Encryption and decryption using FPGA

IOP Conf. Series: Materials Science and Engineering **263** (2017) 052030 doi:10.1088/1757-899X/263/5/052030

Encryption and decryption using FPGA

Nikhilesh Nayak, Akshay Chandak, Nisarg Shah and B Karthikeyan

School of Electronics Engineering, VIT University, Vellore, Tamil Nadu 632014, India

E-mail: bkarthikeyan@vit.ac.in

Abstract. In this paper, we are performing multiple cryptography methods on a set of data and comparing their outputs. Here AES algorithm and RSA algorithm are used. Using AES Algorithm an 8 bit input (plain text) gets encrypted using a cipher key and the result is displayed on tera term (serially). For simulation a 128 bit input is used and operated with a 128 bit cipher key to generate encrypted text. The reverse operations are then performed to get decrypted text. In RSA Algorithm file handling is used to input plain text. This text is then operated on to get the encrypted and decrypted data, which are then stored in a file. Finally the results of both the algorithms are compared.

1. Introduction

Cryptography is a technique used to encrypt and protect messages while transferring from sender to receiver using various algorithms. The sender and receiver use a special key known as cipher key which is then operated on along with the message to create an encrypted message which hides the plain text and then sent to the receiver. The receiver then uses his decipher key to decrypt and separate the message from the encrypted text. In this paper we are using two different algorithms namely RSA and AES [1].

In RSA(Rivest – Shamir – Adleman) Algorithm, we are using file handling to take plain text from a file, operate on it and generate the encrypted text, using a public key. This data is then sent to the receiver who uses a private key, derived from the public key, to decrypt and get back the plain text [2]. This output is then stored in another file. For the key generation we go through the following process:

- Select two random prime numbers : a, b.
- Calculate their modulus n

$$n = a*b. \quad (1)$$

- Calculate ø

$$ø(n)=(a-1)(b-1) \quad (2)$$

- Select a random and prime encryption key k: for [1< k < ø(n)] ,

$$GCD\ (\ n, ø(n)\ ) = 1 \quad (3)$$

- Calculate decryption key d:

$$(\ d*k\ )\ mod\ ø(n) = 1 \quad (4)$$

- Find their public key for encryption:

$$Pk=\{ k, n \} \quad (5)$$

- Find their private key for decryption:

$$Pd=\{ d, n \} \quad (6)$$

For encryption of a Message Q using public encryption key Pk={k, n} gives :

$$Crypt = Q^e \bmod n \quad (7)$$

where 0≤Q<n.

For decryption of Cipher text using the private decryption key Pd={ d, n } gives :

$$Message = Crypt^d \bmod n \quad (8)$$

Figure 1. Block Diagram for RSA methodology.

The basic steps included in the AES algorithm are:

- Plain Text: The message or data given as the input.

- Plain Key: Key length should be same as the input data length, which will be same for encryption and decryption.

- Shift Row: Here we will store the result obtain from xor operation of plain key and plain text in form of state matrix. Then we will keep the first row as it and move the remaining row to the left by some offset.

- Mix Column: Here the four bytes are taken as the input abdothe four byte as the output and this in addition with the shift rows will give the cipher text [3].

Figure 2. Block Diagram for AES methodology.

2. Components Required

Nexys 4 Development Board :
It‟s a FPGA Development board used to implement VHDL/Verilog codes [4]. It has the following specifications :

- Artix-7 FPGA Board by Xilinx
- Four 6-input LUTs & 8 flip-flops in each of the 15,850 logic slices
- 4 Kbytes RAM (fast block)
- Clock management tiles : 6 (with PLL)
- More than 450 MHZ of internal clock speed
- On-chip ADC (analog to digital converter)
- 126 Mbytes DDR2
- Serial Flash
- User Switches : 16
- User LEDs : 16, Tri color LEDs : 2
- 7-Segment Displays (4 Digit) :
2 - VGA output (12 Bit)

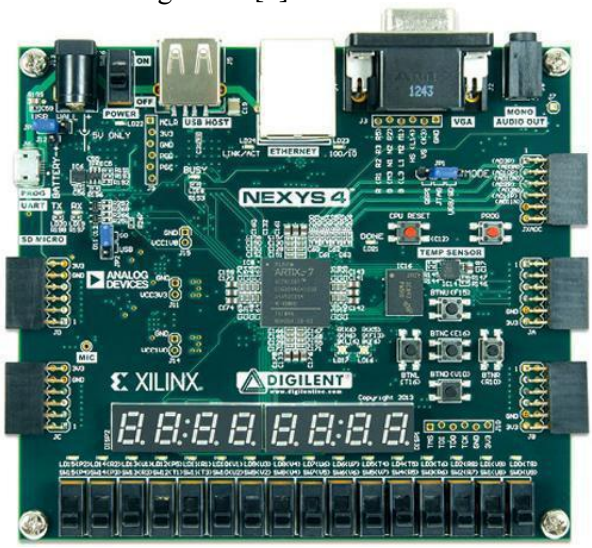

Figure 3.

Xilinx Vivado SDK :
It‟s a software development kit used to compile and execute VHDL/Verilog codes. After a code is compiled a RTL Schematic is generated. The code then can be simulated to check results in a virtual environment. The code is then synthesised and implemented on any FPGA board by generating a bit-stream and uploading it on the FPGA Board. To check result on the board a constraint file is edited to accommodate all inputs and outputs [5].

3. Methodology

For RSA Algorithm :

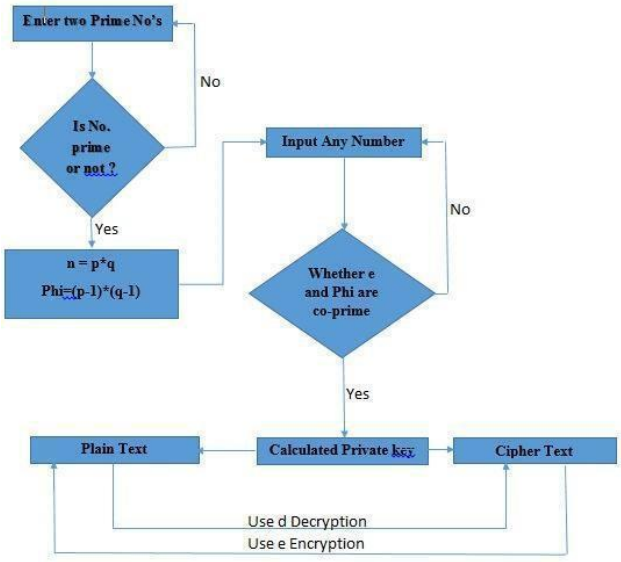

Figure 4.

For AES Algorithm :

We will send the data using 8 input pins of the fpga board, Hence the UART code will act as our top module, which will be port map with the cryptography code. Through UART we will send the hexadecimal values of the ascii code of entered character. Now this acts as the plain text in out code, which will be xor with the cipher key in our sub module. As a result of which the encrypted text will be reflected on the tera term. In the output results, we have shown both the results i.e. with and without encryption [6].

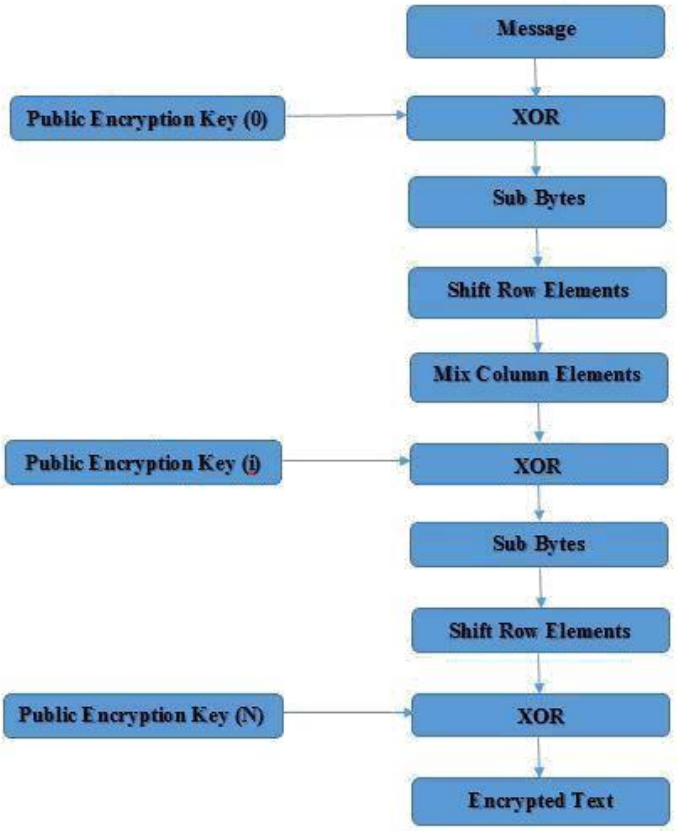

Figure 5.

4. Result and analysis

Simulation Results For encryption using AES algorithm:

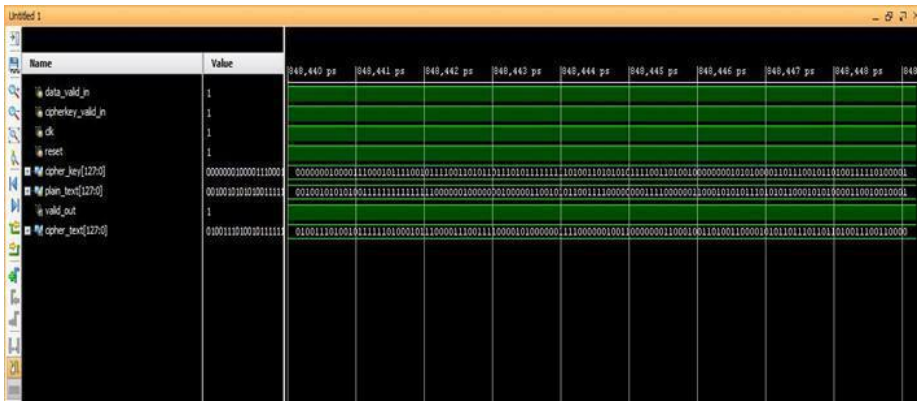

Figure 6.

The above results were obtained using Xilinx Vivado software. The figure shows the input plain text, input key and cipher text.

Hardware results of the AES implementation are shown below:
- Simple serial communication of a without encryption.

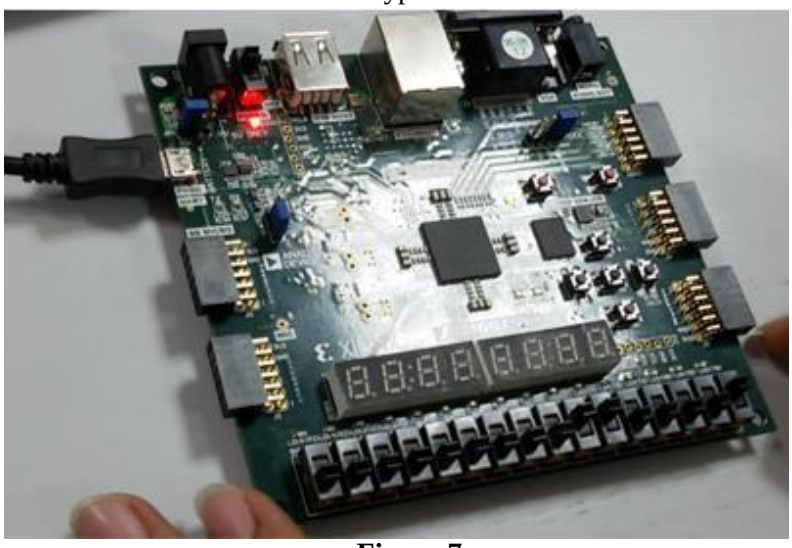

Figure 7.

In the above figure the first 8 switches act as the input. Here we are giving the hexadecimal value of character „a‟ that is 61. This will continuously print the output as „a‟ on the tera term. This is output without encryption. This is shown in the below figure,

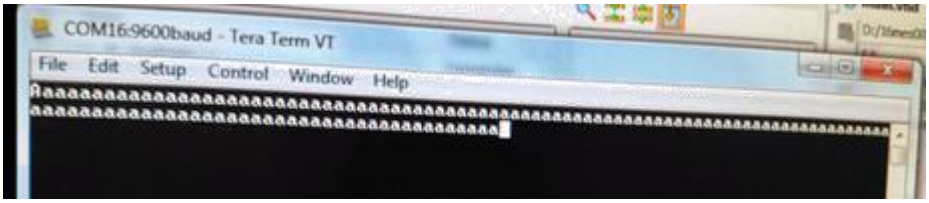

Figure 8.

Encrypted Version of the output:

Figure 9.

Now we are giving the same input to show the encryption that is hexadecimal value of „a". In the below figure it shows the encrypted data instead of the character „a".

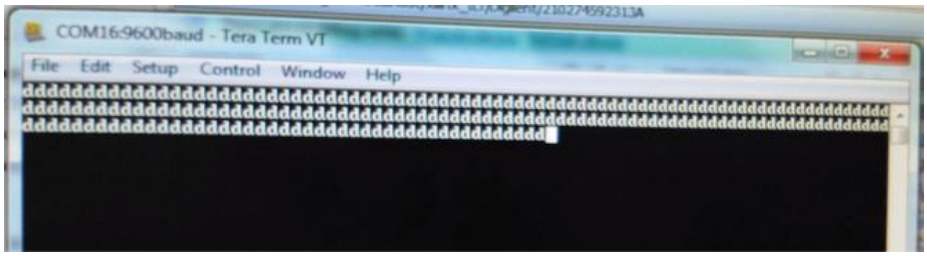

Figure 10.

RSA Algorithm Output :

1. File Handling to simulate input data file using VHDL.
2. Create Encrypted data file to be sent.
3. Decrypt received data and store it in a file.

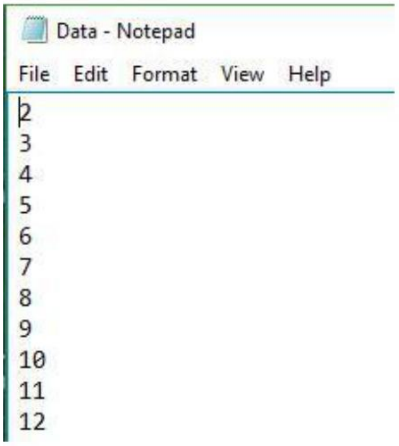

Figure 11. Input File

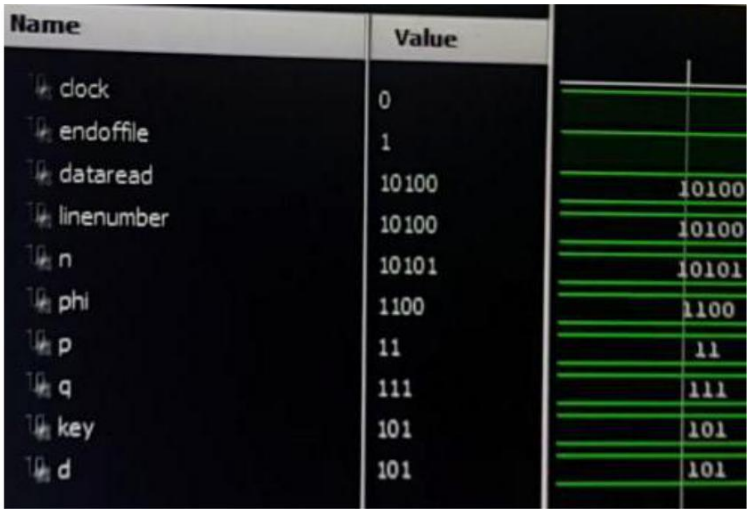

Figure 12. Simulation Result

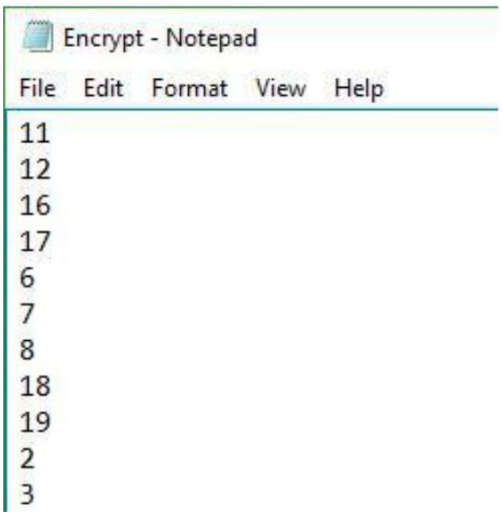

Figure 13. Encrypted Data File

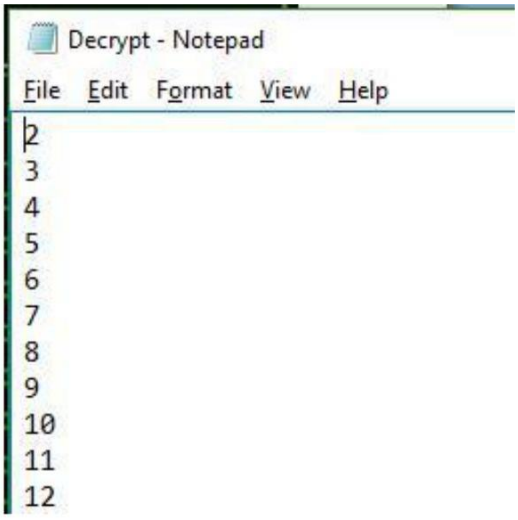

Figure 14. Decrypted Data File

5. Conclusion

Thus, using the above cryptography technique we conclude that, AES is a secure algorithm which uses a 128 bit key for encryption as well as decryption and is better than DES algorithm, which uses a 64 bit key, as larger key size avoids brute force attacks, while maintaining the same level of computational complexity. This algorithm was then implemented on FPGA, Nexys 4, Board and the output was displayed on Tera Term serially. RSA algorithm was then implemented using file handling on vhdl, using an input file. The encrypted and decrypted data were stored in two different files. Here two keys are used, one for encryption called the public key and another for decryption called the private key, which is derived from the public key. This prevents ambiguity as well as misinterpretation of key during exchange between sender and receiver. This also leads to a more secure form of cryptography.

References

[1] Amit Thobbi, Shriniwas Dhage, Pritesh Jadhav and Akshay Chandrachood 2015 Implementation of RSA Encryption Algorithm on FPGA *American Journal of Engineering Research*.

[2] Ari Shawkat Tahir Design and Implementation of RSA Algorithm using FPGA, *International Journal Of Computers & Technology, vol 14 No. 12 2015*

[3] Deepali P. Durgade, Mr. P. C. Bhaskar, Mr. B. P. Kulkarni and P. D. Patil, FPGA Based Authentication Unit Using Asymmetric Key Cryptography *International Journal of Innovative Science, Engineering & Technology*.

[4] Madhuri B. Shinde, 2015 FPGA Based Adaptive Asymmetric Cryptography Implementation *International Journal of Innovative Science, Engineering & Technology*.

[5] A. R. Landgeand A. H. Ansari 2013 RSA algorithm realization on FPGA *International Journal of Advanced Research in Computer Engineering &Technology*.

[6] Shashank Adimulam, 2012 FPGA Implementation of RSA Encryption and CRT based Decryption using Parallel Architecture *Journal of Innovation in Electronics and Communication.*