TELKOMNIKA, Vol. 13, No. 4, December 2015, pp. 1153~1161
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v13i4.1787

Image Encryption using Simple Algorithm on FPGA

Barlian Henryranu Prasetio[1], Eko Setiawan[2,] Adharul Muttaqin[3]
[1,2]Computer System and Robotics Lab, Faculty of Computer Science, University of Brawijaya, Jl. Veteran Malang, Ph. Fax: +62341-577911
[3]Computer System and Robotics Lab, Faculty of Engineering, University of Brawijaya, Jl. Veteran Malang, Ph. Fax: +62341-577911
e-mail: barlian@ub.ac.id[1], ekosetiawan@ub.ac.id[3], adharul@ub.ac.id[3]

Abstract

Data security becomes one of the things that need to be considered. The Sum of Product (SOP) Encryption is one of simple algorithm. The SOP encryption is not a public-key system to encrypt data. it is almost unbreakable through brute force method and vulnerable to attack because the encryption models have a fixed pattern. However, this drawback can be avoided through the compression process to remove the pattern file. SOP encryption algorithm can use a Boolean algebra combination functions such as AND gate and OR gate. Simple algorithm in hardware language VHSIC Hardware Description Language (VHDL) is on data bits level. The purpose of this research is implementation of image encryption algorithm to produce a quick image encryption system. From result test, the image processing time without encryption average 4.999ns/px and 13.51ns/px with encryption.

Keywords*: Encryption, Image, Simple Algorithm, VHDL*

1. Introduction

Embedded system is one that emphasizes the application of security in data communication [1]. Based on the Field Programmable Gate Array (FPGA) security system architecture, a security attack can be categorized into two in an attack on the hardware and software [2]. In general, the encryption process conducted by using software that is programmed in computer. Practice, only a few applications requiring throughput while flexible solutions and low cost encryption / decryption is needed to protect the data that makes sense, especially for embedded hardware applications [3]. The encryption image can be moved into computer by the software before passing it to another device. The use of computers as media encryption on small devices such as surveillance cameras is less appropriately. Some small devices such as FPGA have been potential to be applied to replace the computer as a medium for image encryption. The FPGA speed to data processing can compensate for the perceived performance of the computer. The data encryption system will be optimized with the to be implemented into the FPGA because it has advantages which include flexibility, development cost and costs low per-unit, high speed and has a good level of security [4].

Various kinds of encryption methods can be used to secure the data. Each method has advantages and disadvantages. The main problem is how to know and understand the workings of the encryption method algorithm. With Modular Multiplication Based (MMB) Encryption using 128-bit plaintext iterative algorithms which consists the linear steps and the four major non-linear substitutions parallel application can be reversed. This substitution is determined by a multiplication modulo 232-1 with a constant factor, which has a higher level of security when compared with the method of the International Data Encryption Algorithm (IDEA) that only uses multiplication modulo 216 + 1. MMB using 32-bit sub block text (x0, x1, x2 and x3) and 32-bit sub block key (k0, k1, k2, k3). This makes the algorithm is very suitable to implement on 32-bit processors [5].

Simple image encryption using XOR technique between the pixels of plain-image with a key encryption method is not secure against known-plaintext attack. However, the simple encryption method can have time for encryption and decryption process is relatively fast, it is due to the efficiency that occurs at a key plant so that it can be can be used in real-time systems such as digital phone lines.

Received March 24, 2015; Revised September 30, 2015; Accepted October 14, 2015

Known-plainttext attacks can guess the encryption key by comparing the number of plain-image (the original image) with the cipher-image (image encrypted) corresponding. This variants known-plaintext attack is called selective plaintext attack. In his research, Munir provides an analysis of the selective-plaintext attack against a proposed chaos-based image encryption algorithm. The algorithm combines the techniques of permutations (using Arnold Cat Map) and substitution techniques (using XOR operation). Based on a series of experiments that have been performed can be concluded that scrambles the pixels before the XOR operation can make the algorithm secure against selective-plaintext attack [6].

Almost all existing techniques require a lot of data encryption arithmetic and logical calculations, thus the embedded devices makes difficult to apply [7]. But with the simple algorithm would be easy. One form of Boolean algebra equations used almost throughout in digital device is Sum of Product (SOP). SOP has been potential to be selected as the encryption algorithm on a small device when time factor becomes an important. In addition, SOP algorithm can do the decryption process so it able to restore the original data.

Based on the condition, we research image encryption on Xilinx Spartan 3E FPGA module by applying the SOP algorithm. This paper is an initial step process of images encryption on small devices such as FPGA. The purpose of this paper is that it can be a reference for future work such as making the equipment that originally did not have a security network by adding this device has a secure.

Operators Exclusive-OR (XOR) is simple in principle the same as Vigenere cipher key using pieces are repeated periodically throughout the plain-text [8]. Vigenere table consists of pairs each letter plain-text, key and cipher-text results. In principle, every bit plain-text XOR with each bit of the key to the cipher-text produced bits. Commercial programs are DOS-based or Macintosh using a simple XOR algorithm [9]. SOP encryption algorithms can be prepared using the basic Boolean XOR operator.

SOP Encryption is not a public-key system to encrypt data. Encryption can be solved through brute force method. It is vulnerable to attack because the encryption models have been fixed pattern. However, this drawback can be avoided through the compression process to remove the pattern file. SOP encryption algorithm can used a combination of Boolean algebra functions such as AND and OR. In addition, the designers of computer systems are using advantages of Xilinx Spartan-3E FPGA. The Xilinx FPGA families carry microblaze processors [10], type of generation that can be applied to a variety of user applications, it has adjustment interface and data standards, and it can distinguish functionality with minimal design time. It can be allows a low cost image encryption for embedded systems while still providing a good trade-off between performance and hardware resources [11]. Author emphasizes the Sum of Product (SOP) block arithmetic algorithms to collection of cascaded blocks performing unit operations [12-15].

2. Research Method

The overall system design includes an encryption algorithm conversion into hardware architecture, software installation support, configuration, and design of reliability system. The overall process of encryption and decryption is showed in Figure 1.

SOP algorithm works by generating a random key. The performance comparison of cipher text shows that over the normal text, cipher text is very difficult, and time consuming to crack [16, 17].

In the FPGA module, one color pixel from plain-image is represented in 3 bits data. An image consists of 3 bits of data that make up the overall pixel color plain-image. If 3 bits of data in a single pixel is represented by A_{mn} variable, with m indicating the number of rows and n is the number of columns of pixels of an image, so a plain-image can be described in matrix form as described in Figure 2.

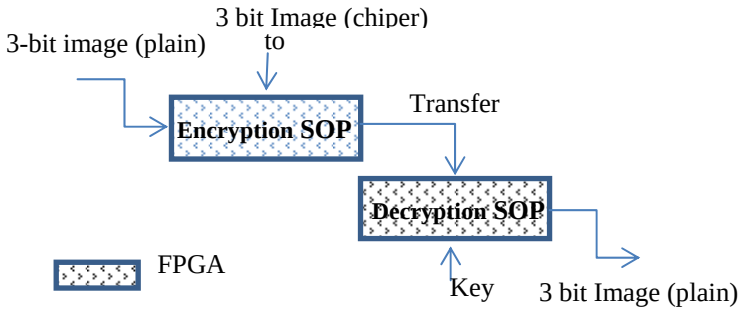

Figure 1. The illustration of plain image

A_{00}	A_{01}	A_{02}	A_{03}
A_{10}	A_{11}	A_{12}	A_{13}
A_{20}	A_{21}	A_{22}	A_{23}
A_{30}	A_{31}	A_{32}	A_{33}

Figure 2. The matrix of plain image

The general form of SOP is given by :

$$xy + x\bar{y} + \bar{x}y \quad (1)$$

x and y are variables that make up a function. x' is a negation form of x. Form of negation is true of all the variables that make the equation.The implementation of SOP are easy and stand in level bit, so make the encryption process faster.

Equation of encryption and decryption processes are described as follow :

Ci = E(Pi,Ki)
Pi = D(Ci,Ki)

where: Ci : Chipper
Pi : Plain
Ki : Key

In the encryption process, a cipher of data (C) can be determined by processing in the data plain (P) and encryption key (K) by using the SOP. The encryption process is performed on each pixel composing the plain-image. Cipher data from certain columns and rows of pixels obtained through SOP between plain pixel data in the same position with the encryption key. Illustration SOP operation on each pixel is described by the following matrix as follows:

$$C_{mn} = A_{mn}\bar{K} + \overline{A_{mn}}K_i \quad (2)$$

The results of pixels cipher will be re-constructed accordantly with the position of the rows and columns of pixels plain. Illustration of the placement of each pixel cipher in cipher-image is described in Figure 3.

C_{00}	C_{01}	C_{02}	C_{03}
C_{10}	C_{11}	C_{12}	C_{13}
C_{20}	C_{21}	C_{22}	C_{23}
C_{30}	C_{31}	C_{32}	C_{33}

Figure 3. The illustration of chipper

Pseudo code from encryption method by using SOP is described by:

```
for (i=0;i<m;i++){
    for (j=0;j<n;j++){
    if (k==8){
            k=0;}
    else    {
k=k+1;}
     C[i][j]=( A[i][j] AND NOT K[k] ) OR ( NOT A [i][j] AND K[k]);
}
}
```

3. Results and Discussion

In this section, testing was conducted by connecting the monitor to the VGA port LEDs contained in the FPGA module. The next step of the test is to enter and run into the program modules. Figure 4 is showed FPGA module condition that the monitor cable is connected to the LED.

Figure 4. Xilinx Spartan 3E FPGA module is connected to monitor

RTL Schematics Diagram of the system can be seen on figure 5.

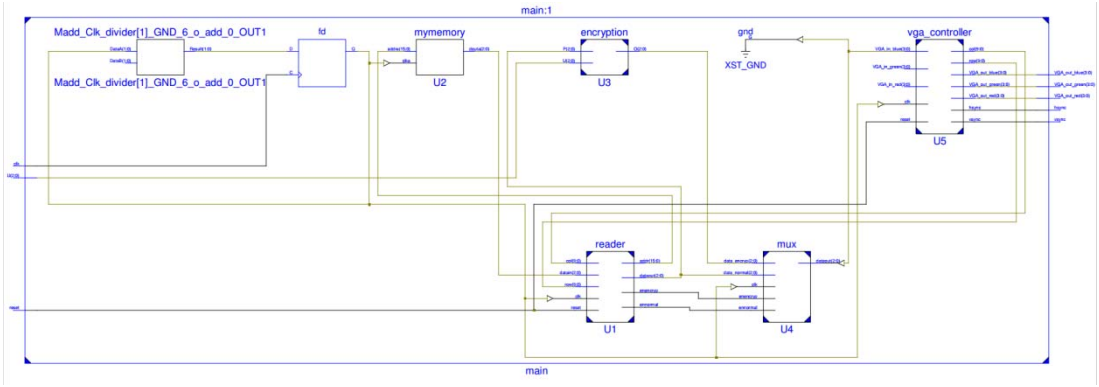

Figure 5. RTL Schematics Diagram

Figure 5 showed that the system implemented using RTL Schematics. The system consists of clock divider, buffers, memory, encryption, mux, reader and VGA controller. The RTL Schematics Description showed in table 1.

Table 1. The RTL Schematics Description

Main System

Inputs:
- clk – Clock signal, port A8 of the FPGA.
- reset – Push button 0 (BTN0)
- U – 3-bit user input password.
  Bit 0 – BTN1.
  Bit 1 – BTN2.
  Bit 2 – BTN3.

Outputs:
- hsync – Horizontal synchronism of the VGA. Port R6 of the FPGA.
- vsync – Vertical synchronism of the VGA. Port R7.
- VGA_out_red – Red signal to the FPGA.
  Bit 0: P16.
  Bit 1: P15.
  Bit 2: T7.
  Bit 3: R5.
- VGA_out_green – Green signal.
  Bit 0: N15.
  Bit 1: J16.
  Bit 2: K16.
  Bit 3: K15.
- VGA_out_blue – Blue signal.
  Bit 0: L15.
  Bit 1: M16.
  Bit 2: M15.
  Bit 3: N16.

Component: encryption

Inputs:
- U→ U of the system.
- P→signal dataread.

Output:
- O →signal dataencryp.

Component: mux

Inputs:
- clk → clk of the system.
- ennormal →signal enormal.
- enencryp →signal eencryp.
- data_normal →signal dataread.
- data_encryp →signal dataencryp.

Output:
- dataout →signal colors.

Component: vga_controller

Inputs:
- clk → clk of the system.
- reset → reset of the system.
- VGA_in_red(0) →signal colors(2). The bits 1 and 2 of VGA_in_red have to be connected to a constant '0'.
- VGA_in_green(0) → signal colors(1). The bits 1 and 2 of VGA_in_green have to be connected to a constant '0'.
- VGA_in_blue(0) → signal colors(0). The bits 1 and 2 of VGA_in_bluehave to be connected to a constant '0'.

Outputs:
- col →signal cols.
- row →signal rows.
- VGA_out_red → VGA_out_red of the system.
- VGA_out_green → VGA_out_green of the system.
- VGA_out_blue → VGA_out_blue of the system.
- hsync → hsync of the system.
- vsync → vsync of the system.

Mapping declaration of the vga_controller:

```
myvga: vga_controller
port map (  clk => clk,
  reset => reset,
   VGA_in_red(0) => colors(2),
   VGA_in_red(3 downto 1) => "000",
   VGA_in_green(0) => colors(1),
   VGA_in_green(3 downto 1) => "000",
   VGA_in_blue(0) => colors(0),
   VGA_in_blue(3 downto 1) => "000",
   col => cols,
row => rows,
   VGA_out_red => VGA_out_red,
   VGA_out_green => VGA_out_green,
   VGA_out_blue => VGA_out_blue,
hsync => hsync,
vsync => vsync
   );
```

Testing is done by applying encryption on several different pieces of plain image. Some images of Plain-image and cipher-image is showed in Figure 6.

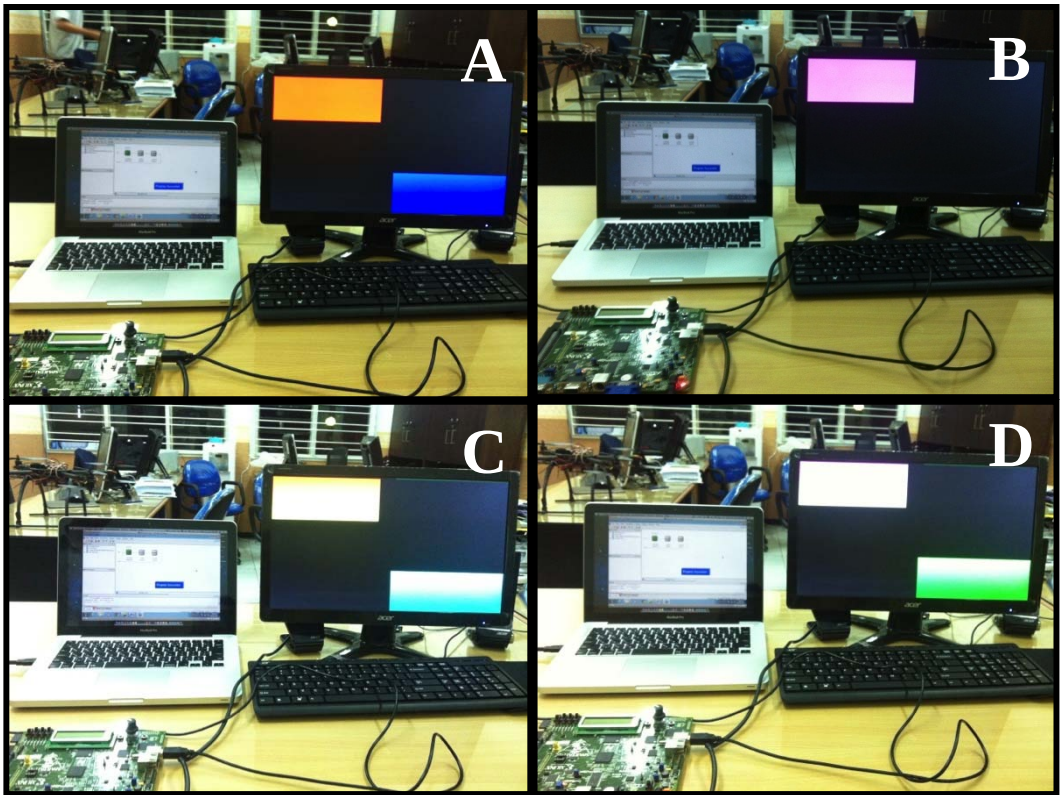

Figure 6. The image encryption process Testing

Spartan 3E FPGA module can be displaying every pixel of representation image in 3-bit. The combination with image displayed bits are showed in Table 2.

Table 2. The 3-bit data and color FPGA combination [18]

VGA Red	VGA Green	VGA Blue	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

Key encryption is used on the test is '101 '. In first test shown in Figure 6A, the plain-image that is used is red with data representation '100 ' on all pixels. SOP encryption algorithm will calculate every bit plain-image and key by using the following equation to get the cipher-image.

$$P = 100$$
$$K = 101$$
$$C_2 = \overline{P_2}K_2 + P_2\overline{K_2} = 0.1 + 1.0 = 0$$
$$C_1 = \overline{P_1}K_1 + P_1\overline{K_1} = 1.0 + 0.1 = 0$$
$$C_0 = \overline{P_0}K_0 + P_0\overline{K_0} = 1.1 + 0.0 = 1$$
$$C = 001$$

Based on table 2, cipher-image with a bit value of "001" will show a blue color on the LED screen. This test explains that the FPGA module can encrypt various plain-image into cipher-image corresponding to the given key. Encrypted images have different color combinations with the original image.

Tests were also performed on some of the color combinations that can be generated by the module FPGA. In Figure 6B, plain-image used was '101 '. Same as the key used previously so that cipher-image as follows:

$$P = 101$$
$$K = 101$$
$$C_2 = \overline{P_2}K_2 + P_2\overline{K_2} = 0.1 + 1.0 = 0$$
$$C_1 = \overline{P_1}K_1 + P_1\overline{K_1} = 1.0 + 0.1 = 0$$
$$C_0 = \overline{P_0}K_0 + P_0\overline{K_0} = 0.1 + 1.0 = 0$$
$$C = 000$$

Cipher-generated image is a black accordance with table 2. In testing using data plain-image '110 'as shown in Figure 6C, will produce cipher-image the data below:

$$P = 110$$
$$K = 101$$
$$C_2 = \overline{P_2}K_2 + P_2\overline{K_2} = 0.1 + 1.0 = 0$$
$$C_1 = \overline{P_1}K_1 + P_1\overline{K_1} = 0.0 + 1.1 = 1$$
$$C_0 = \overline{P_0}K_0 + P_0\overline{K_0} = 1.1 + 0.0 = 1$$
$$C = 011$$

Cipher code-generated image is '011 ', represented by the cyan color in accordance with Table 2. In Figure 6D, testing is done by inserting the plain-image pixels '111 '. Cipher-image pixels are generated from the encryption process is obtained as follows:

$$P = 111$$
$$K = 101$$
$$C_2 = \overline{P_2}K_2 + P_2\overline{K_2} = 0.1 + 1.0 = 0$$
$$C_1 = \overline{P_1}K_1 + P_1\overline{K_1} = 0.0 + 1.1 = 1$$
$$C_0 = \overline{P_0}K_0 + P_0\overline{K_0} = 0.1 + 1.0 = 0$$
$$C = 010$$

Cipher-generated image is '010' which is represented by the color green. These tests indicate that the encryption method could have been implemented into hardware-level programming language and is able to produce different color codes with the data plain-image.

System testing is emphasized at the image execution time. The system was tested using several picture sizes and calculate time encryption. The image test results with and without encryption based on image size can be seen in Table 3.

Table 3. Image Processing Time Comparison between without and with encryption based on image size

Image size (px)	Without encryption (ns)	With encryption (ns)
1x1	5.001	13.514
10x1	50.003	135.150
10x5	250.035	675.700
10x10	500.002	1351.500
20x50	5000.006	13514.200
20x80	8000.001	21624.000
40x90	18000.024	48650.400
40x100	20000.864	54056.000
50x100	25001.359	67570.089
50x150	37500.997	101362.500
75x150	56251.000	152032.500
75x200	75000.056	202734.080
100x200	100000.845	270280.100
100x250	125000.126	337825.005
150x300	225000.221	608130.013

Table 3 shows that time encryption is very fast in nanoseconds. Enhanced image size of each 10-fold. Encryption each time the image size increased an average 13.5ns.

4. Conclusion

Based on the test results it is concluded that the encryption method in image encryption sop can be implemented into the FPGA module. Simple method can encrypt the original image (the plain-image) into the encrypted image (cipher-image) is different. The encryption implemention method into fPGA module is done with the first set of input output modules according to the hardware configuration used. The ssuccess implementation is the starting point of the application of encryption images on small devices.

Xilinx Spartan-3E FPGA module datasheet explained that it can only handle a maximum of 3 bits of image data [18]. The use of FPGA devices is capable of handling larger bits images, will be able to encrypt images with lots of color choices.

From result test, the image processing time without encryption average 4.999ns and 13.51ns with encryption.

References

[1] Adib El S, Raissaoni N. AES Encryption Algorithm Hardware Hardware Implementation Architecture: Resource and Execution Time Optimization. *IJINS.* 2012; 1(2): 110-118. 2012.
[2] B Badrignans et al. *Security Trends for FPGAS.* Perancis: Springer. 2011.
[3] Abdelwahab, M. Encryption Implementation of Rock Chipher Based on FPGA*. Journal of Engineering. IOSR*. 2014; 04(01): 14-20. 2014.
[4] Dekate, K. Blowfish Encryption: A Comparative Analysis using VHDL*. Internationa Journal of Engineering and Advance Technology. IJEAT*. 2012; 1(5): June. 2012.
[5] Al-Hazaimeh O. *A* New Approach for Complex Encrypting and Decrypting Data*. International Journal of Computer Network & Communications. IJCNC*. 2013; 5(2).
[6] Munir R. *Analisis Serangan dengan Selective Plaintext pada Sebuah Algoritma Enkripsi Citra Berbasis Chaos*. *Seminar Nasional dan Expo Teknik Elektro*. 2012; 24.
[7] Hou F et al. *Novel Physically-Embedded Data Encryption for Embedded Device.* International conference on Trust, Security and Privacy in Computing and Communications. *12th IEEE International Conference.* 2013.
[8] Yumnam Kirani. Generalization of Vinegere cipher. *ARPN Journal of Engineering and Apllied Science*. 2012; 7(1): 39-44.
[9] Juaneni. *Enkripsi*. Diktat Kuliah Universitas Widyatam. Bandung. 2012.
[10] Vincke B et al. Real time Simultaneous Localization and Mapping: Towards low-cost Multiprocessor Embedded Systems. *Journal on Embedded System. EURASIP*. 2012; 5(): 2012.
[11] Marek Wójcikowski et al. FPGA-Based Real-time Implementation of Detection Algorithm for Automatic Traffic Surveillance Sensor Network*. Journal of Signal Processing System*. 2012; 68(1): 1-18.
[12] Uma R et al. Synthesis Optimization for State Machine Design in FPGAs. *International Journal of VLSI Design & Communication System. VLSICS*. 2012; 3(6): 79-89.

[13] Qasim M. FPGA Design and Implementation of Matrix Multiplier Architecture for Image and Signal Processing Application*. International Journal of Computer Science and Network Security. IJCSNS*. 2010; 10(2).
[14] M Nagaraju et al. High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics*. International Journal of Engineering Research and Applications. IJERA*. 2013; 3(1): 1079-1084.
[15] Thomos C. FPGA-based Architecture and Implementation techniques of a low-complexity hybrid RAKE receiver for a DS-UWB Communication System. *Telecommunication System. Springer Science*. 2013; 52(4): 2083-2099.
[16] Avireddy S. *Random 4: An Application Specific Randomized Encryption Algorithm to prevent SQL injection.* WAran research foundation (WARFT). www.warftindia.org/Publications/random4.pdf.
[17] http://lastbit.com/rm_bruteforce.asp
[18] Xilinx. *Spartan-3E FPGA Family Datasheet.* USA. 2012