Indian Journal of Science and Technology, *Vol 8(19), DOI: 10.17485/ijst/2015/v8i19/76853, August 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

FPGA Implementation of Hiding Information using Cryptography

P. T. Thasneem Salim* **and T. Vigneswaran**

School of Electronics and Communication Engineering, VIT University, Chennai – 600127, Tamil Nadu, India; thasneem.salim2013@vit.ac.in, vigneswaran.t@vit.ac.in

Abstract

Background: The main threat in communication is the unauthorized access of information third party without the knowledge of sender and receiver. Hence the security plays a vital role in data transmission systems. Confidential data like internet banking account passwords and email account passwords needs security in their text data. Data can be text, image, video and audio. **Methods:** The purpose of this research work is to implement a mechanism to hide information (image) using cryptography. Advanced Encryption Standard (AES) is a type of symmetric cryptography standard which can be used to transfer a block of information securely during transmission. **Findings:** The idea of this paper is to generate an encrypted image by giving image input to AES encryption system and getting the decrypted image as original image by giving encrypted image as input to the AES decryption system. The input image is given through MATLAB R2012b to ModelSim6.3 and the system simulation and synthesis is done by integrating Model Sim to the Altera DE2 115 board through Quartus II software. **Conclusion:** The maximum frequency attained from this workis 165.462MHz. Finally image is displayed through an LCD which is connected to the board using Video Graphics Array (VGA) Connector.

Keywords: AES, Cryptography, Decryption, Encryption, Hiding

1. Introduction

Cryptography is a method of transferring a block of information in a secure manner, which is the science of information security and becoming a critical part in present day's computing systems for data transmission and storage[1].

Seventy percentage of the data transmission on the web representing digital images is critical parts of network exchanges. Then again, the image data, which is not quite the same as text message, has bigger size of information, higher excess and stronger relationship between pixels. That's why it is better to convert the image input to text and processing. At last the resulted text can be converted to image for getting result using tools like MATLAB. Conventional encryption algorithms, like, DES (Data Encryption Standard), IDES (International DES), are against the text messages to be proposed, which are not suitable for computerized image encryption, subsequently, a dependable advanced image with qualities is in critical need of the encryption plan AES is suitable for picture encryption, and decryption with is nearly identified with a few progress of its own attributes[2]. Exertion towards adding to the algorithm was begun by National Institute of Standards and Technology (NIST) from January 1997. AES algorithm is a symmetric key encryption algorithm, and NIST made a worldwide public call for the algorithm to supersede Data Encryption Standard (DES). At first 15 algorithms were chosen. After subtle element examination they were decreased down to 5 algorithms namely MARS, RC6, Rijndael, Serpent and Two fish. Every one of these algorithms was iterated Block ciphers[3]. The following sections are describing about the origin of AES, AES algorithm, simulation results and discussion, analysis and synthesis report and conclusion.

*Author for correspondence

Hiding the information indicates that the real existence of the image or the shape of image is hiding or masking by encrypting the image. It is not possible to find out any details about the image from this technique.

2. The Origin of AES

The major drawback of Triple DES (which was prescribed in 1999, Federal Information Processing Standard FIPS PUB 46-3 as new standard with 168-bit key) is that the algorithm is moderately slow in programming. Another downside is the utilization of 64-bit block size. For more productivity and security, a bigger block size is attractive in a way. In 1997, National Institute of Standards and Technology NIST announced a call for proposition for another Advanced Encryption Standard (AES), which ought to have security quality equivalent to or better than Triple DES, and fundamentally enhanced productivity. Likewise, NIST additionally indicated that AES must be a symmetric block cipher with a block length of 128 bits and backing for key lengths of 128, 192, and 256 bits. In a first round of assessment, 15 proposed algorithms were acknowledged. In second round it is limited to 5 algorithms. NIST completed its assessment transform and distributed a last standard (FIPS PUB 197) in November, 2001. NIST chose Rijndael as the proposed AES algorithm. Dr. Joan Daemon and Dr. Vincent Rijmen from Belgium are the two examines of the algorithm[4].

2.1 AES Evaluation

- Security – 128 insignificant key sizes gives more security.
- Cost – AES ought to have high computational efficiency.

2.1.1 Security

This alludes to the exertion needed to crypt analyze an algorithm. The emphasis in the assessment was on the common sense of the assault. Since the base key size for AES is 128 bits, brute-force attacks with present and anticipated innovation were viewed as unfeasible. Accordingly, the emphasis, regarding this point, is cryptanalysis other than a brute-force attack.

2.1.2 Cost

NIST means AES to be viable in an extensive variety of utilizations. Likewise, AES must have high computational efficiency, to be usable in rapid applications, for example, broadband links (William Stallings 20)[5].

2.2 The AES Cipher

In symmetric-key cipher, AES Algorithm both the sender and the receiver utilize a solitary key for encryption and decryption. The block length of information is altered to be 128 bits, while the length can be 128, 192, or 256 bits. What's more, the AES algorithm is an iterative algorithm. Every iteration can be known as a round, and the aggregate number of rounds is 10, 12, or 14, when key length is 128, 192, or 256, individually. The 128 bit information input is partitioned into 16 bytes. These bytes are mapped to a 4x4 array called the state, and all the inward operations of the AES algorithm are performed on the state[6]. Continuing sections are describing about the AES algorithm, simulation results and discussion, analysis and synthesis report and conclusion.

3. AES Algorithm

AES comes in three styles, to be specific AES - 128, AES - 192, and AES-256, with the number for every aspect speaking to the size (in bits) of the key utilized. All the modes are done in 10, 12 or 14 round relies upon the size of the block and the key length chosen. AES only permits a 128 bit information length that can be partitioned into four essential operation blocks. These blocks forms state by working on array of bytes and composed as a 4*4 framework or matrix[7]. The algorithm starts with an Add round key stage took after by nine rounds of four stages. Tenth round consists of three stages which applies for both encryption and decryption algorithm.

The rounds for algorithm are administered by the four stages listed below.

- Substitute Bytes.
- Shift Rows.
- Mix Columns.
- Add Round Key.

In the tenth (final) round Mix column stage is excluded. The initial nine rounds of the decryption algorithm are administered by the accompanying four stages[8].

- Inverse Substitute Bytes.
- Inverse Shift Rows.
- Add Round Key.
- Inverse Mix Columns.

Again in the last (tenth) round Inverse Mix columns stage is excluded[9]. The figure which describes the overall flow

of AES Algorithm is shown in Figure 1. Each steps of this flow are described below.

3.1 Add Round Key

The process involving in this step is just the addition of plain text (input text) and key. Exor operation is performing for addition. Here we are using 128 (a) bits of text input and 128 (b) bit key. Key for each step is derived from key schedule process[10]. The input and key are of same size to get the next state (k) is shown in Figure 2.

3.2 Substitute Bytes

Here using S-box or substitution box. This is performed mainly for converting the system into nonlinear. A 16×16 matrix of bytes are already defined by AES. Totally there are 256 numbers in that box. The function of this block is to interchange or substitute the values in state array with the corresponding values in the S-box. Figure 3. corresponds to this type substitution. Inverse substitute byte performs the inverse operation of S-box[11].

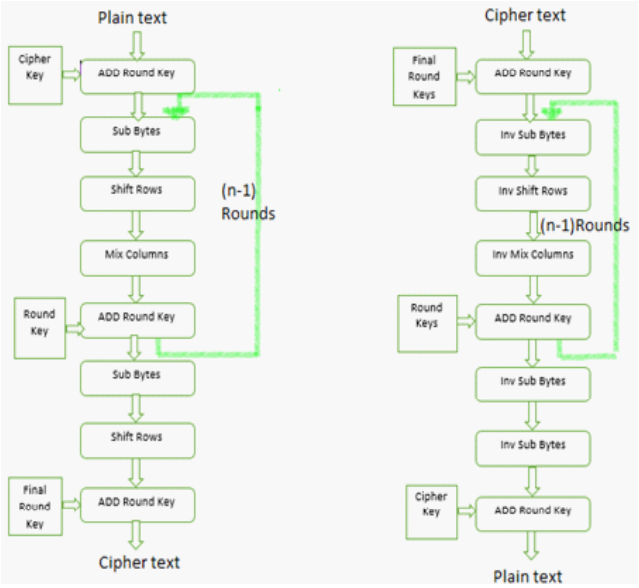

Figure 1. Flow Chart of AES Algorithm. **(a)** Encryption **(b)** Decryption.

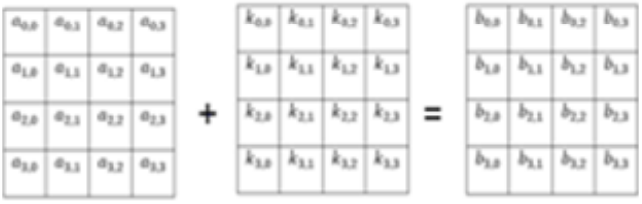

Figure 2. Add Round Key process.

3.3 Shift Rows

Row shifting operations are happening in shift rows. In this the first row of the array will not change and the continuing rows, i.e, second, third and fourth rows are shifted to left by one bit, two bits and three bits in that order[12]. Inverse shift rows performs the inverse operation of shift rows such that it shifts the second, third and fourth rows of state matrix to one, two and three bytes to the right, which is shown by Figure 4.

3.4 Mix Columns

In mix columns operations are performed on each column by column of the state array. Here each column of the state array can be considered as a polynomial in the Galois Field (GF 2^8) and the polynomials are multiplied with modulo $x^4 + 1$. The result obtained will be corresponding output

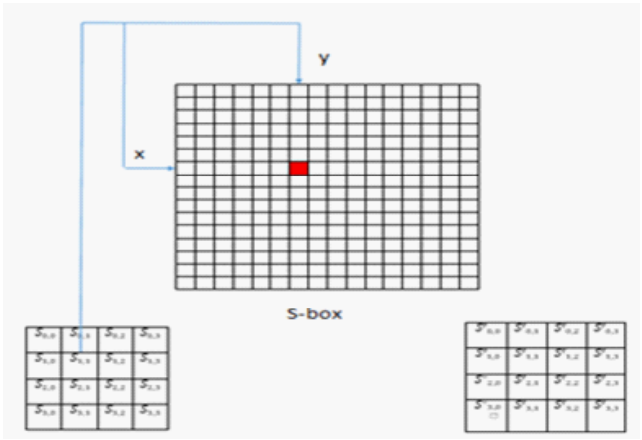

Figure 3. S-box operation.

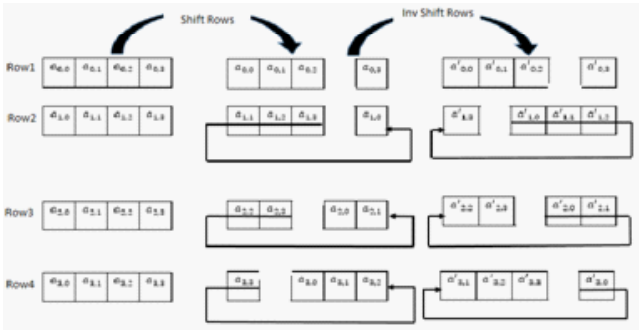

Figure 4. Shift Rows and Inv Shift Rows operation.

state of the column. Figure 5. corresponds to mix column operation[13]. Inverse mix column performs the inverse operation of mix column. Continuing sections will give simulation results and discussion, analysis and synthesis report and conclusion.

4. Results and Discussion

4.1 Simulation Results from MATLAB, Model Sim and LCD Display

Main tools used for the processing are MATLAB and ModelSim. MATLAB is used to perform image processing tasks and Model Sim environment is helpful for HDL simulation and synthesis.

It is not possible to give an image directly in to verilog. So that, at first image needs to be changed over into text file using MATLAB. Before text conversion color image is changed over to grey imageand afterward it is reshaped into one dimensional array. Then it is converted in to text file[14]. The simulation results for these techniques are given in Figure 6. below.

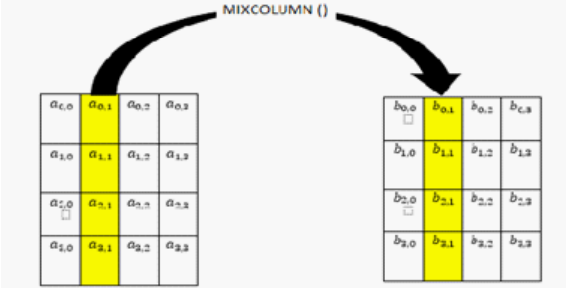

Figure 5. Mix Column operation.

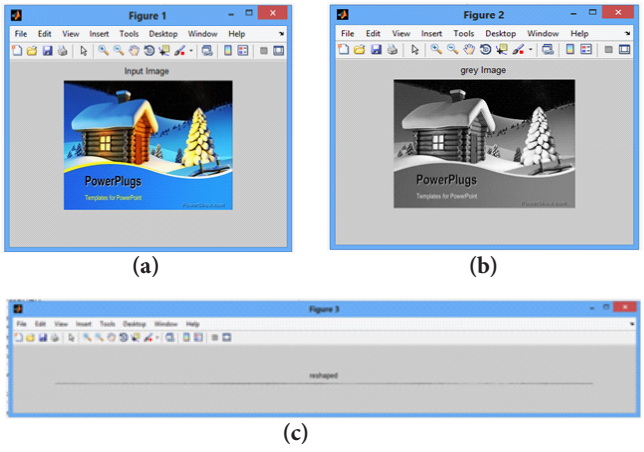

Figure 6. **(a)** Input Colour Image. **(b)** Colour to Grey Image Conversion. **(c)** Grey to one dimensional Array Conversion.

The text file from MATLAB is input to the verilog using '$read memh', this command is used to read the pixel values of image in hex format. The applied input image can also visualize in a LCD monitor using VGA (Video Graphics Array) Connector instead of viewing in MATLAB. VGA is used for display purposes, which was introduced by IBM PS/2 IN 1987. VGA Connector is 15 pin DE-15 connector, in which pins are arranged in three rows. It is used in many video cards, high definition televisions, system monitors etc. Here for our work two computers are used. First computer is connected to Altera DE2 115 Board using a USB cable and then Connect a VGA-compatible monitor to the VGA port on the DE-series board and open the project in host computer. Then download the code from system onto the board by clicking 'Download SOPC Builder System'. Then select 'Compile and Load'. Once the program is downloaded onto the system, click the run button. The visualization of input image using VGA is given in Figure 7.

The text output from MATLAB is given as input to encryption module in Verilog and is simulated using ModelSim software is given as in Figure 8. Data input for encryption is 128bits and key given is also 128 bits.

First 128 bit input and 128 bit key is given as input to add round key step, and the result (sel_1) is given as input to sub bytes. Sub bytes output is given as input to shift row, its result is given as input to mix column, the result of this and key is given as input to add round key and result (sel_2) given as input to inv sub bytes and continue the processes 10 rounds. Encryption data out is the final output of encryption process. This encryption dataout is already stored as text file in the final stage of coding in Verilog. The result of encryption process from model Sim is saved as a text in coding itself. That text is again given as input in matlab. This text file is again converted in to image in MATLAB to obtain the encrypted image as in Figure 9.

Figure 7. Input image using VGA Connector.

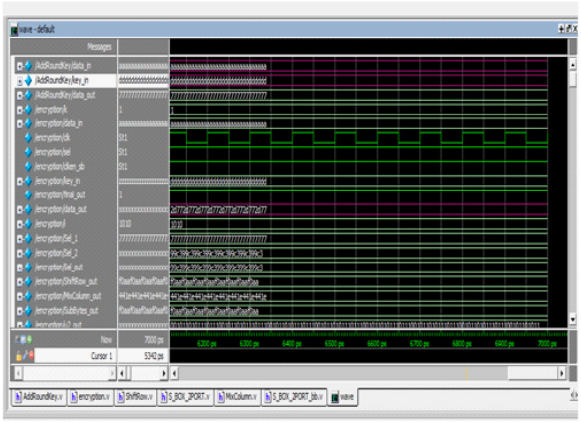

Figure 8. Encrypted simulation waveform in Model Sim.

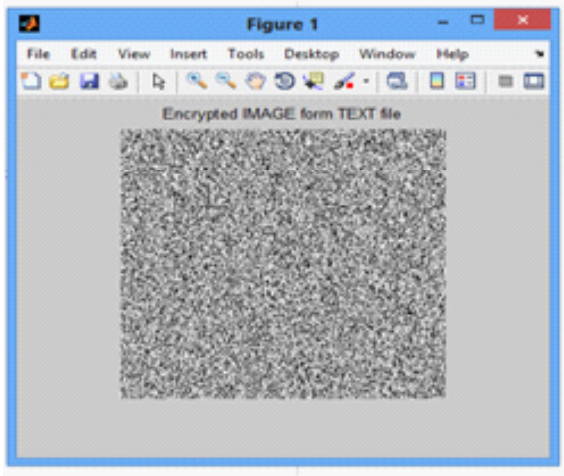

Figure 9. Encrypted image from MATLAB.

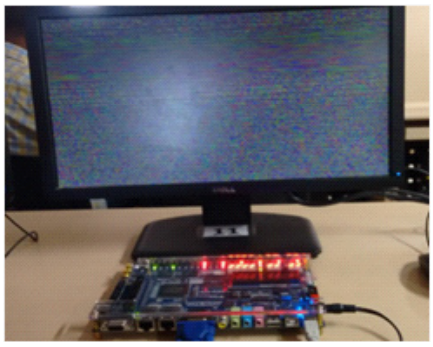

Figure 10. Encrypted image using VGA Connector.

It again needs to be converted to colour image in MATLAB using commands for giving as input to Quartus II for displaying in LCD monitor via VGA Connector. The encrypted image can be viewed in a LCD monitor using VGA Connector in Figure 10. It is obtained by connecting the VGA Connector from monitor to the Altera board and dumping the program in to another system using Quartus II software and the resultant output is connected to the Altera board as input to the board.

The RTL (Register Transfer Level) schematic of encryption module is given in Figure 11. It is obtained through Xilinx ISE Design Suite 14.3, which gives an outline deliberation which models a synchronous digital circuit as far as the stream of digital signals between hardware registers, and the logical operations performed on those signals.

Output text file from encryption module is given as input to decryption module. After encryption process, the command '$write memh' is used to write the encrypted text file in the verilog coding. This text file will give as input to the decryption module written in verilog. After decryption process using encrypted out and key, will get the result as decryption dataout, which is same as input text, which is required, is given in Figure 12. The sel_1 is the output of first add round key and sel_2 is the result of

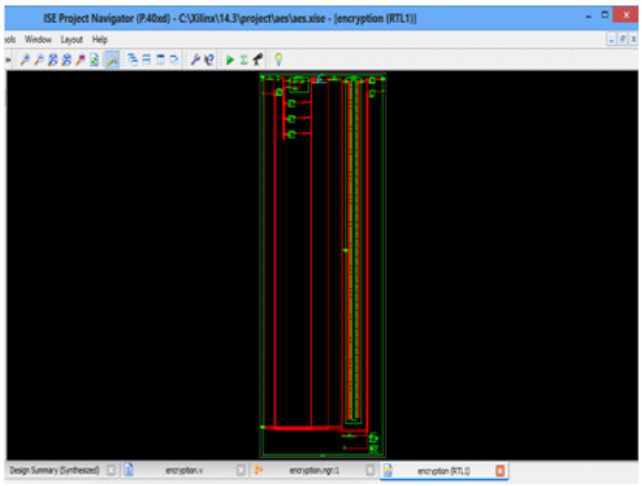

Figure 11. RTL schematic of Encryption module.

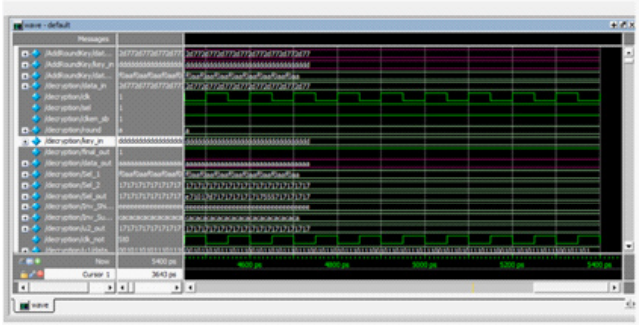

Figure 12. Decrypted simulation waveform in Model Sim.

second phase add round key and u2_out is the output of final phase add round key. Then by converting this text to image will get the decrypted image which is same as input image, which is shown in Figure 13. The visualization of decrypted image using VGA Connector is given in Figure 14.

The decrypted image seen in the LCD monitor is same as the input we have given, which is the required result. It is getting by connecting Altera board to a system which is loaded with decryption verilog code using Quartus II software and the result of Altera board will be integrate to the monitor using VGA Connector. The RTL schematic of decryption module is given in Figure 15 below. Which gives an abstract model of the decryption design in terms of flow of signals between hardware registers and about the logical operations between them.

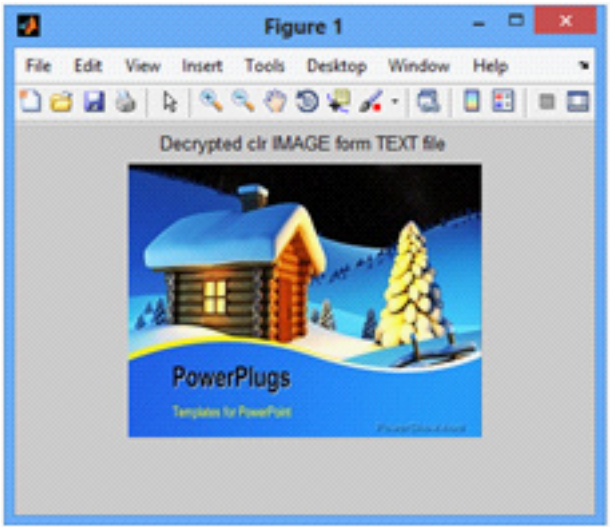

Figure 13. Decrypted image from MATLAB.

Figure 14. Decrypted image using VGA Connector.

4.2 Analysis and Synthesis Reports from Altera Quartus II

Altera Quartus II is very useful design software produced by Altera. This software enables analysis and synthesis of HDL designs, design compilation, RTL diagrams examination, timing analysis, to check a design's reaction to different stimuli, and target device configuration. Both VHDL and Verilog can be implemented for hardware description. Analysis and synthesis reports obtained from the board for cryptography is given in the Table 1 below. Also the comparison of the results with DES and RSA algorithms for image cryptography which are obtained from literature survey is also listed in table.

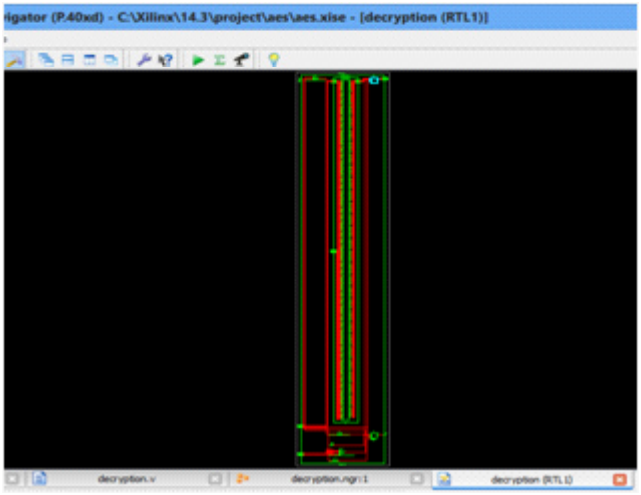

Figure 15. RTL schematic of Decryption module.

Table 1. Synthesis Utilization

	AES	RSA	DES
Number of Slice Registers	1638 out of 30064 5%	960 out of 3722 387%	16%
Number of Slice LUTs	1456 out of 15032 10%	1920 out of 4325 225%	28%
Number of bonded IOBs	21 out of 186 11%	1145 out of 108 1060%	22%
Maximum Frequency (fmax)	165.462MHz	37.578MHz	172.23MHz

5. Conclusion

This paper shows the successful implementation of hiding information using AES algorithm using FPGA to protect confidentiality of image information from an unapproved access. The successful implementation of cryptography using AES is a very challenging job, which takes large key size, components and time in RSA algorithm and requiring more utilization of components in DES even though the time consumption is less. It serves to explore the way to implement such an algorithm utilizing verilog code that is synthesized and simulated utilizing Model Sim, Quartus II software and Altera DE2 115 Board and displaying image using VGA Connector. The Maximum Frequency attained to from the design is 165.462 MHz.

6. Acknowledgement

I owe my heartfelt appreciation to God all-powerful for all the blessings he has showered on me amid the attempt. I take this chance to express my earnest appreciation to all the people who have been instrumented in bringing out this work to the right frame.

7. References

1. Soliman MI, Ghada AY. FPGA implementation and performance evaluation of a high throughput crypto coprocessor. Microelectronics Journal. 2013; 71(8):1075–84.
2. Saraf KR, Jagtap VP, Mishra AK. Text and image encryption decryption using advanced encryption standard. IJETTCS. 2014; 3(3):118–26.
3. Manoj B, Harihar MN. Image encryption and decryption using AES. IJEAT. 2012; 1(5):45–51.
4. Radadevi P, Kalpana P. Secure image encryption using AES. IJRET 2012; 1(2):115–7.
5. Stallings W. Evaluation criteria for AES. Cryptography and Network Security. 5th ed. Prentice Hall; 2010. Available from: http://mercury.webster.edu/aleshunas/COSC%205130/H-AES-Evaluation.pdf
6. Rohith S, Poornima, Mahesh C. FPGA implementation of 16 bit RSA cryptosystem for text message. International Journal of Computer Applications. 2014; 92(8):92–8.
7. Sumalatha Patil M, Mala LM. Design of high speed 128 bit AES algorithm for data encryption. International Journal of Current Engineering and Technology. 2013; 2(3):338–43.
8. Sharma IR, Gupta V. Comparative analysis of DES and S-DES encryption algorithm using verilog coding. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering. 2013; 1(9):489–73.
9. Karthikaikumar P, Rasheed S. Simulation of image encryption using AES algorithm. IJCA Special Issue on Computational Science - New Dimensions & Perspectives NCCSE; 2011; 166–72.
10. Singh S, Jain A. An enhanced text to image encryption technique using RGB substitution and AES. IJETT. 2013; 4(5):2108–12.
11. Padate R, Patel A. Encryption and decryption of text using AES algorithm. International Journal of Emerging Technology and Advanced Engineering. 2014; 4(5):54–9.
12. Kavitkar SG, Paikrao PL. FPGA based image feature extraction using xilinx system generator. IJCSIT. 2014; 5(3):3743–7.
13. Mohammed A, Rachid E, Laamari H. High level FPGA modeling for image processing algorithms using xilinx system generator. International Journal of Computer Science and Telecommunications. 2014; 5(6):67-72
14. Muralikrishna B, Deepika KG, Kanth BR, Vemana VGS. Image processing using IP core generator through FPGA. International Journal of Computer Applications. 2012 May; 46(23):712–20.