J. Comp. Int. Sci. (2019) 10(1):-11-20
http://epacis.net/jcis/PDF_JCIS/JCIS-0155.pdf
jcis@epacis.net

PAN-AMERICAN ASSOCIACION OF COMPUTATIONAL INTERDISCIPLINARY SCIENCES

A compact 128/192/256-bits symmetric cryptography hardware module

Otávio de Souza Martins Gomes[a 1], Robson Luiz Moreno [b 2]

[a]Instituto Federal de Minas Gerais, Formiga, MG, Brazil
[b]Universidade Federal de Itajubá, MG, Brazil

Received on Nov, 2018 / Accepted on Jan , 2019

Abstract

This article describes the implementation of Twofish - one of the Advanced Encryption Standard (AES) finalists, in Field Programmable Gate Array - FPGA. The core was implemented in Altera Quartus Cyclone board. The code is totally portable and can be used in any FPGA. The algorithm was implemented for 128-bit word and 128, 192 and 256-bit keys. The main goal of this work was the implementation of an efficient, compact and modular Twofish hardware module that can find a wide range of applications as an alternative from AES-Rijndael.

Keywords: Cryptography, Twofish, AES, FPGA, Communication.

I. Introduction

In 1997, the NIST (National Institute of Standards and Technology) released a contest to choose a new symmetric cryptograph algorithm that would be called Advanced Encryption Standard (AES) to be used to protect confidential data in the USA. The algorithm should meet few requirements such as copyright free, faster than the 3DES, cryptograph of 128 bit blocks using 128, 192 and 256 bit keys, possibility of hardware and software implementation, among others. The three finalists were Rijndael, Twofish and Serpent. Each one has its own design features to achieve the requirements. In 2000, after analysis by cryptography experts, Rijndael was the winner[1][2]. With the advance of technology and the increment of hardware utilization for cryptography interfaces, it is interesting to analyze Twofish implementation in comparison with the winner, Rijndael. Some researchers are re-evaluating the characteristics of the finalists to achieve the best algorithm for their applications [3][4][5][6]. This work was implemented in FPGA due to its flexibility and reconfiguration capability [7]. A reconfigurable device is very convenient for a cryptography algorithm since it allows cheap and quick alterations. Section II provides a brief introduction of Twofish

[1]otavio.gomes@ifmg.edu.br
[2]moreno@unifei.edu.br

and its processing phases. Section III describes the chosen FPGA and the circuit implementation. Section IV compares the results of this work with others presented in the literature. Section V presents the conclusions of this work and finally Section VI shows the authors expectations and proposals for future work.

II. Twofish Algorithm

In order to better understand the Twofish structure, it is based on Blowfish algorithm. The Twofish cipher is a 128-bit block cipher and supports key sizes of 128, 192 and 256-bits. It is used a Feistel function for 16 times (16 rounds to all key sizes) to provide its strength, that consists of a fundamental block called F function and is key dependent. Figure 1 shows the structure of the Feistel function in this algorithm.

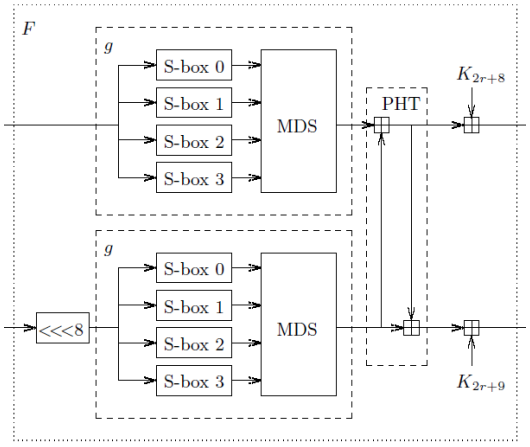

Figure 1 - Feistel Network[7].

This algorithm receives an input block of 128-bits that is divided into 4 blocks of 32-bits. Like Serpent algorithm, a pair of pre-processing and post-processing blocks are used, called input and output whitening. After the input processing, the lower part will be used as the input of Feistel function, which consists of 4 sequential operations, Primary rotation, g function, Pseudo-Hadamard Transforms (PHT), the combination of the key and the final rotation [8].

The encryption and decryption circuits are almost the same except that the internal keys are applied in a reverse order when decrypting. This structure makes the encryption and decryption units look very similar. Internally of F function, there are g functions that consist of key dependent S-boxes. Each S-box contains permutations q_0 and q_1, that are fixed permutations

on 8-bit values. Each S-box has its own way to calculate the q functions. Between each calculation of q function it is necessary to execute a XOR operation with the correspondent S Key.

The results are multiplied by 4X4 MDS (Maximum Distance Separable) using Galois Field- GF (2^8). The primitive polynomial is $x^8+x^6+x^5+x^3+1$ [7]. The Pseudo-Hadamard transform is a simple addition with module 2^{32}. The last operation is an addition that is performed with the output from PHT and the round keys.

A. g Function

The g function receives a 32-bit block that is divided into 4 blocks of 8-bit that is inserted into independent S-boxes. Each S-box is a bijective function which takes 8 bits as input and returns an 8-bit output and use keys S_0 and S_1 to process the XOR operations between the q functions. Each of the S-boxes contains permutation structure q_0 and q_1, which works as follows:

$a_0, b_0 = [x/16], [x \bmod 16]$
$a_1 = a_0 \oplus b_0$
$b_1 = a_0 \oplus \mathrm{ROR4}(b_0; 1) \oplus 8a_0 \bmod 16$
$a_2, b_2 = t_0[a_1], t_1[b_1]$
$a_3 = a_2 \oplus b_2$
$b_3 = a_2 \oplus \mathrm{ROR4}(b_2; 1) \oplus 8a_2 \bmod 16$
$a_4, b_4 = t_2[a_3], t_3[b_3]$
$y = 16\ \ b_4 + a_4$

Where: ROR indicates rotation right of the first argument by amount of the second argument.

B. Key Schedule

Each round of Feistel network needs two sub-keys. The key schedule is the function responsible for expand the initial key in 40 sub-keys. The first 8 keys, generated by the key schedule, are used for input and output whitening (K_0-K_7), other keys are used during the 16 rounds (K_8-K_{39}).

The key schedule for keys K_0-K_{39} uses the same primitives as the round function. The S-Boxes keys are generated by Reed Solomon (RS) mapping function. The keys S_0, S_1, S_2 and S_3 are obtained by matrix multiplication over Galois Field- GF (2^8) using the 128-bit user key. The primitive polynomial is $x^8+ x^6+ x^3+ x^2 + 1$. The RS matrix is a 4X8 matrix derived from the RS code [7].

The keys could be pre-calculated or can be calculated at the same time of the rounds (called on-the-fly). Twofish algorithm provides less noise sensitivity due to the lower key sensitivity and plaintext sensitivity [6].

III. Implementation

Twofish algorithm could be implemented in modules, where all of them could be independently tested and characterized, and therefore they can be used in any combination, according to its purpose. In order to conduct tests on all blocks, it was assembled a 128/192/256 bits encryption/decryption Twofish set in an Altera Cyclone IV FPGA. The test results are presented in Section IV.

The description implemented on FPGA was made using the VHDL, with the goal to fit any of the FPGAs. Those are the basic sub-blocks implemented in this work: Adder32-bit, S-Keys, PHT, MDS, qFunction and multipliers. They were developed and tested according to the algorithm standard[7].

The block of multipliers was created to be used on MDS and Reed-Solomon calculations. The adder 32-bit block was used on Pseudo-Hadamard Transform (PHT) and in the sub-Keys addition. The qFunction block was developed to be used on S-Boxes. The S-Box block processes the q function and implements one read-only memory to store the tables of q_0 and q_1 [7] [2].

The S-Keys block receive one key of 64-bit and return a 32-bit key. It must be used two times on 128-bit key, three times on 192-bit key and four times on 256-bit key. From these basic blocks, greater functions were developed. Figure 2 shows how the functions g and Pseudo-Hadamard Transform are executed.

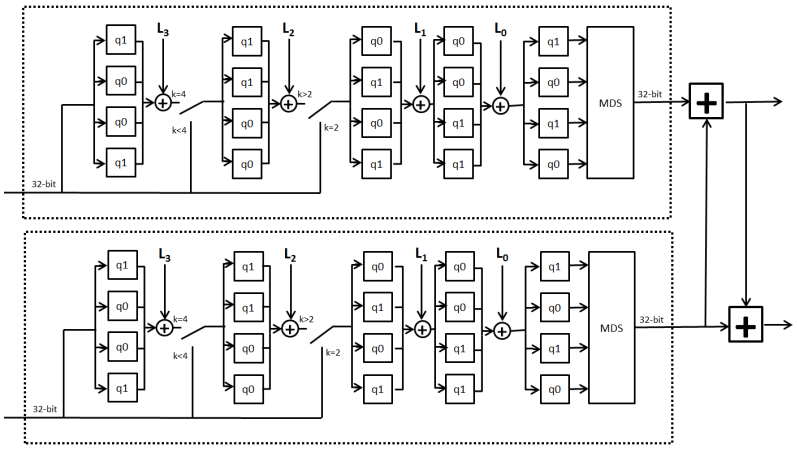

Figure 2 - g Functions.

The S-Box top level view could be represented by Figure 3. A state machine controls this compact module inserting the correct words and keys for each state, according the signal in_selSecLvl, that controls the security level of cryptography (128, 192 or 256). In this implementation, the change of key-size will not change the size of this module.

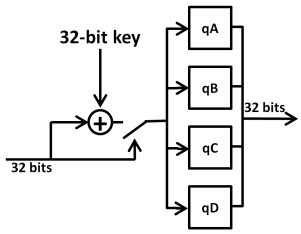

Figure 3 - S-Box compact module.

The Key Expansion block, called h function, is developed with the same basic blocks, but there are some transformations that make the blocks g and h different [7].

The hardware description code shows the entity declaration of block Feistel, that imports the component blocks: PHT, g function and adder32. The block that implements the Feistel function has some input signals to control its utilization. The signals *clk* and *rst* are common to all top blocks. The signal *in_selSecLvl* is used to select the security level of cryptography (128, 192 or 256) and is loaded with the crypto key and the cipher or plain text.

```
entity feistel is
port (
     clk : IN std_logic;
     rst : IN std_logic;
     in_load : IN std_logic;
     in_data_LS  : IN std_logic_vector(31 downto 0);
     in_data_MS : IN std_logic_vector(31 downto 0);
     in_key_2r8  : IN std_logic_vector(31 downto 0);
     in_key_2r9  : IN std_logic_vector(31 downto 0);
     in_load_key : IN std_logic;
     in_L0  : IN std_logic_vector(31 downto 0);
     in_L1  : IN std_logic_vector(31 downto 0);
     in_L2   : IN std_logic_vector(31 downto 0);
     in_L3  : IN std_logic_vector(31 downto 0);
     in_selSecLvl : IN std_logic_vector (1 downto 0);
     out_data_LS : OUT std_logic_vector(31 downto 0);
     out_data_MS : OUT std_logic_vector(31 downto 0);
     out_ready : OUT std_logic
);
end feistel;
```

In case of pre-computing keys, the signal *in_load_key* could be ignored, but when the keys are being calculated on-the-fly, this signal shows that the round keys are ready to be used. When all the data is ready to be read, the data signals are put in their buses (*out_data_LS and out_data_MS*) and the signal *out_ready* is on high level.

Figure 4 shows the inputs and outputs signals of the Twofish hardware developed.

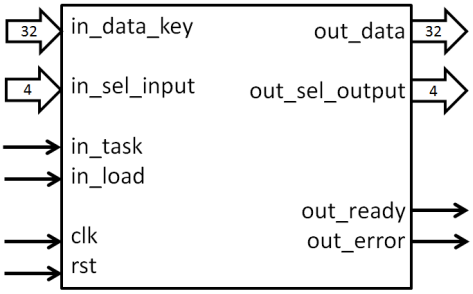

Figure 4 - Twofish top level block.

The signals *in_sel_input* and *out_sel_output* offer to the user the control over the hardware and the status about the work in progress. Table 1 shows the status signals of top level.

Table 1: Status signals

in_sel_output	**Action**
0000	LSWord of 128-bit Data Ready
0001	2nd LSWord of 128-bit Data Ready
0010	3rd LSWord of 128-bit Data Ready
0011	MSWord of 128-bit Data Ready
0100	Waiting all input values
0101	All input values OK
0110	Busy / working
0111 - 1111	To be defined

The control of the top level could be described by finite state machine. The hardware module will start the ciphering or deciphering process with the action of in_load signal. When signal *in_load* is in high level the inputs are loaded according the user commands. This process checks the inputs of the user and will finish when the user command the start of ciphering or deciphering. In case of a value missing, the module will inform that it is Waiting all input values, through the signal *out_sel_output* = 0100. During the process of the text, the module presents status *out_sel_output* = 0110.

When the work is done, the signal *out_ready* will be in high level and the user will use the input commands to read the results. The results will be available in the module interface until a reset process or a new load process.

IV. Tests and hardware verification

The hardware was tested and the functions were verified according to the standard and test vectors of AES documentation design, available in [7] and [8]. All the results were obtained according to the benchmarks.

The FPGA board used in tests was Altera DE2-115 (Cyclone IV E) with the tools Altera Quartus II 64-Bit Version 13.1.0 Build 162 and Altera ModelSim version 10.1d. Table 2 shows some information about the synthesis. The implementation ran with the frequency of 250MHz. Figure 5 shows the waveform of g Functions (presented on Figure 2).

Table 2: Synthesis results

Blocks	Logic elements	Nr of clock cycles	I/O pins
PHT	155	5	132
q Function	161	6	21
MDS	185	3	67
S-Boxes	1,947	30	200
g Function	3,289	35	73
Key Expand	3,378	40	345
Feistel Function	3,911	42	330

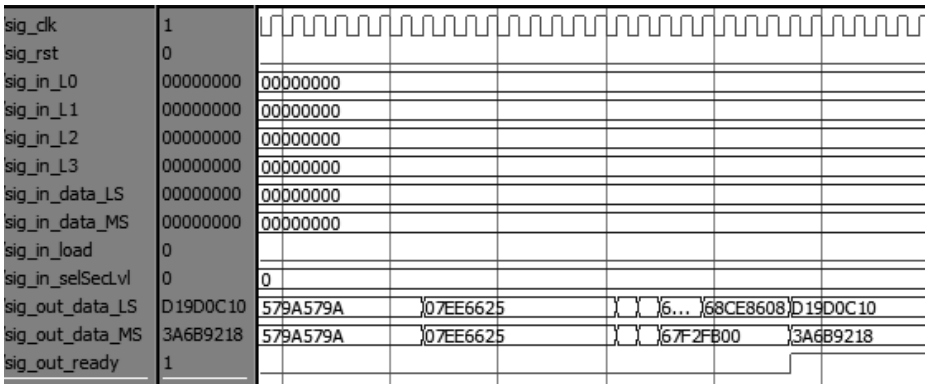

Figure 5 - Waveform of g Functions.

V. Results comparison

According to the results in AES competition, Twofish is a strong algorithm in hardware implementation [10][11]. It could be seen through the number of works that is studying and using the AES finalists after some advances on technology [3][4][5][6]. The authors compared some AES implementations with this implementation of Twofish.

The AES module related on [9] conduct performance comparison with other works and was observed that the first AES work [9] was at least 50% faster that the fastest circuit reported. This result was related to the architecture of the hardware developed and the same environment related in others works was used to make the correct comparison. The environment of tests was the Altera Cyclone IV E (EP4CE115F29C7). The AES implementation occupied 12.8K Logic Elements of the board, while Twofish implementation occupies 10.6K Logic Elements. The complete version of Twofish (128/192/256) occupies less logic elements than the 128-bit version of AES. Twofish occupies 83% of total Logic Elements in comparison with

AES implementation.

The AES algorithm has different execution times for encryption and decryption, while Twofish has the same time for both. In the AES, the decryption process needs to complete the expansion key before to start the decrypt process, increasing the decrypt latency [12]. Twofish has the same structure for encryption and decryption [7]. Twofish, using a 128 bits block size, the dictionary attack will require 2^{128} different plaintexts to allow the attack to decrypt arbitrary message under an unknown key [13].

According X. Lai [14], the Higher-Order Differential Cryptanalysis attack work best, only a few rounds, against algorithms with simple algebraic function and poor short term diffusion. According [7], not any higher order differential attack reported is successful against more than 6 round of the target cipher. In this case, using Twofish, it is not found any higher-order differentials can be exploited in the cryptanalysis.

A compact S-Box module controlled by a state machine reduces the reconfigurable logic occupation. Another advantage of Twofish algorithm is the possibility to execute the expansion key on-the-fly increasing the noise signal against side-channel attacks. Even with the expansion on-the-fly, the Twofish 128/192/256-bit is small (logic elements) than AES-128 bit.

VI. Conclusions

This article presented a compact Twofish 128/192/256 cryptography hardware module that can have many applications. The circuit implementation is compact and can be customized to a wide range of applications, with greater security level when compared with AES-128.

The authors intend to use this work as part of larger projects, including smart metering in power systems and security interface in data communications.

Acknowledgments.

The authors acknowledge CAPES, CNPq and FAPEMIG for their financial support.

References

[1] FIPS FIPS-197, Federal Information Processing Standards Publication FIPS-197, Advanced Encryption Standard - AES, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 1999.

[2] DAEMEN, J. AND RIJMEN, V., The design of Rijndael: AES The Advanced Encryption Standard. Springer-Verlag, 2002.

[3] RIZVI, S.A.M, HUSSAIN, S. Z., WADHWA, N. Performance Analysis of AES and TwoFish Encryption Schemes. 2011 International Conference on Communication Systems and Network Technologies.

[4] VERMA, H. K., SINGH, R. K. Performance Analysis of RC6, Twofish and Rijndael Block Cipher Algorithms. International Journal of Computer Applications (0975 8887), Volume 42 No.16, March 2012.

[5] MUSHTAQUE, A.; DHIMAN, H.; HUSSAIN, S.; MAHESHWARI, S. Evaluation of DES, TDES, AES, Blowfish and Twofish Encryption Algorithm: Based on Space Complexity. International Journal of Engineering Research and Technology - IJERT, Vol. 3, Issue 4, April 2014.

[6] NAEEMABADI, M.; ORDOUBADI, B. S.; DEHNAVI, A. M.; BAHAADINBEIGY, K. Comparison of Serpent, Twofish and Rijndael encryption algorithms in tele-ophthalmology system. Advances in Natural and Applied Sciences, Pages: 137-149, April-2015.

[7] SCHNEIER , B.; KELSEY, J.; WHITING, D.; WAGNER, D.; HALL, C.; FERGUSON, N. Twofish: A 128-bit Block Cipher; 15 June, 1998. Technical Report available at http://www.counterpane.com/twofish.html

[8] SCHNEIER B., Twofish- A New Block Cipher, Technical Report available at http://www.schneier.com/twofish.html

[9] GOMES, O. S. M.; PIMENTA, T. C.; MORENO, R. L., "A highly efficient FPGA implementation", 2nd Latin America Symposium on Circuits and Systems (LASCAS-2011), February 2011.

[10] PESCATORE, J. Using Hardware-Enabled Trusted Crypto to Thwart Advanced Threats. A SANS Whitepaper Written, September/2015

[11] SCHNEIER, B.; WHITING, D. A Performance Comparison of the Five AES Finalists. 2000.

[12] DAEMEN, J. AND RIJMEN, V., The design of Rijndael: AES The Advanced Encryption Standard. Springer-Verlag, 2002.

[13] ANDERSON, R., BIHAM, E. and KNUDSEN, L., Serpent: A Proposal for the Advanced Encryption Standard", NIST AES Proposal. 1998.

[14] LAI, X. Higher Order Derivations and Differential Cryptanalysis, "Communications and Cryptography: Two Sides of One Tapestry, Kluwer Academic Publishers, pp. 227 233, 1994.