficient security. In summary, the local information entropy is more reliable than the global information entropy in verifying the security of ciphertext. Finally, we compare the global information entropy with the local information entropy of our proposed chaotic encryption algorithm as shown in Table 9.

V. CONCLUSION

In this paper, we design a new chaotic system based on Lorenz's equations and verified it chaotic. After that, we consider the impact of floating-point operation on the chaotic system and find that difference of calculation by parts leads to different iteration values of the system after time $t > 50$s. Based on this finding, we further discussed the optimization of the chaotic system circuit that reuses the arithmetic unit, reducing the iteration time, and applies the optimized circuit to our key generator.

The key generator is designed for the consideration of the characteristics of confusion and diffusion. And it extracts the characteristics of the key application object, and then adaptively generates a key whose content and length are all changed.

We summarize the traditional chaotic encryption scheme as the permutation and XOR operation two steps. And we propose an improvement of the first encryption step, in which a permutation requiring a large number of complicated calculations is changed to be the Bits Insertion encryption with an increase in the length of the key, so that only a single scan of the file and a small amount of calculation are required to provide sufficient security and put this encryption step after XOR operation.

We then applied the key generator and our encryption algorithm to our Nios II soft-core based wireless transmission system built on the DE2-115 motherboards and RFS daughter cards. Finally, we use histogram analysis, correlation analysis, differential attack analysis, and entropy analysis to verify that our proposed encryption scheme has sufficient security.

REFERENCES

[1] C. Hun-Chen, Y. Jui-Cheng, and G. Jiun-In, "Design of a new cryptography system," in *Advances in Multimedia Information Processing—PCM*. Berlin, Germany: Springer, Dec. 2002, pp. 1041–1048.

[2] M. Feki, B. Robert, G. Gelle, and C. M. Colas, "Secure digital communication using discrete-time chaos synchronization," *Chaos, Solitons Fractals*, vol. 18, no. 4, pp. 881–890, Nov. 2003.

[3] J. C. Yen, H. C. Chen, and S.-M. Wu, "Design and implementation of a new cryptographic system for multimedia transmission," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2005, pp. 6126–6129.

[4] C.-H. Yang and S.-J. Huang, "Secure color image encryption algorithm based on chaotic signals and its FPGA realization," *Int. J. Circuit Theory Appl.*, vol. 42, no. 12, pp. 2444–2461, Dec. 2018. doi: 10.1002/cta.2572.

[5] C. Dong, "Color image encryption using one-time keys and coupled chaotic systems," *Signal Process., Image Commun.*, vol. 29, no. 5, pp. 628–640, May 2014.

[6] R. Boriga, A. C. Dăscălescu, and I. Priescu, "A new hyperchaotic map and its application in an image encryption scheme," *Signal Process., Image Commun.*, vol. 29, no. 8, pp. 887–901, Sep. 2014.

[7] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Phys. D, Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, Jul. 1985.

[8] Calculation. (2004). *Calculation Lyapunov Exponents for ODE*. [Online]. Available: https://ww2.mathworks.cn/matlabcentral/fileexchange/4628-calculation-lyapunov-exponents-for-ode

[9] M. Preishuber, T. Hütter, S. Katzenbeisser, and A. Uhl, "Depreciating motivation and empirical security analysis of chaos-based image and video encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2137–2150. Sep. 2018.

[10] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[11] V. Gupta, D. Stebila, S. Fung, S. C. Shantz, N. Gura, and H. Eberle, "Speeding up secure Web transactions using elliptic curve cryptography," in *Proc. NDSS*, vol. 93, no. 11, Feb. 2004, pp. 231–239.

[12] Microsoft. (2018). *Default Cluster Size for NTFS, FAT, and exFAT*. [Online]. Available: https://support.microsoft.com/en-us/help/140365/default-cluster-size-for-ntfs-fat-and-exfat

[13] *IEEE Standard for Ethernet*, IEE Standards Association, IEEE Standard 802.3-2012, 2018, pp. 108–111. [Online]. Available: https://ieeexplore.ieee.org/document/6419735/

[14] Altera University Program. (2015). *Using the SDRAM on Altera's DE2-115 Board with Verilog Designs [Ebook]*. [Online]. Available: ftp://ftp.altera.com/up/pub/Intel_Material/15.1/Tutorials/Verilog/DE2-115/Using_the_SDRAM.pdf

[15] Altera University Program. (2015). *Altera University Program Secure Data Card IP Core [Ebook]*. [Online]. Available: ftp://ftp.altera.com/up/pub/Altera_Material/15.1/University_Program_IP_Cores/Memory/SD_Card_Interface_for_SoPC_Builder.pdf

[16] Altera University Program. (2014). *Altera University Program Video IP Cores [Ebook]*. [Online]. Available: ftp://ftp.altera.com/up/pub/Intel_Material/14.1/University_Program_IP_Cores/Audio_Video/Video.pdf

[17] Terasic. (2016). *RFS User Manual [Ebook]*. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=1025&FID=d1d10684aa5c6a87efc407b52d504104

[18] Espressif Systems. (2018). *ESP8266 AT Instruct. Set [Ebook]*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

[19] C. Rosenberg (2011), *The Rest Lenna Story*. [Online]. Available: http://www.lenna.org/

[20] The USC-SPI Image Database. (2018). *SIPI Image Database*. [Online]. Available: http://sipi.usc.edu/database/database.php

[21] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber J., Multidisciplinary J. Sci. Technol. J. Sel. Areas Telecommun.*, vol. 1, no. 2, pp. 31–38, Jan. 2011.

[22] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon entropy measure with statistical tests for image randomness," *Inf. Sci.*, vol. 222, pp. 323–342, Feb. 2013.

[23] Z. Hua, B. Zhou, and Y. Zhou, "Sine-transform-based chaotic system with FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 65, no. 3, pp. 2557–2566, Mar. 2018.

[24] P. Gope and T. Hwang, "A realistic lightweight anonymous authentication protocol for securing real-time application data access in wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 63, no. 11, pp. 7124–7132, Nov. 2016.

[25] D. Liu, Z. Liu, Z. Yong, X. Zou, and J. Cheng, "Design and implementation of an ECC-based digital baseband controller for RFID tag chip," *IEEE Trans. Ind. Electron.*, vol. 62, no. 7, pp. 4365–4373, Jul. 2015.

[26] H. Kim, T.-H. Ki, S. Lee, and H.-S. Lee, "CMOS security-enhanced passive (SEP) tag supporting to mutual authentication," *IEEE Trans. Ind. Electron.*, vol. 61, no. 9, pp. 4920–4930, Sep. 2014.

[27] U. Kretzschmar, A. Astarloa, J. Jiménez, M. Garay, and J. D. Ser, "Compact and fast fault injection system for robustness measurements on SRAM-based fpgas," *IEEE Trans. Ind. Electron.*, vol. 61, no. 5, pp. 2493–2503, May 2014.

[28] K.-K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial Internet-of-Things: Research challenges and opportunities," *IEEE Trans. Ind. Inform.*, vol. 14, no. 8, pp. 3567–3569, Aug. 2018.

[29] K. Gai and M. Qiu, "Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3590–3598, Aug. 2018.

[30] Y. Yang, X. Liu, and R. H. Deng, "Lightweight break-glass access control system for healthcare Internet-of-Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3610–3617, Aug. 2018.

[31] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3618–3627, Aug. 2018.

[32] R. Zhou, X. Zhang, X. Du, X. Wang, G. Yang, and M. Guizani, "File-centric multi-key aggregate keyword searchable encryption for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3648–3658, Aug. 2018.

[33] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. Wang, and S. W. Baik, "Secure surveillance framework for IoT systems using probabilistic image encryption," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3679–3689, Aug. 2018.

[34] N. Zhou, H. Jiang, L. Gong, and X. Xie, "Double-image compression and encryption algorithm based on co-sparse representation and random pixel exchanging," *Opt. Lasers Eng.*, vol. 110, pp. 72–79, Nov. 2018.

[35] L. Gong, X. Liu, F. Zheng, and N. Zhou, "Flexible multiple-image encryption algorithm based on log-polar transform and double random phase encoding technique," *J. Mod. Opt.*, vol. 60, no. 13, pp. 1074–1082, 2013.

[36] Y. Zhuang, N. Jiang, Q. Li, H. Hu, and K. W. D. Chiu, "Towards professionally user-adaptive large medical image transmission processing in mobile telemedicine systems," *Multimedia Syst.*, vol. 24, no. 2, pp. 123–145, 2018.

[37] S. Amina and F. K. Mohamed, "An efficient and secure chaotic cipher algorithm for image content preservation," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 60, pp. 12–32, Jul. 2018.

[38] E. Yavuz, R. Yazici, M. C. Kasapbaşi, and E. Yamaç, "A chaos-based image encryption algorithm with simple logical functions," *Comput. Electr. Eng.*, vol. 54, pp. 471–483, Aug. 2016.

[39] H. Liu and X. Wang, "Color image encryption based on one-time keys and robust chaotic maps," *Comput. Math. Appl.*, vol. 59, no. 10, pp. 3320–3327, May 2010.

[40] J. S. Khan and J. Ahmad, "Chaos based efficient selective image encryption," *Multidimensional Syst. Signal Process.*, vol. 30, no. 2, pp. 943–961, Apr. 2019. doi: 10.1007/s11045-018-0589-x.

[41] H. Liu and X. Wang, "Color image encryption using spatial bit-level permutation and high-dimension chaotic system," *Opt. Commun.*, vol. 284, nos. 16–17, pp. 3895–3903, 2011.

[42] X.-Y. Wang, L. Yang, R. Liu, and A. Kadir, "A chaotic image encryption algorithm based on perceptron model," *Nonlinear Dyn.*, vol. 62, no. 3, pp. 615–621, 2010.

[43] J. S. Khan, M. A. Khan, J. Ahmad, S. Hwang, and W. Ahmed, "An improved image encryption scheme based on a non-linear chaotic algorithm and substitution boxes," *Informatica*, vol. 28, no. 4, pp. 629–649, Jan. 2017.

[44] S. Bahrami and M. Naderi, "Image encryption using a lightweight stream encryption algorithm," *Adv. Multimedia*, vol. 2012, 2012, Art. no. 767364. doi: 10.1155/2012/767364.

[45] S. Janakiraman, K. Thenmozhi, J. B. B. Rayappan, and R. Amirtharajan, "Lightweight chaotic image encryption algorithm for real-time embedded system: Implementation and analysis on 32-bit microcontroller," *Microprocess. Microsyst.*, vol. 56, pp. 1–12, Feb. 2018.

Authors' photographs and biographies not available at the time of publication.