Traitement du Signal
Vol. 39, No. 3, June, 2022, pp. 805-814
Journal homepage: http://iieta.org/journals/ts

Optimized Piccolo Lightweight Block Cipher: Area Efficient Implementation

Ayoub Mhaouch[1*], Wajdi Elhamzi[2], Abdessalem Ben Abdelali[1], Mohamed Atri[3]

[1] Laboratory of Electronics and Microelectronics (EμE), Faculty of Sciences of Monastir, University of Monastir, Monastir 5000, Tunisia
[2] College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University Al-kharj, Al-kharj 11942, Saudi Arabia
[3] College of Computer Science, King Khalid University, Abha 61421, Saudi Arabia

Corresponding Author Email: wajdi.elhamzi@essths.rnu.tn

https://doi.org/10.18280/ts.390305

Received: 1 March 2022
Accepted: 16 May 2022

Keywords:
piccolo block cipher, serial architectures, hardware implementation, low-resource, VHDL, FPGA, security analysis

ABSTRACT

Piccolo algorithm is one of the lightweight block ciphers designed specifically for low-resource devices which present physical constraints in terms of area, power, and memory. Various hardware architectures for Piccolo block cipher have been proposed in recent years with the aim of obtaining a more appropriate low-resource design for specific constrained applications. The latter must meet real-time processing constraints without affecting the need for hardware resources. Finding a good compromise between computation time and implementation resource consumption is a major consideration in the design process. In this paper, we suggest six serial hardware architectures for Piccolo lightweight algorithm with a 128 bits key length. Proposed architectures are compared to existing designs based on hardware resource occupancy, latency, and throughput. Also, we tested the security of the Piccolo algorithm, and the obtained results show the good robustness of the Piccolo block cipher against statistical attacks. Thus, we can use the Piccolo algorithm in lightweight applications that require a high level of privacy.

1. INTRODUCTION

Recent advances in technology have boosted the need for mobile devices with limited computation resources. Such devices are the basic component of the internet of things (IoT) [1]. The idea behind connecting objects through sensors has been raised in the past decade but many challenges have blocked the way to reach this achievement. The development of IoT was very slow because of the unavailability of fast connectivity and small effective sensors with low power consumption. Sensors such as RFID tags were considered a feature key for connecting billions of objects. RFID tags are low-power devices that can connect wirelessly. Combining such devices with the available internet, cellular, and wireless networking has solved most of the problems.

Connecting a huge number of objects has many concerns. Devices must be equipped with a limited computation processor to be able to run at low power, since most of them are based on batteries [2]. Also, the connection between objects must be performed securely without any overlapping to maintain the continuity and safety of data communication [3, 4]. In this context, encryption algorithms for IoT devices must run in real-time. Nevertheless, ordinary cryptography algorithms were designed to run on powerful devices without any power consumption limit. Algorithms such as AES [5] and DES [6] were mostly used for data encryption in cloud-based systems [7]. However, they cannot be used for limited computation devices and real-time systems. The need for optimized cryptography techniques taking into consideration the requirements of constrained IoT applications is highly imposed.

In this paper, we propose an optimized lightweight cryptography algorithm suitable for IoT devices. The proposed technique is based on the PICCOLO algorithm [8]. The latter has been designed for hardware implementation to deal with low computation needs. The algorithm can be implemented in different ways: The parallel implementation which is used for speed purposes and the iterative one which is designed for low area occupation. The suggested algorithm aims to reduce the hardware implementation area occupation, while maintaining a good level of processing speed.

Optimized iterative implementation is proposed to achieve a trade-off between processing speed, computation resources, and power consumption. Six different implementations were developed and two optimization techniques were applied. The first optimization technique is based on using different data paths. First, a single data path was used to load the data, and after, the data path was divided into two parts to load the data. The second optimization technique is based on changing the number of bits used to load the data: 8-bits, 16-bits, and 32-bits were used to implement the algorithm.

The original implementation of the PICCOLO algorithm was performed on ASIC. The parallel implementation requires 757.75 GE (gate equivalent) and achieved a throughput of 12.12 Kbps and 528 clock cycles of latency, while the iterative implementation occupies 1196.50 GE and has a throughput of 193.94 Kbps and 33 clock cycles of latency.

The rest of the paper is organized as follows: the background of the Piccolo algorithm will be presented in section 2. The proposed hardware design will be detailed in section 3. The experiments and the result comparison will be presented in section 4. Finally, the paper's content will be concluded in section 5.

2. DESCRIPTION OF PICCOLO ALGORITHM

Piccolo's algorithm is a symmetric lightweight block cipher, that was introduced by Shibutani et al. in CHES 2011 [8]. It is an iterative block cipher based on a Feistel network structure. To encrypt any message by Piccolo block encoder, the Piccolo algorithm needs a 64-bit block size of plaintext to initiate the encryption process with a variable key size of 128 bits or 80 bits. In addition, Piccolo has 31 or 25 rounds for encryption or decryption of a message with a 128-bit and 80-bit key length, respectively.

Piccolo block cipher uses the same data path as the encryption process for the decryption operation of ciphertext, whereas the difference is in the key scheduling, as the decryption process uses inverse key scheduling compared to the encryption process. The round keys can be calculated from the input key by the key scheduling part, which consists of Multiplexers and XOR operations to generate two subkeys for each round. The specification for the key scheduling algorithm of Piccolo is presented in Alg.1 and Alg.2 for a key size of 128-bit and 80-bit, respectively.

Datapath of Piccolo block cipher has two branches of Feistel structure, which is illustrated in Figure 1. Each branch of the Feistel structure consists of two layers S-Box, MixColunns, AddSubKey and RP.

S-Box: In this function, every four bits of the message are converted to another value in output by Piccolo's substitution. The S-Box converts the input to an output value based on a predefined Look-Up-Table. The Look-Up-Table used for the conversion of the S-Box is showcased in Table 1. It is obtained based on statistical analyzes whose objective is to minimize the probability of correlation between the plaintext and the ciphertext.

Algorithm 1. Round key operation for Piccolo-128 with r = 31.

```
1  Input: A 128-bit secret key K.
2  $wk_0 = k_0^L|k_1^R$ , $wk_1 = k_1^L|k_0^R$ , $wk_2 = k_4^L|k_7^R$ , $wk_3 = k_7^L|k_4^R$
3  For i = 1 to r do
4      If ((I + 2) mod 8 = 0)
5          $(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7) \leftarrow (k_2, k_1, k_6, k_7, k_0, k_3, k_4, k_5)$
6      End if
7  End for
8  $rk_i \leftarrow k_{(i+2)mod\ 8} \oplus con_i^{128}$
9  Return $rk_i$, $Wk_i$
```

Algorithm 2. Round key procedure for Piccolo as described in [8] with r = 25 for Piccolo-80.

```
1   Input: A 128-bit secret key K.
2   $wk_0 = k_0^L|k_1^R$ , $wk_1 = k_1^L|k_0^R$ , $wk_2 = k_4^L|k_3^R$ ,$wk_3 = k_3^L|k_4^R$
3   For i = 1 to r do
4       If (i mod 8 = 3)
5           $Sk_i \leftarrow (k_4, k_4)$
6       elsif (i mod 8 = 0 or 2)
7           $Sk_i \leftarrow (k_2, k_3)$
8       else
9           $Sk_i \leftarrow (k_0, k_1)$
10      End if
11  End for
12  $rk_i \leftarrow Sk_{(i)} \oplus con_i^{80}$
13  Return $rk_i$, $Wk_i$
```

Table 1. Substitution box of Piccolo block cipher

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S	E	4	B	2	3	8	0	9	1	A	7	F	6	C	5	D

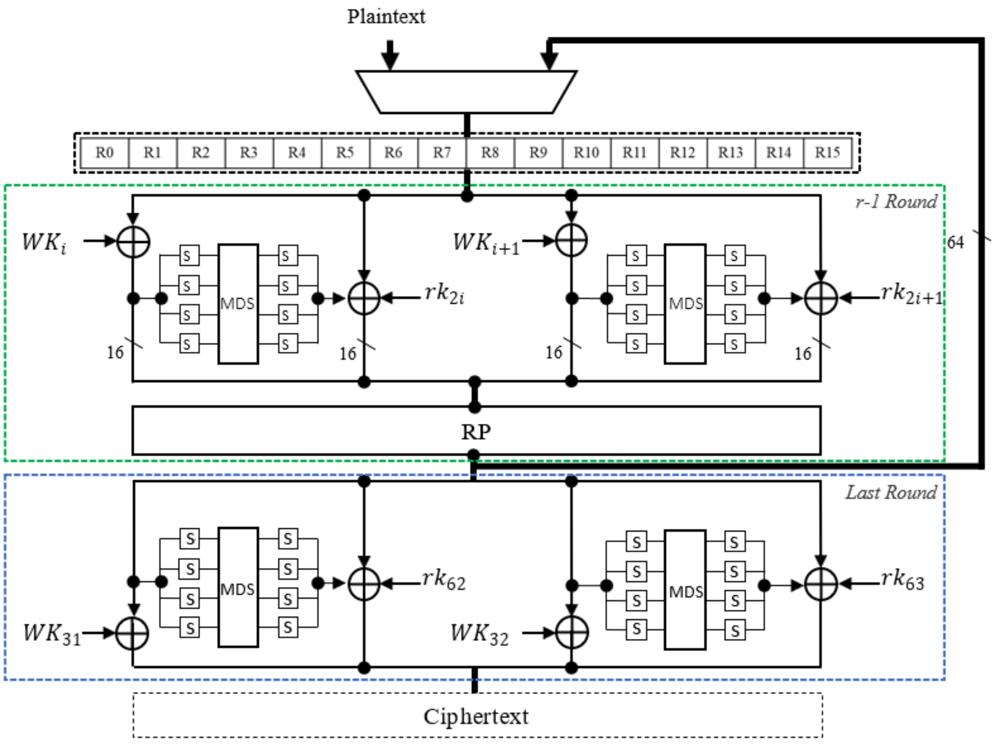


Figure 1. Piccolo encryption process

MixColumnns: The MixColumns function uses a 16-bit input to generate another one with the same size. This function is used to multiply the input value by the diffusion matrix MD. The MixColumns operation is defined using the following equation:

$$\begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} * \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} V_0' \\ V_1' \\ V_2' \\ V_3' \end{pmatrix} \qquad (1)$$

INPUT MD OUTPUT

RP: RP function consists in rearranging all bytes in the message, as listed in Table 2.

Table 2. RP function of Piccolo block cipher

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Byte 2	Byte 7	Byte 4	Byte 1	Byte 6	Byte 3	Byte 0	Byte 5

AddSubKey: This process performs to add sub-keys for each round (RKs and WKs) to a message by the XOR operation.

3. PROPOSED HARDWARE IMPLEMENTATION OF PICCOLO ALGORITHM

3.1 Single path for data encryption process

In this section, we detailed the use of a single path to encrypt the data and discuss its impact on different hardware architectures of Piccolo block cipher.

As mentioned in the last section, the Piccolo block cipher takes input data and keys with a length of 64 bits and 128 bits, respectively. But in the proposed architectures, we generate data packages of 4 bits. So, the proposed architectures need 32 cycles to load 64 bits of data and 128 bits of the key.

The purpose of the suggested encryption process design presented in this paragraph consists of processing data by using a single path. It is composed of 64-bit registers, AND gates, XOR gates, and a single block of the F function. The AND gates are used to enable the XOR operation between subkey, data from the F function and plain data.

In this work, an improved F function hardware design that uses fewer resources compared to paper [8] has been developed. The role of this f-function in the Piccolo algorithm is to enhance the security level of the encrypted data. The mathematical expression of the F function is given by Eq. (2), while Y is a line vector of 4 4-bits values (x). Before the MixColumns function application, each X value will be substituted using the SBox.

$$\boldsymbol{F}(Y) = \boldsymbol{SBox}(\boldsymbol{MixColumns}(\boldsymbol{SBox}(Y))) \qquad (2)$$

In the proposed design, the F function architecture is based on only one S-Box layer (i.e., four S-Box) instead of 8 [8], two multiplexers and 16-bit registers as shown in Figure 2 and Figure 3, which respectively represent the single path block cipher 8-bit architecture and 16-bit architecture.

The proposed 8-bit architecture was designed to be suitable for any applications with low-resources. The inputs and output of the F function require 8 bits. It contains two S-Boxes. This design requires 8 clock cycles to generate all 64-bit of data, which gives 248 cycles as a total latency of the Piccolo algorithm with a key length of 128 bits.

The proposed 16-bit architecture for Piccolo block cipher using a single path for the data encryption process contains four S-Boxes. For this architecture, the F function block requires 8 bits as input and 8 bits as output, which minimizes the total latency of one round by a half. Compared to 8-bit architecture. The total response time of the Piccolo block cipher required 124 cycles for Piccolo-128 (Piccolo algorithm with 128 bits of key size).

In our work, we also proposed a 32-bit architecture (Figure 4) for the Piccolo block cipher implementation. In the suggested architecture, we reduced the number of Feistel network structures to one. This design requires two cycles to perform one round. As a result, this design needs 62 cycles to generate the ciphertext output of Piccolo-128.

3.2 Double paths for data encryption process

The proposed 8-bit architecture for Piccolo block cipher includes two blocks of F functions. Each block takes 4 bits input and generates a 4 bits output. It is composed of one S-Box. This design requires 8 cycles to generate an output for one round, giving 248 cycles total latency for the Piccolo algorithm with a key length of 128 bits. The detail of the proposed 8 bits hardware architecture for Piccolo block cipher using two paths of the data encryption process is presented in Figure 5.

Figure 6 is shown the proposed 16-bit architecture for Piccolo block cipher using two paths for the data encryption process. Each block of the F function contains two S-Boxes. It requires 8 bits for input and 8 bits for output, which minimize latency by half compared to 8-bit architecture. The total response time of this design required 124 cycles for Piccolo-128.

For the proposed 32-bit hardware architecture for Piccolo block cipher, Figures 7 illustrate the proposed design. This design uses an improved Feistel network architecture to encryption data. It has four S-Boxes. As a result, the proposed architecture has 8 S-Boxes for two paths. To perform the encryption of 64 bits of data, the proposed design requires two cycles to perform one round. It needs 62 cycles a total latency to generate the ciphertext output of Piccolo-128.

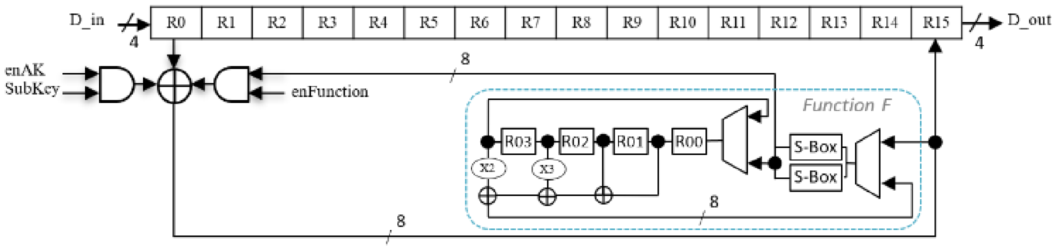


Figure 2. Proposed 8-bit architecture for Piccolo block cipher using the single path for data encryption process

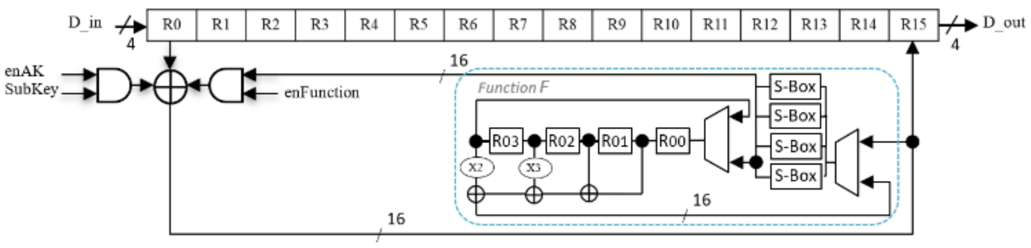


Figure 3. Proposed 16-bit architecture for Piccolo block cipher using the single path for data encryption process

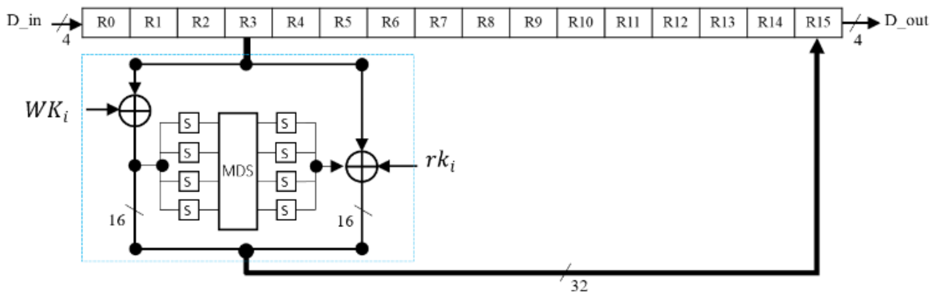


Figure 4. Proposed 32-bit architecture for Piccolo block cipher using the single path for data encryption process

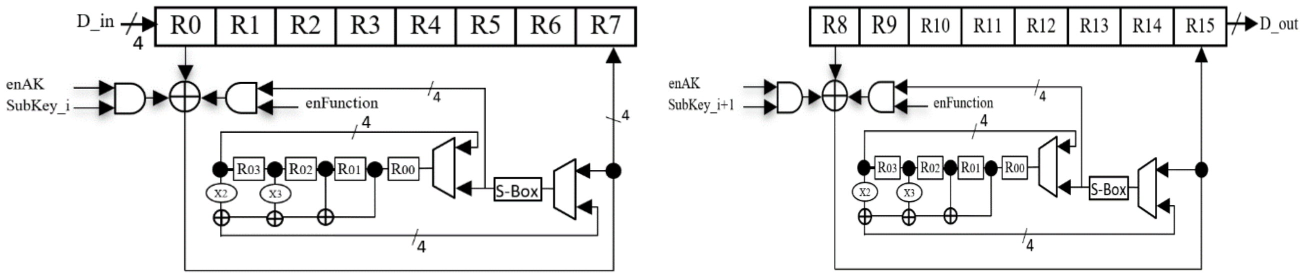


Figure 5. Proposed 8-bit architecture for Piccolo block cipher using two paths for the data encryption process

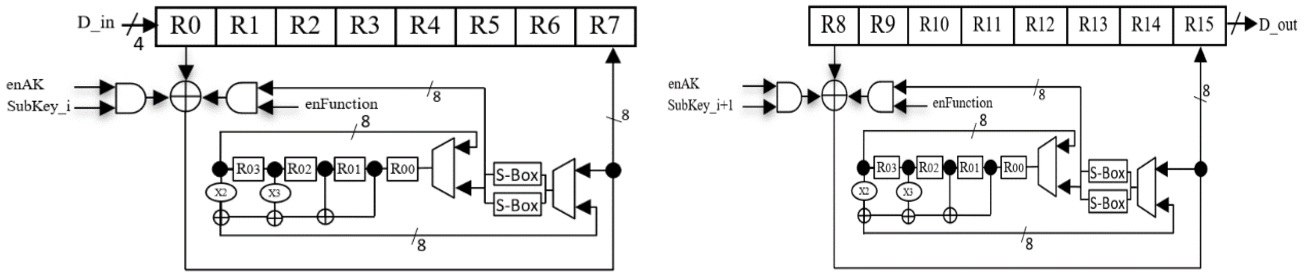


Figure 6. Proposed 16-bit architecture for Piccolo block cipher using two paths for the data encryption process

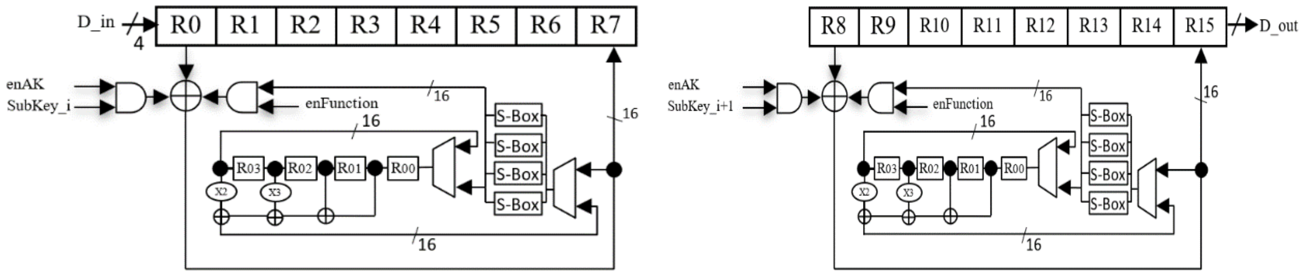


Figure 7. Proposed 32-bit architecture for Piccolo block cipher using two paths for the data encryption process

4. EXPERIMENTS AND RESULTS

4.1 Security analysis

The main goal of using the Piccolo lightweight algorithm in IoT applications is to protect the private data. In this section, we assess the security of the Piccolo algorithm against a known statistical analysis.

In order to assess the security of the Piccolo algorithm, we perform a statistical analysis of some evaluation parameters of the encrypted image by the Piccolo algorithm (using a key size of 128 bits). The proposed experiments and the performances analysis evaluation were performed on an Intel Core i5-3337U@1.80GHz processor using MATLAB 2018.

To evaluate the level of security provided by the piccolo algorithm for the encrypted image, we use the following five metrics. They are mainly based on the analysis of pixel intensities between the original images and the encrypted ones:

- Image histogram
- Image entropy
- Correlation analysis
- NPCR
- UACI

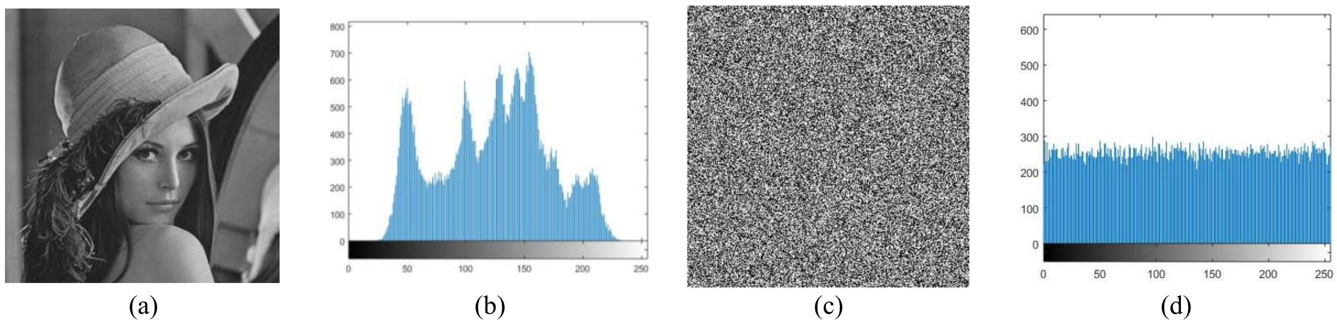


Figure 8. Histogram results of Lena's image. (a) Lena image, (b) histogram of Lena image, (c) encrypted Lena image, and (d) histogram of encrypted Lena image

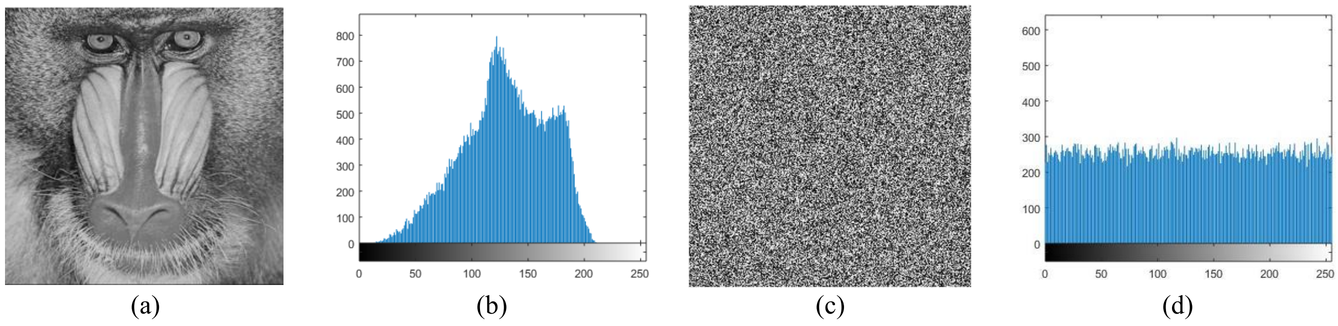


Figure 9. Histogram results of Baboon image. (a) Baboon image, (b) histogram of Baboon image, (c) encrypted Baboon image, and (d) histogram of encrypted Baboon image

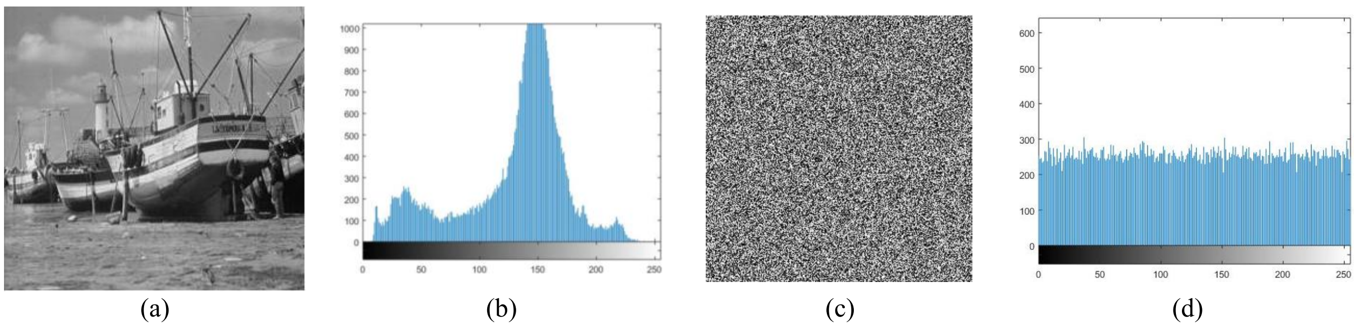


Figure 10. Histogram results of Boat image. (a) Boat image, (b) histogram of Boat image, (c) encrypted Boat image, and (d) histogram of encrypted Boat image

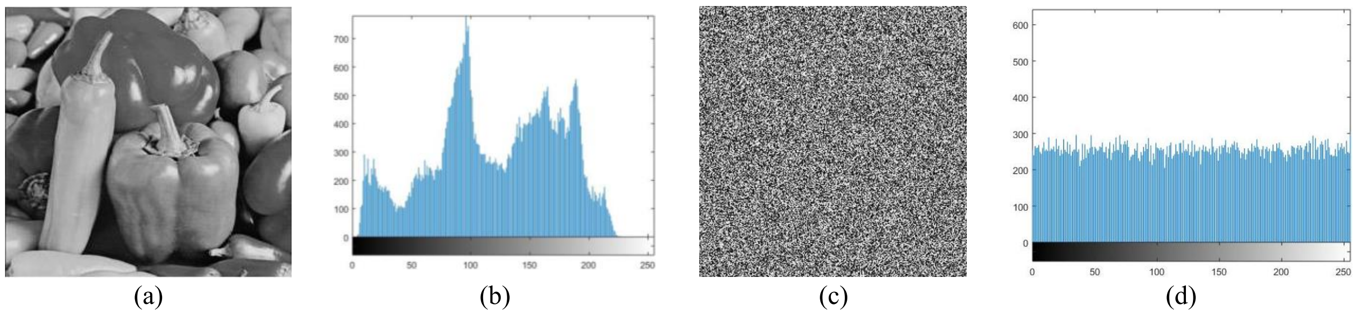


Figure 11. Histogram results of Pepper image. (a) Pepper image, (b) histogram of Pepper image, (c) encrypted Pepper image, and (d) histogram of encrypted Pepper image

4.1.1 Image histogram

This metric is especially used for reducing the probability of images attack and to hide the main information from input images. The histogram is used to evaluate the performance of the encryption process. It shows how the gray pixel levels are distributed in the image, which should be very close to the uniform distribution. Generally, extracted histogram of the encrypted image is significantly different from the input image. As shown in Figure 8 to Figure 11, compared to the histogram of the input image, the histogram of the encrypted image is totally different and flat. In this case, successful attacks may not be possible. We note that this type of analysis (based on the histogram) is highly recommended to verify the risk of attacks.

4.1.2 Image entropy

Image entropy presents an efficient metric to measure the degree of the randomness of the input image or the message in general. It is also an efficient metric used to produce a powerful cryptosystem. Generally, data that presents full entropy contributes to no meaningful features that can be extracted. The low value of entropy provides a high possibility to predict forthcoming extracted values. Entropy algorithms are highly recommended to secure encryption systems and hash functions. Entropy values must be secret or unpredictable in order to ensure a high-security process. Therefore, using entropy algorithms is critical for the cryptosystem's security. The entropy gives an idea about the degree of uncertainty. Generally, a high entropy value indicates high uncertainty.

The information entropy was introduced by Shannon [9, 10]. It is defined using the following equation:

$$\boldsymbol{H(x)} = -\sum_{i} \boldsymbol{P(x_i).\log(P(x_i))} \tag{3}$$

H(x) is the entropy of the encrypted image. The Eq. (3) is applied on the intensity (x) while $P(x_i)$ is the probability of the ith intensity x_i. The maximum entropy value is 8 for an encrypted image [11]. In this paper, Table 3 presents the results of four tested images.

4.1.3 Correlation

Correlation measures the degree of similarity between two adjacent pixels. In order to contribute to a secure system, we need to have a low correlation value between adjoining pixels. The correlation metric presents a numerical measure preceding a relationship between two variables. Good encryption is expected to remove the relationship between the original data and its encryption. So, the relationship connecting the plaintext and its encryption cannot be determined and no useful data can be extracted. In this work, we calculated the association relating the original image to its encrypted output. The correlation coefficient $r_{x,y}$ is calculated using Eq. (4).

$$\boldsymbol{r_{x,y}} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y))}{\sqrt{(\frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2)}\sqrt{(\frac{1}{N}\sum_{i=1}^{N}(y_i - E(y))^2)}}, \tag{4}$$

$$\text{Where } \boldsymbol{E(x)} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

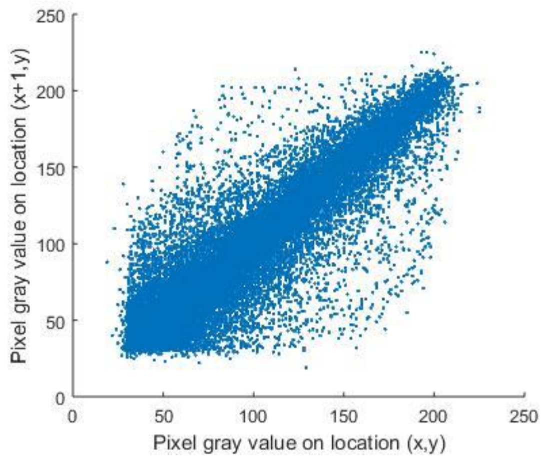
The correlation $r_{x,y}$ should be equal to 0 for the perfect cipher and it will be equal to 1 for the worst cipher. This criterion is best explained by the theory of Shannon [12].

The correlation of adjacent pixels is illustrated in Figure 12 to Figure 15. It shows the contrast between the original and the encrypted image. The original image can be thought of as having a high correlation coefficient. Whereas the encrypted image does not appear to have any correlation.

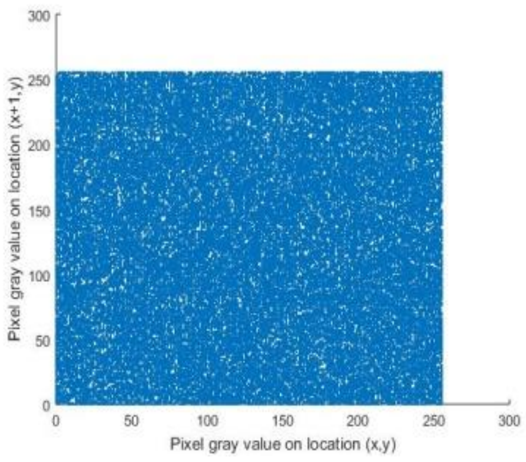
The correlation and entropy of Lena, Peppers, Baboon, and Boats of size 256 x 256 are presented in Table 3. The obtained entropies result proved the robustness of the evaluated encryption algorithm, where the achieved results are close to the maximum value. From these results, we conclude that the Piccolo block cipher has a high degree level of resilience.

Table 3. Correlation and entropy results for Lena, peppers, baboon, and boats of size 256 x 256

	Lena		**Baboon**		**Boats**		**Peppers**	
	Orig	**Encp**	**Orig**	**Encp**	**Orig**	**Encp**	**Orig**	**Encp**
H(x)	7.176	7.997	7.228	7.997	7.158	7.997	7.577	7.997
$\boldsymbol{r_{x,y}}$	0.952	0.001	0.874	0.005	0.927	0.001	0.964	0.002



(a)



(b)

Figure 12. Correlation of adjacent pixels for: (a) the plain-image of Lena, (b) the cipher-image of Lena

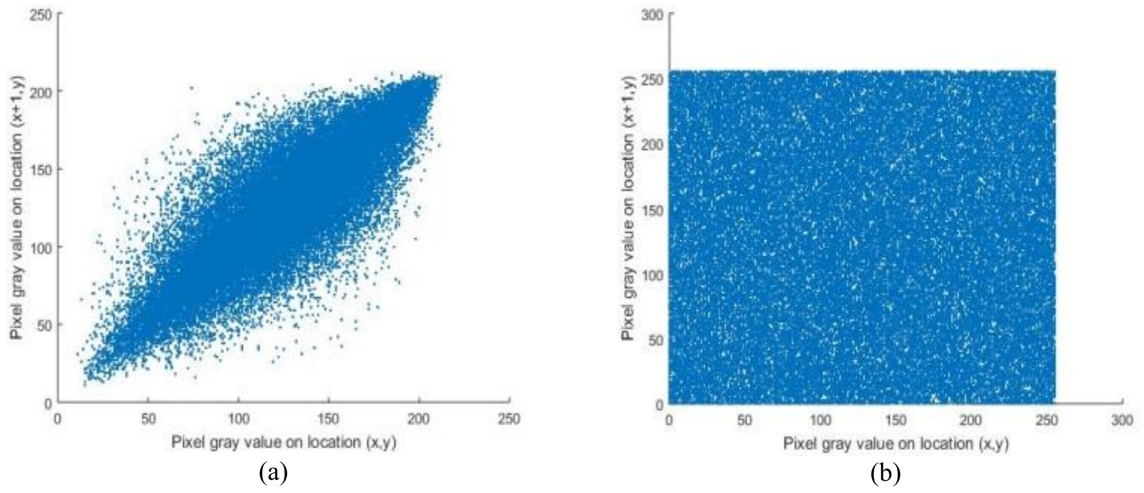


Figure 13. Correlation of adjacent pixels for: (a) the plain-image of Baboon, (b) the cipher-image of Baboon

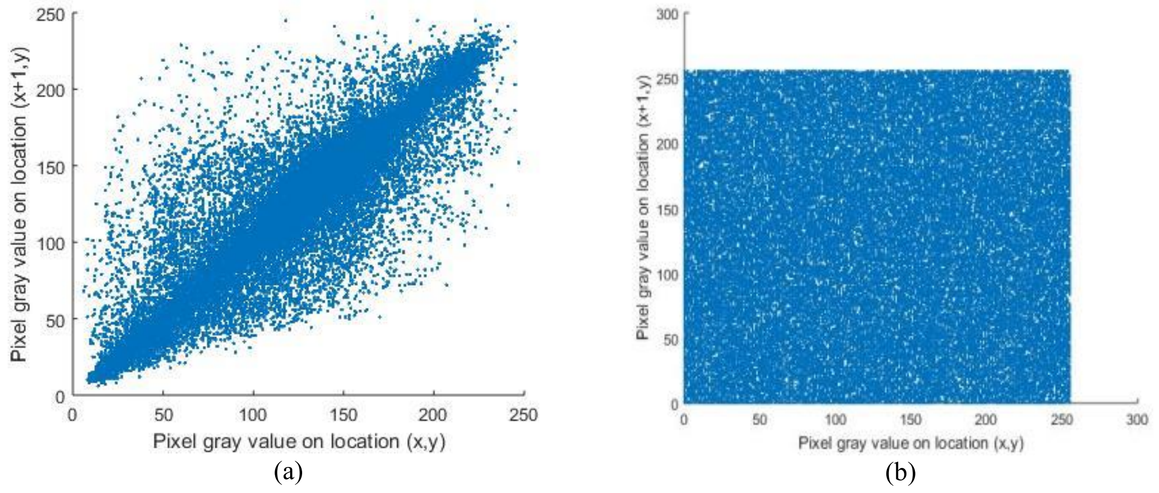


Figure 14. Correlation of adjacent pixels for: (a) the plain-image of Boats, (b) the cipher-image of Boats

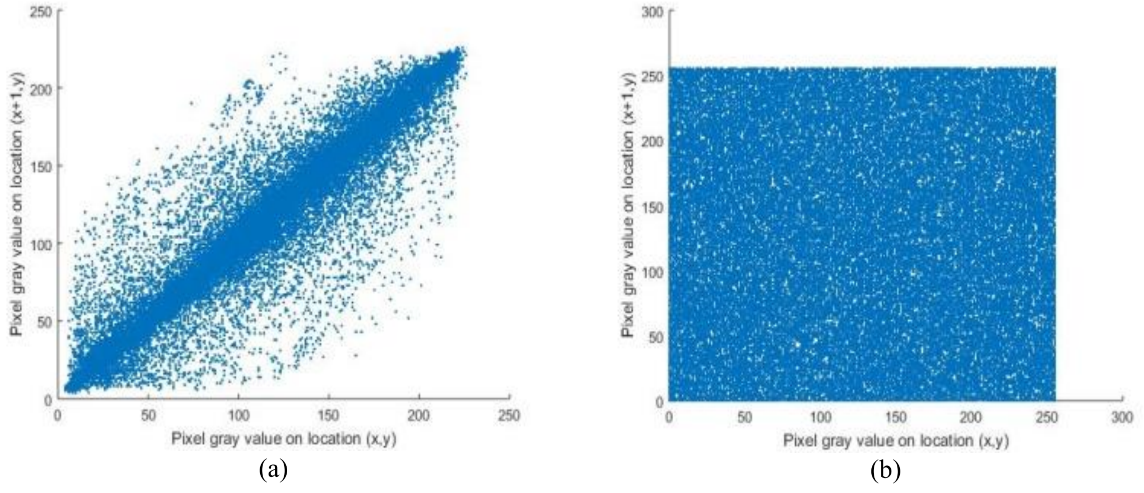


Figure 15. Correlation of adjacent pixels for: (a) the plain-image of Peppers, (b) the cipher-image of Peppers

4.1.4 NPCR and UACI

For a strong encryption system, the Piccolo algorithm should be sensitive to a light input variation, even to one-bit change in the input image. That means changing any bit in the input image will produce a different encrypted image compared to the old encrypted image. To measure Piccolo block cipher sensitivity, the number of pixel change rate (NPCR) and the unified average changing intensity (UACI) [13, 14] were deployed. These measures are defined as follows:

$$NPCR = \frac{1}{M * N} \sum_{i,j} D(i,j) * 100\% \qquad (5)$$

Table 4. Performance result of the proposed designs for piccolo block ciphers

Design	Area (Resources)					Speed		Efficiency	FPGA Device
	No. of Slices	No. of FFs	No. of LUTs	No. of LUTs+FFs	Clock Cycles	Max. Freq (MHz)	Throughput (Mbps)	Eff. (Mbps/slices)	
Proposed 8-bit architecture (section 2.A)	271	260	512	772	248	47.83	12.34	0.04	XC3S50-5
Proposed 8-bit architecture (section 2.B)	237	243	439	682	248	49.65	12.81	0.05	
Proposed 16-bit architecture (section 2.A)	279	270	524	794	124	49.07	25.32	0.09	
Proposed 16-bit architecture (section 2.B)	281	241	532	773	124	47.63	24.58	0.08	
Proposed 32-bit architecture (section 2.A)	286	206	545	751	62	69.56	71.8	0.25	
Proposed 32-bit architecture (section 2.B)	301	248	575	823	62	48.23	49.78	0.16	

Table 5. Performance result of the existing FPGA implementation for Piccolo block ciphers

Design	Key size (bits)	Datapath size (bits)	Area (Resources)				Clock Cycles	Speed		Efficiency	FPGA Device
			No. of Slices	No. of FFs	No. of LUTs	No. of LUTs+FFs		Max. Freq (MHz)	Throughput (Mbps)	Eff. (Mbps/slices)	
AES [15]	128	128	17425	-	-	-	-	196.1	25.1	1.44	Spartan-3 XC3S2000-5
Piccolo [16]	128	4	265	260	442	-	496	45.85	5.92	0.02	Spartan-3 XC3S50-5
Piccolo [16]	128	64	397	207	757	-	31	81.82	168.9	0.49	Spartan-3 XC3S50-5
Lilliput [17]	80	4	-	205	592	797	-	119.2	28	-	Spartan-3 XC3S50-5
Klein [17]	80	4	-	194	597	791	-	116	26	-	Spartan-3 XC3S50-5
AES [18]	128	8	393	-	-	-	534	-	16.86	0.04	Spartan-3 XC3S50-5

$$UACI = \frac{1}{M * N * 255}\sum_{i,j}|C_1(i,j) - C_2(i,j)| * 100\% \tag{6}$$

$$D(i,j) = \begin{cases} 1 \; if \; C_1(i,j) \neq C_2(i,j) \\ 0 \; if \; C_1(i,j) = C_2(i,j) \end{cases} \tag{7}$$

where, C_1 and C_2 correspond to two encrypted images with a one-bit difference. The achieved values of NPCR and UACI are summarized in Table 6. The obtained results show that the Piccolo algorithm has a high sensitivity to one-bit change in the input images.

Table 6. Results for the number of pixels change rate (NPCR) and unified average changing intensity (UACI)

	Lena	Baboon	Boats	Peppers
NPCR	89.76%	89.97%	90.63%	90.27%
UACI	30.04%	30.15%	30.78%	30.34%

4.2 Results of hardware implementations

In this section, the configuration of the hardware implementation is displayed. The Spartan-3 was used for the hardware implementation of the proposed architectures.

Various performance metrics are derived from the implementation results, such as area, response time and throughput. To choose the best design, Table 4 summarizes the results of the resource usage and the performance of the proposed implementations on the Spartan-3 device.

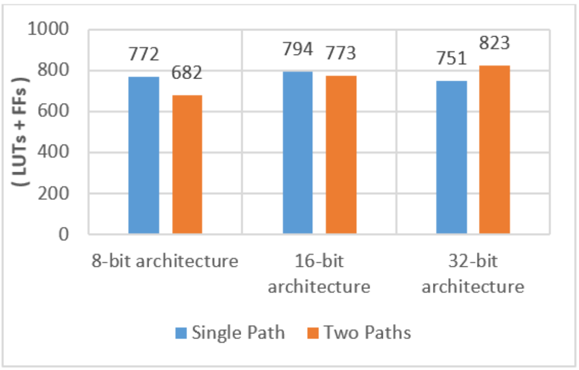


Figure 16. Occupied resources for the different implementations of the Piccolo

Figure 16 presents the occupied resource for the different developed implementations of the Piccolo.

Figure 17 presents the efficiency of the different hardware implementations of the Piccolo. The efficiency is defined as (Efficiency = throughput/slices).

The 8-bit single path implementation has occupied an area of 772 (LUTs+FFs) and the 8-bit two paths implementation has occupied 682 (LUTs+FFs). For the 16-bit architecture, the single path implementation has used 794 (LUTs+FFs) and the two paths implementation has used 773 (LUTs+FFs). The 32-bit single path implementation has occupied an area of 751 (LUTs+FFs) and the 32-bit two paths implementation has occupied 823 (LUTs+FFs).

The 32-bit single-path architecture achieves higher throughput and efficiency compared to the 16-bit single-path architecture. However, it requires more hardware resources than the 16-bit architecture.

For 8-bit and 16-bit, the proposed two-path architectures are less complex than the single-path ones. The reason is that the single-path architectures use a larger number of Substitution box units, additional multiplexers, and registers. Only the 32-bit single-path architecture makes the exception because the F Function uses only Sboxes (8) and simple logic gates for combinatorial operations. In this version of architecture, the F function is applied directly to the 32 bits of the message without having to subdivide it into slices of 16 bits or 8 bits. Therefore, there will be no need to use MUX for data routing and Registers for temporary data storage. While for the 32-bit dual-path architecture, it consists of 8 SBOx, 8 registers and 4 MUX.

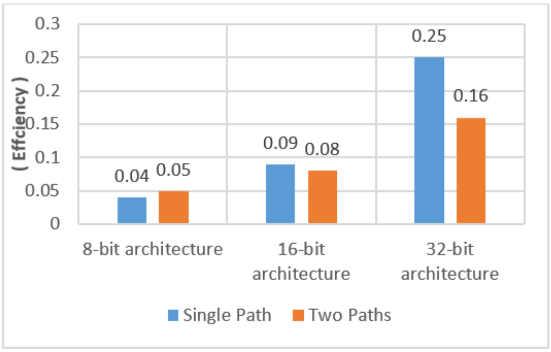


Figure 17. Comparison of the efficient implementation of the various implementations proposed

Based on the achieved results, the two paths implementations have lower resources occupancy. However, the single path implementations have a higher throughput. The two paths implementations are more efficient for hardware implementation because it provides a better trade-off between the implementation area and throughput.

Table 5 presents the results of existing FPGA implementation for some block cipher algorithms. The proposed designs present the best efficiency in terms of hardware resources implementation compared to the other designs. They achieved a high throughput (Mbps) as indicated by the results in Table 4.

5. CONCLUSION

To propose an efficient hardware architecture of the Piccolo algorithm, we have proposed six architectures. The proposed architectures were implemented on the Xilinx Spartan-3 XC3S50pq208-5 FPGA device. The proposed implementations have different paths sizes (i.e. 8-bits, 16-bits and 32-bits) to perform the encryption process. From the result of hardware implementations on FPGA, the best implementations are the architectures that use two paths for encryption processing. They present a higher performance and higher throughput/resource-occupation efficiency compared to the existing design.

In the second step, we tested the security level of the Piccolo algorithm, when it is used to encrypt an image. To assess the security of the Piccolo algorithm, certain evaluation parameters were used, such as image histogram, entropy, NPCR and UACI. The obtained results show that the Piccolo algorithm has high-level security for the analysis between the original and the encrypted images.

In future work, we plan to implement the proposed efficient architecture of the Piccolo algorithm with the VANET protocol to assure the security of the connected vehicle data.

REFERENCES

[1] Ali, Z.H., Ali, H.A., Badawy, M.M. (2015). Internet of Things (IoT): Definitions, challenges and recent research directions. International Journal of Computer Applications, 128(1): 37-47. https://doi.org/10.5120/ijca2015906430

[2] Javed, F., Afzal, M.K., Sharif, M., Kim, B.S. (2018). Internet of Things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review. IEEE Communications Surveys & Tutorials, 20(3): 2062-2100. https://doi.org/10.1109/COMST.2018.2817685

[3] Gupta, A.K., Chakraborty, C., Gupta, B. (2019). Monitoring of epileptical patients using cloud-enabled health-IoT system. Traitement du Signal, 36(5): 425-431. https://doi.org/10.18280/ts.360507

[4] Gupta, A.K., Chakraborty, C., Gupta, B. (2021). Secure transmission of EEG data using watermarking algorithm for the detection of epileptical seizures. Traitement du Signal, 38(2): 473-479. https://doi.org/10.18280/ts.380227

[5] Vincent, R., Daemen, J. (2001). Advanced encryption standard. Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp. 19-22.

[6] Smid, M.E., Branstad, D.K. (1988). Data encryption standard: past and future. Proceedings of the IEEE, 76(5): 550-559.

[7] Babitha, M.P., Babu, K.R. (2016). Secure cloud storage using AES encryption. In 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), pp. 859-864. https://doi.org/10.1109/ICACDOT.2016.7877709

[8] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T. (2011). Piccolo: An ultra-lightweight blockcipher. In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 342-357. https://doi.org/10.1007/978-3-642-23951-9_23

[9] Wu, Y., Zhou, Y., Saveriades, G., Agaian, S., Noonan, J.P., Natarajan, P. (2013). Local Shannon entropy measure with statistical tests for image randomness. Information Sciences, 222: 323-342.

https://doi.org/10.1016/j.ins.2012.07.049

[10] Zhu, C. (2012). A novel image encryption scheme based on improved hyperchaotic sequences. Optics Communications, 285(1): 29-37. https://doi.org/10.1016/j.optcom.2011.08.079

[11] Huang, X., Ye, G. (2018). An image encryption algorithm based on time-delay and random insertion. Entropy, 20(12): 974. https://doi.org/10.3390/e20120974

[12] Shannon, C.E. (1949). Communication theory of secrecy systems. The Bell System Technical Journal, 28(4): 656-715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

[13] Ye, G., Huang, X. (2017). An efficient symmetric image encryption algorithm based on an intertwining logistic map. Neurocomputing, 251: 45-53. https://doi.org/10.1016/j.neucom.2017.04.016

[14] Liu, H., Wang, X. (2011). Color image encryption using spatial bit-level permutation and high-dimension chaotic system. Optics Communications, 284(16-17): 3895-3903. https://doi.org/10.1016/j.optcom.2011.04.001

[15] Good, T., Benaissa, M. (2005). AES on FPGA from the fastest to the smallest. In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 427-440. https://doi.org/10.1007/11545262_31

[16] Mhaouch, A., Elhamzi, W., Atri, M. (2020). Lightweight hardware architectures for the piccolo block cipher in FPGA. In 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), pp. 1-4. https://doi.org/10.1109/ATSIP49331.2020.9231586

[17] Marchand, C., Bossuet, L., Gaj, K. (2017). Area-oriented comparison of lightweight block ciphers implemented in hardware for the activation mechanism in the anti-counterfeiting schemes. International Journal of Circuit Theory and Applications, 45(2): 274-291. https://doi.org/10.1002/cta.2288

[18] Kaps, J.P., Sunar, B. (2006). Energy comparison of AES and SHA-1 for ubiquitous computing. In International Conference on Embedded and Ubiquitous Computing, pp. 372-381. https://doi.org/10.1007/11807964_38