AES-512: 512-Bit Advanced Encryption Standard Algorithm Design and Evaluation

Abidalrahman Moh'd
Engineering Mathematics and Internetworking
Dalhousie University
Halifax, Canada
Abidalrahman.Mohd@dal.ca

Yaser Jararweh, Lo'ai Tawalbeh
Cryptographic Hardware and information Security lab
Computer Engineering Department
Jordan University of Science and Technology
Irbid, Jordan
{yijararweh, tawalbeh}@just.edu.jo

***Abstract*—This paper presents an FPGA architecture for a new version of the Advanced Encryption Standard (AES) algorithm. The efficient hardware that implements the algorithm is also proposed. The new algorithm (AES-512) uses input block size and key size of 512-bits which makes it more resistant to cryptanalysis with tolerated area increase. AES-512 will be suitable for applications with high security and throughput requirements and with less chip area constrains such as multimedia and satellite communication systems. An FPGA architectural for AES-512 was developed using VHDL, and synthesized using Virtix-6 and Virtex-7 chips. AES-512 show tremendous throughput increase of 230% when compared with the implementation of the original AES-128.**

Keywords; Advanced Encryption Standard; FPGA; Enhanced Security; Cryptography.

I. INTRODUCTION

Involving information security techniques to get secure communication systems is becoming very essential in today's applications. Almost we need security in our daily transactions especially the ones that involves very private date. The information security can be achieved via using cryptographic algorithms such as DES, Two fish and many others. The AES [9] is the Advanced Encryption Standard algorithm FIPS-197 [1] that been in use since 2001 since it provides high level of security and can be implemented easily.

Many hardware implementations for AES algorithm were previously proposed and evaluated for the basic standard key size i.e. 128, 192, and 256-bit, as in [7] [9] [17] [18] and [19]. On the other hand, a compact AES implementations were presented in [5] [6] and [12]. These various implementations for AES support the fact that different applications required different implementation for the same algorithm. Some applications has strict area requirements and a compact AES implementation will be very useful to provide security as in the some embedded systems cases. On the other side, some applications highly need the most level of security that can be obtained without caring about the area/time limitations [21].

As more security wanted for a certain system as more chip area needed, and this is due to complex algorithm flow or by increasing the algorithm parameters that include key size, and plaintext size. This paper presents a new variation of the AES algorithm (called AES-512 bits). The hardware architecture for the AES-512 algorithm is also presented. The goal of this research is to present the AES-512 to be used when higher levels of security and throughput are required without increasing the overall design area when compared with the original AES-128 bits. The new algorithm has similar structure to the original AES with larger plaintext size and key size (512-bit instead of 128-bit). Using inputs of 512-bit instead of 128 has impact on the whole algorithm structure, as it will be discussed in details later on. The procedure to generate the new 512-bit key will be presented as well.

The AES is a symmetric cipher algorithm with block size of 128-bit supports key sizes of 128, 192, and 256 bits with 10, 12, or 14 iteration rounds, respectively. Four major operations are performed during each round: byte substitution, shifting rows, mixing columns, and finally adding the round key. AES 128-bit key is considered secure compared to the other existing symmetric cipher algorithms. It is widely used in many applications where the security is a very important. The new AES algorithm provides even more security and double the throughput. More security comes from using larger key size, and more throughput comes from using four times larger block size than the block size used in the original AES. The only disadvantage of AES-512 is the need for more design area.

The proposed AES-512 algorithm has four main different byte-based transformations. The first transformation is the Byte Substitution which substitutes the values of 512 bits and this is achieved via using parallel S-boxes. The second transformation is Shifting Rows, that shifts the rows of the output from the previous step by an offset equals to the row number. The third transformation is Mixing Columns, where each column of the output from the previous step is multiplied by different value. The final transformation in the round is Adding the Round Key to the final result of this round.

The rest of the paper is organized as follows, related work is presented in section II. The overall architecture of the AES-

978-1-4577-2155-7/11/$26.00 ©2011 IEEE

512, and the detailed transformations are provided in Section III. The key expansion operation of the proposed algorithm is explained in Section IV. Section V shows the experimental synthesis results of the AES-512 algorithm and comparisons. The conclusion is presented in Section VI.

II. Related Work

There are several hardware implementations for AES that can be found in literature. These implementations were done for the original standard AES (not more than 256 bits key size), such as the work presented in [8] [9] [10]. Before choosing Rijindael to be the AES in November of 2001, many related implementations were proposed to how much the structure of the proposed candidates for the AES competition is suitable for hardware implementation [3] [7] [18] [19] [20]. They provide a multiple architecture options for AES finalist candidates with an implementation analysis for each architecture based on both area and speed optimization. Also, the authors in [7] and [8] provided a detailed comparison of the FPGA hardware performance of the AES candidates.

A Field Programmable Devices (FPD) implementation for Rijndael was presented in [4] with a comparison with Xilinx FPGA implementation. The authors claim a performance gain (speed) using FPD over other FPGA implementation. An optimized S-Box design was presented in [5] for an efficient compact hardware design. Another compact AES design was presented in [7]. The compact designs target the small embedded systems market with low memory constraints. On the other hand, a VLSI Implementation with 1.82-Gbits/sec proposed and evaluated in [10].

In [13], application based speed/area tradeoffs design was presented for the AES. A high speed and low power sub-pipelined structure proposed in [12] for compact AES design. An FPGA parallel implementation of the AES proposed in [13].

On the other hand, in [14] and [15] a 128, 192, and 256-bit key size ASIC implementation of AES with high-throughput cost-effective design were proposed.

III. AES-512 Architecture

The top level architecture of the AES-512 bits is shown in Figure 1. The plaintext and the key size are 512-bits each (organized in bytes). The AES-512 algorithm processes the data in 10 rounds.

The key and the input data are loaded when the *Loadkey* control signal is one and zero, respectively. The *Encrypt* signal starts the encryption process, while *reset* resets everything to zero. The resulting ciphertext is also 512-bits.

More details about each of the transformations used in the AES-512 are described in the coming subsections. Where the key expansion procedure is explained later since each round needs its own key generated according to this procedure.

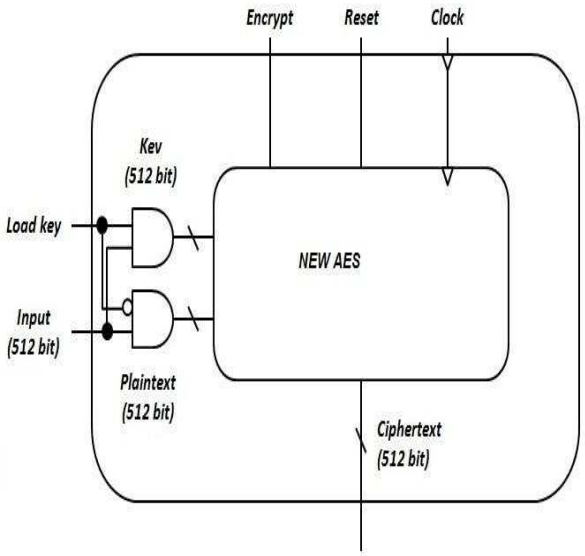

Figure 1. Top level of architecture AES-512

A. Byte Substitution

The 512-bits input plaintext are organized in array of 64-bytes and are substituted by values obtained from Substitution boxes. This is done (as in the original AES) to achieve more security according to diffusion-confusion Shannon's principles for cryptographic algorithms design.

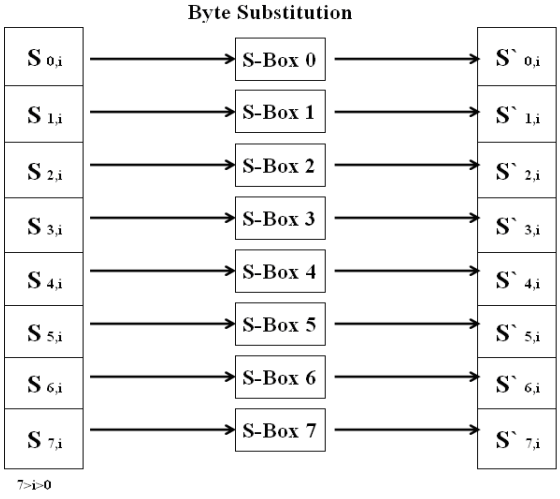

Figure 2. Byte substitution in AES-512

To overcome the overhead of the huge data size used (512-bits), the Substitution boxes are implemented as lookup tables, and accessed in parallel as shown in Figure 2.

B. Shift Row

After the original 512-bit data is substituted with values from the S-boxes, the rows of the resulting matrix are shifted in a process called *ShiftRow transformation*. What happened in this part is that the bytes in each row in the input data matrix will be rotated left. The number of left rotations is not the same in each row, and it can be determined by the row number. For example, row number zero is not shifted, the first row is shifted by one byte, and so on.

C. Mix Colomn

Now, and after the rows of the input data are rotated left by different offsets, an operation must be applied to the columns of the data matrix. The *MixColumn transformation* multiplies the columns of the data matrix by a pre-defined matrix.

The AES-512 and original AES process the data in bytes basis. Each byte is considered as polynomials over GF (2^8) with 8 terms. To explain how the *MixColomn* works, we have to explain the concept of polynomials over GF(2^n) in general and for GF (2^8) as example when n=8.

A binary extension field element Y (x) is a polynomial of degree less than n and greater than -1, (i.e. Y (x) ≠ 0), and has coefficients in GF(2).

The polynomial basis is one representation for the elements of GF(2^n). The addition in GF(2^n) corresponds to a polynomial addition, which is done as a bitwise logic exclusive OR operation between the two bit vectors being added. An irreducible field polynomial p(x) of degree *n* is used to reduce intermediate results in GF(2^n). In other words, the polynomials are reduced mod p(x) through long division operation to keep their degree less than *n.*

The *MixColomn* operation (shown in Figure 3) multiplies the columns in the data matrix with a fixed polynomial of a(x), given by:

$$a(x)=[02]x^7+[01]x^6+[03]x^5+[01]x^4+[01]x^3+[01]x^2+[01]x^1+[01]x^0$$

The multiplication result is taken (modulo p(x)= x^8 + 1) to keep the resulting polynomial with degree less than 8.

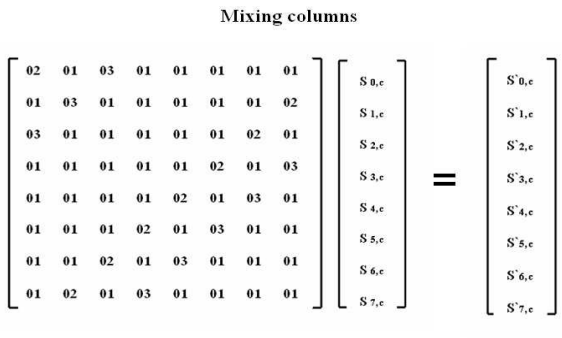

Figure 3. Mix column operation in AES-512

In the inverse of the Mix Column transformation, the input array is multiplied with the inverse of the polynomial a(x), denoted as $a^{-1}(x)$, which is given by:

$$a^{-1}(x)=[0E]x^7+[01]x^6+[09]x^5+[01]x^4+[0D]x^3+[01]x^2+[0B]x^1+[01]x^0$$

D. AddRound Key

To make the relationship between the key and the ciphertext more complicated and to satisfy the confusion principle, the *AddRoundKey* operation is performed.

This addition step takes the resulting data matrix from the previous step and performs on it a bitwise XOR operation with the sub key of that specific round (addition operation in GF(2^n)). We must mention that the round key is 512 bits that is arranged in a square matrix of eight columns where each column has 8 bytes.

IV. Key Expansion and Rounds

The 512-bit input key of the new AES-512 algorithm is used to generate ten sub-keys for each of the ten AES rounds. The round –keys expansion process involves arranging the original 512-bits input key into eight words of eight bytes each. After that, the round keys expansion is performed according to the following equations:

$$W(i)=W(i-8)\oplus W(i-1) \quad i \text{ isn't a multiple of } 8$$

$$W(i)=W(i-8)\oplus T(W(i-1)) \quad i \text{ is a multiple of } 8$$

Where the *T (i)* transformation is defined as:

$$T(i)=ByteSub\left(ShiftLeft\left(W(i)\right)\right)\oplus RoundConst$$

The round constant is defined by the following equation:

$$RoundConst=00000010^{(i-8)/8} \quad i \text{ is the round number}$$

Table 1 shows the round constants for all rounds in AES-512.

Table I: Round Constant for AES-512 rounds

Round	i	Round Constant
1	8	0100000000000000
2	16	0200000000000000
3	24	0400000000000000
4	32	0800000000000000
5	40	1000000000000000
6	48	2000000000000000
7	56	4000000000000000
8	64	8000000000000000
9	72	1B00000000000000
10	80	3600000000000000

The round structure of the AES-512 algorithm (shown in Figure 4) uses the transformation defined in the previous section. First, byte substitution is performed on 512 bits data, followed by row rotation according to the row number, where 0-7 left rotations are performed in this step. Then, the columns are multiplied by the new defined matrix column by column in the Mix Column transformation (except in the 10^{th} round). The last operation will be the bitwise XORing with the round key expanded using the key expansion process. The output at of the 10^{th} round will be the 512-bit encrypted message.

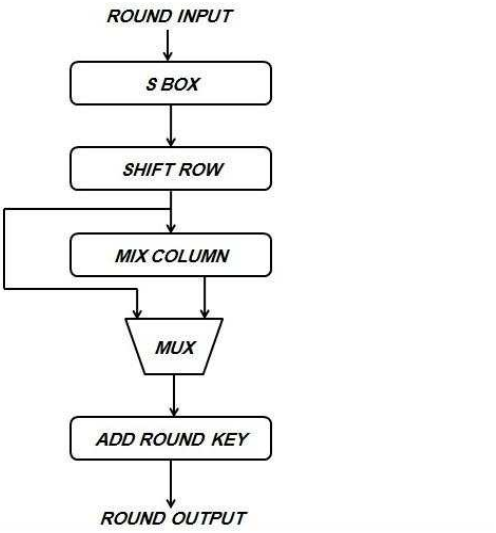

Figure 4. Single round of AES-512 algorithm

V. Results and Evaluation

The proposed AES-512 bits designs and the original AES algorithm design were coded in VHDL. The codes were verified and simulated for functional correctness. Figures 5, and 6 shows an encryption/decryption example on ModelSim.

Xilinx ISE 13.1 was used to synthesize the VHDL codes of both designs using the Virtex-6 and Vertix-7 FPGA family devices. Virtex-7 is a new FPGA family based on 28nm architecture designed for high performance, high throughput, and low power consumption. Virtex-7 power efficiency helps in mitigating the power requirements of the increased design area.

The synthesis results are shown in Table II. The results include the operating frequency, throughput, and the number of control logic blocks (CLBs). Table II shows also the results for Virtex-6. The proposed Algorithm has about 230% higher throughput compared to the original AES-128 design. On the other hand the area increase is about 206% compared to the AES-128. Using larger key size makes the algorithm more immune against brute-force attack which results in more security. The new algorithm is ideal for secure system with high throughput requirements where the hardware area and energy consumptions do not have large significance. Virtex-7 shows a promising results compared with older FPGA families from Xilinx like Virtex-5 and Virtex-6 .

Table III shows a comparison of the proposed AES-512 algorithm with previous implementation for the AES 128 bits. The AES-512 provides a significant Throughput/Area increase of about 210% compared with the results presented in [17][18][19]. The usage of Virtex-7 family will help in reducing the power requirements for the AES-512. A basic power evaluation using Xilinx XPower Analyzer (XPA) for AES-512 with Virtex-7 shows static power saving over older Virtex families of about 60%.

VI. Conclusion

Due to the increasing needs for secure communications, a more safe and secure cryptographic algorithms has to be proposed and implemented. The Advanced Encryption Standard (AES-128bit) is widely used nowadays in many applications. In this paper, we proposed a new variation of AES (AES-512) with 512-bit input block and 512-bit key size compared with 128-bit in the original AES-128 algorithm. A complete hardware implementation for the new AES-512 was also presented in this paper.

After comparing the hardware implementation results, we found that our new design has about 230% throughput compared with the original AES-128 design. The larger key size make the algorithm more secure, and the larger input block increases the throughput. The extra increase in area can be tolerated and makes the proposed algorithm ideal applications in which high level of security and high throughput are required such as in multimedia communications.

Acknowledgment

The authors wish to thanks Jordan University of Science and Technology, Cryptographic Hardware and information Security lab (CHiS lab), and the Scientific Research Support Fund in MOHE in Jordan for their support to this research.

Table II: AES-512 Implementation results

Design	*Device*	*Freq. (MHz)*	*Area (CLBs)*	*Throughput (Mbps)*
AES-512	*Virtex-7*	*318.7*	*6701*	*1163*
AES-512	*Virtex-6*	*261.1*	*6701*	*954*
AES-128	*Virtex-7*	*378.4*	*3243*	*495*
AES-128	*Virtex-6*	*250.4*	*2243*	*320*

Table III: AES-512 Comparison with other Designs

DESIGN	THROUGHPUT (MBPS)	AREA (CLBS)	THROUGHPUT /AREA
AES-512	1163	6701	0.173
[17][19]	294	3528	0.083
[18]	353	5673	0.062

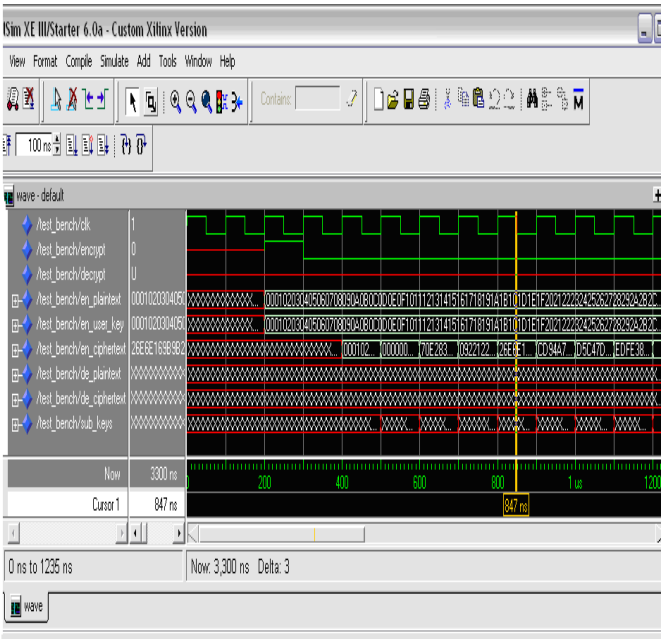

Figure 5. Encryption starts example

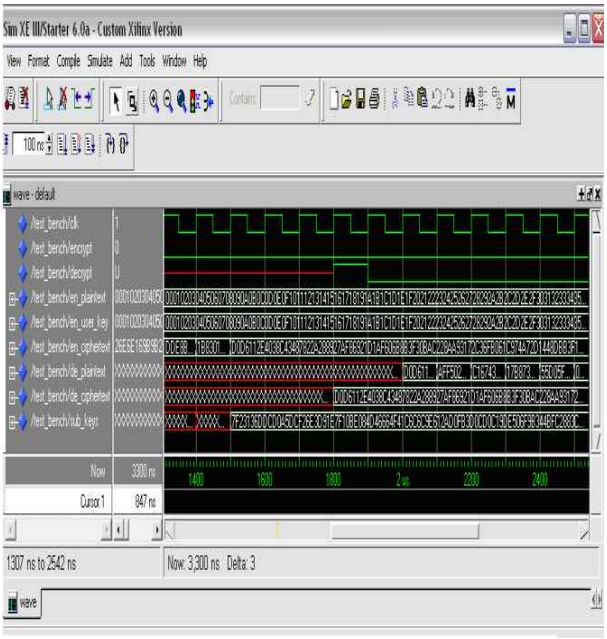

Figure 6. Encryption/decryption example

REFERENCES

[1]. Federal Information Processing Standards (FIPS), "Announcing the Advanced Encryption Standard (AES)," National Institute of Standards and Technology (NIST), November 2001.

[2]. Joan Daemen and Vincent Rijmen "The Design of Rijndael" Springer 2002.

[3]. J. Elbirt, W. Yip, B. Chetwynd, C. Paar, "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists," IEEE Transactions on VLSI Systems, Vol. 9, No. 4, pp. 545–557, August 2001.

[4]. Viktor Fischer, Milos Drutarovsky, "Two Methods of Rijndael Implementation in Reconfigurable Hardware". CHES 2001, Proceedings, LNCS Vol. 2162, pp. 77-92.

[5]. Akashi Satoh and Sumio Morioka and Kohji Takano and Seiji Munetoh "A Compact Rijndael Hardware Architecture with S-Box Optimization" ASIACRYPT 2001, Proceedings, LNCS Vol. 2248, pp. 239-254.

[6]. Gaël Rouvroy, François-Xavier Standaert, Jean-Jacques Quisquater, Jean-Didier Legat "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of AES Rijndael Very Well Suited for Small Embedded Applications" ITCC 2004, IEEE Computer Society.

[7]. K. Gaj and P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware," Proc. 3rd Advanced Encryption Standard Conference, New York, April 2000, pp. 40-54.

[8]. M. Bean et al. "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms". (Nov. 2001).

[9]. J. Daemen and V. Rijmen, "The Rijndael Block Cipher: AES Proposal," Proc. 1st AES Candidate Conf., 1998;

[10]. H. Kuo and I. Verbauwhede, "Architectural Optimization for a 1.82-Gbits/sec VLSI Implementation of the AES Rijndael Algorithm" Cryptographic Hardware and Embedded Systems (CHES 2001), Lecture Notes in Computer Science 2162, Springer-Verlag, Heidelberg, Germany, 2001, pp. 53-67.

[11]. F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, "Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs", Proc. CHES 2003, , 2003.

[12]. Alma'aitah, A., Abid, Z.-E., "Area efficient-high throughput sub-pipelined design of the AES in CMOS 180nm", Design and Test

[13]. Workshop (IDT), 2010 5th International, On page(s): 31 - 36, V, 14-15 Dec. 2010

[14]. Mourad,O.-c., Lotfy, S.-M., Noureddine, M., Ahmed, B., Camel, T., "AES Embedded Hardware Implementation", Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on, On page(s): 103 - 109, V:5-8 Aug. 2007.

[15]. Gang Zhou, Michalik, H., Hinsenkamp, L., "Efficient and High-Throughput Implementations of AES-GCM on FPGAs", Field-Programmable Technology, 2007. ICFPT 2007. International Conference on, On page(s): 185 - 192, Volume: Issue: , 12-14 Dec. 2007.

[16]. Qingfu Cao, Shuguo Li, "A high-throughput cost-effective ASIC implementation of the AES Algorithm", ASIC, 2009. ASICON '09. IEEE 8th International Conference on, On page(s): 805 - 808, Volume: Issue: , 20-23 Oct. 2009.

[17]. J. Wolkerstofer,An ASIC implementation of the AES Mixcolumn Operation, in Proc. Austrochip 2001,Vienna, Austria, Oct,pp.129-132, 2001.

[18]. Elbirt A.J., Yip W., Chetwynd B., Paar C. "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists", IEEE Transactions onVery Large Scale Integration (VLSI) Systems, Volume: 9 Issue: 4, August 2001.

[19]. Dandalis A., Prasanna V.K., Rolim J.D. "A Comparative Study of Performance of AES Final Candidates Using FPGAs", Cryptographic Hardware and Embedded Systems Workshop (CHES 2000), Worcester, Massachusetts, 2000.

[20]. Elbirt A.J., Yip W., Chetwynd B., Paar C. "An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists", Third Advanced Encryption Standard (AES3) Candidate Conference, New York, 2000

[21]. L. A. Tawalbeh and Q. Abu Al-Haija. "Enhanced FPGA Implementations for Doubling Oriented and Jacobi- Quartics Elliptic Curves Cryptography". Journal of Information Assurance and Security, Dynamic Publishers, Inc., USA. March 2011