Efficient AES-XTS Pipelined Implementation on FPGA

** Shakil Ahmed and * Muhammad Naseem*

***Abstract*—In past years, it has been considered that only data communicated via networks need to be secured. This paradigm now shifted towards securing data at rest. With its increasing significance, IEEE has introduced a mode of Advanced Encryption Standard (AES) named as XTS-AES. Few of its implementations exist. This paper presents a high throughput and highly efficient fully unrolled pipelined design of AES-XTS on FPGA. The proposed implementation incorporates only one AES core for both tweak value encryption as well as data encryption. Further our proposed design calculates tweak value in parallel to data encryption/decryption process. The results have achieved a throughput of 35.8 Gbps with an efficiency of 8.4 Mbps/slice. This design offers the best result for Throughput/Area that is 4.641 Mbps/area.**

***Index Terms* —Cryptography, XTS-AES, FPGA.**

I. INTRODUCTION

Securing data at move is considered to be one of the critical issues in current ages. Now, in the past few years the focus has been shifted towards securing data at rest. Theft of confidential data includes capturing of data as well as complete takeover of physical equipment [1]. Encryption and authentication are considered two main security mechanisms [2] to prevent such threats. IEEE, keeping this in view has introduced a mode named as XTS-AES for secure of static data via Encryption.

Different methods of storage encryption are given in [2]. Storage encryption can be done either through software or it may be done through dedicated hardware. Software based encryption is relatively slow [1], consumes more power and also not secure [3] in comparison to hardware based encryption. In this paper we present fully unrolled pipelined design of XTS-AES on FPGA.

XTS-AES is a tweakable block cipher that encrypts or decrypts 128 bit of data or it's multiple. A Cipher operation for a given plaintext and key k is modelled as $C = E_K(P)$, where as a block cipher that involves tweakability is described as $C = E_K(T, P)$, where T is the value of tweak. A non-negative integer is assigned to each data unit that is called tweak value. Encryption of tweak as well as of data is being done using AES algorithm.

This implementation uses the fully unrolled pipelined design using only one AES core to encrypt the tweak as well as data. Also we have also incorporated the multiplication of tweak value by α^j in a parallel approach thus minimizing the computing cycles.

Section II presents literature review, and then we briefly discuss about XTS-AES algorithm in Section III, followed by the proposed implementation, its performance results and comparison in sections IV-VII. Finally we give our conclusions.

II. LITERATURE REVIEW

In this section, we have summarized results that have implemented AES on FPGA using pipelining. This related work includes one parameter throughput/area since one of the common parameters Throughput/Slice (TPS) [4] is used to compare FPGA designs but this parameter sometimes does not reflect true performance of the design as many of the designs use heavy memory resources e.g., BRAMs, DSP Slices etc but it is not used in calculation of Throughput/Slice. So one other parameter Throughput/Area (TPA) is used so that efficiency of the design can be judged truly as being highlighted in [5]. Therefore, while comparing with our proposed design we have used both parameters.

Elbirt [4] proposed design which achieved throughput of 1.94 Gb/sec with an expense of 10992 CLB slices. Standert [6] achieved very good throughput and efficiency but with high use of number of BRAMs that effects throughput/area parameter. Jarvenin [7] got throughput of 17.8 Gbps with 10750 slices with an efficiency of 1.656 Mbps/slice. Hodjat [8] proposed two designs, one of with Block RAMs and one without it and got a throughput of 21 Gbps. Saggese [5] got high throughput design but with higher number of BRAMs. The design by S.M. Yoo [9] explored the good operating frequency but high use of BRAMs and slices that deteriorates the efficiency as well as TPA. The other designs with its performance parameters are listed in Table I. One of the recent implementations given in [10] achieved the highest efficiency but at the cost of higher number of BRAMs and DSP slices. This design has been implemented on Virtex 5 FPGA and in this implementation we are utilizing the same device. Our design although has got less efficiency but we have effectively used the hardware resources and used only 27 BRAMs thus in terms of TPA our design is best in all. Our design also has included key expansion while the design given in [10] does not incorporate this. Table I provides summary of all key parameters of these AES pipelined FPGA implementations.

* Computer Engineering Department, Sir Syed University of Engineering and Technology, Karachi, Pakistan (atshakil@yahoo.com)

TABLE I
SUMMARY OF AES PIPELINED FPGA IMPLEMENTATIONS

Ref	CLB Slices	Device	Throughput (Gpbs)	Freq (MHz)	Block RAMs	Efficiency (Througput/Slice) (Mbps/slice)	Efficiency (Throughput/ Area) (Mbps/Area)
[4]	10992	XCV1000-4	1.94	31.8	0	0.176	0.176
[6]	2784	XCV3200E-8	11.77		100	4.228	0.755
[7]	10750	XC2V2000-5	17.8		0	1.656	1.656
[8]	9446	XC2VP20-7	21.64	169.1	0	2.291	2.291
[5]	5810	XVE2000-8	20.3		100	3.494	1.091
[9]	7761	Xc2vp70-7	29.8	232.6	200	3.840	0.893
[11]	2457	XCV812E-8	12	93.9	226	4.884	0.382
[8]	5177	XC2VP20-7	21.54	168.3	84	4.161	1.352
[12]	12600	XCV1000 –6	12.1		80	0.960	0.530
[13]	2136	XCV-812	2.87	22.41	100	1.343	0.192
[14]	4901	XC2V4000	23.57		95	4.810	1.381
[11]	2000	XCV812E	12.02		244	6.010	0.361
[15]	3576	XC2V6000-6	24.92	194.7	80	6.970	1.803
[10]	3775	Virtex V-SX50T[2]-3ff1136	56.3	440	80	14.914	4.017
Ours	4,258	Virtex V-XC5vlx50-3ff676	35.8	279.822	27	8.408	4.641

III. XTS-AES ALGORITHM

Figure 1 [16] shows the encryption architecture of AES-XTS. Key 2 is used to encrypt the tweak while key 1 is used to encrypt the plaintext. The encryption of tweak/data using AES involves four steps as outlined in [17], namely Byte-Substitution, Shift-Row, Mix-Column and Add-Round-key. During the ten rounds of encryption process, Mix-Column is not performed in the end. For AES encryption, a key expansion method is used to generate the round keys. The round keys can be generated using either pre-computation method or on the fly method.

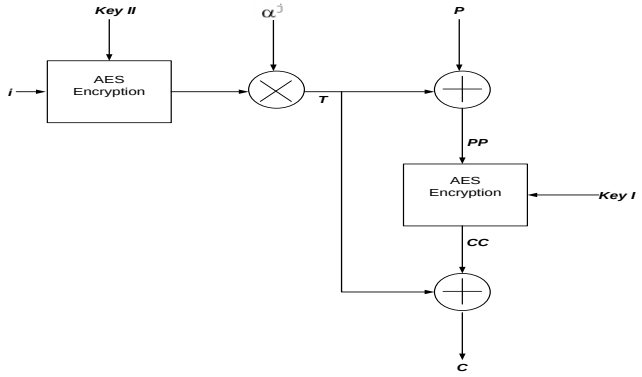

Fig. 1. AES-XTS Encryption block diagram

A. Multiplication by a primitive element α

A 16-byte value which is generated after AES encryption is multiplied by α^j, where α is a primitive element of GF(2^{128}). This multiplication is done in following manner

Input: j is the power of α
byte array: $a_0[l], l = 0,1,2, \ldots .15$
Output: byte array: $a_j[l], l = 0,1,2, \ldots .15$, where where $a_0[0]$ is the first byte of the AES block.

The output array is defined repeatedly by the following formula where i is iterated from 0 to j. Equation (1) is used to calculate the output array recursively

$$a_{i+1}[0] \longleftarrow (2(a_i[0] \bmod 128)) \oplus (135 \lfloor a_i[15]/128 \rfloor)$$

$$a_{i+1}[l] \longleftarrow (2(a_i[l] \bmod 128)) \oplus \lfloor a_i[l-1]/128 \rfloor, \quad l = 1,2, \ldots .15 \qquad (1)$$

B. XTS-AES Encryption/Decryption Procedure

The XTS-AES encryption or decryption involves one process in common that is tweak value encryption. Depending upon the core implementation, there is either data encryption or decryption. For XTS-AES encryption to work; the value of

128-bit tweak needs to be encrypted using AES algorithm using key 2. After tweak encryption, there is a modular multiplication with α^j. The resultant is exclusive OR with plaintext or ciphertext depending on the function of the core. Then there is a either data encryption or ciphertext decryption using key 1. For encryption within XTS-AES, the keys have been computed on-the-fly; while for decryption; the sub-keys need to be pre-calculated. After the data encryption or decryption process, there is final exclusive OR with resultant of modular multiplied value to get the final ciphertext or plaintext.

The process of AES-XTS encryption/decryption, can be summarized using following equations.

1) $T \leftarrow AES - enc\ (key_2, i) \otimes \alpha^j$
2) $PP \mid CC \leftarrow (P|C) \oplus T$
3) $CC \mid PP = (AES - enc \mid AES - dec)(key_1, PP \mid CC)$
4) $C \mid P = (CC \mid PP) \oplus T$

Where
$\oplus$ Bit-wise XOR (Exclusive OR) operation

$\otimes$ Modular multiplication of two polynomials over the binary field GF(2), modulo $x^{128} + x^7 + x^2 + x + 1$,

α, A primitive element of $GF(2^{128})$ that corresponds to polynomial x (i.e., 0000…0102); In both Gal`ois Field is abbreviated as GF.

IV. Proposed Hardware Architecture

This work has been focused to achieve a sustainable throughput for the secondary storage devices that involve different data transfer standards e.g., SATA. The aim of this work is to not only get the higher throughput but with minimum area so that higher efficiency (Throughput/Slice) and (Throughput/Area) is obtained. The main contribution of this work is to integrate such kind of techniques that helps us in achieving high performance. The techniques include use of parallelism e.g., Pipelining and sharing of resources that helps in achieving the higher throughput with minimum area. In the AES-XTS algorithm, the resource sharing can be done for tweak value encryption and data encryption as single core can be used to do both in a time multiplexed method. Further there could be resource sharing for data encryption and round key generation as they use the same Substitution Box (Sbox). Further the modular multiplication is being done in parallel to the last block data encryption/decryption to minimize computing cycles. After incorporating all these techniques our core has reached a throughput of 35.8 Gb/sec, that is sufficient for SATA III data transfer rate, and also would be able to match to the data rate standards which requires higher throughput for data encryption/decryption in future. We first designed our AES-XTS core in an iterative fashion and results were verified using test vectors present in FIPS P1619 standard [16]. After verification of the results we pipelined our design and got the highest efficient (TPA) design using fully unrolled outer round pipelined technique. We used single core for data as well as for data encryption. Our primary focus of this implementation was not only to achieve high throughput but also an efficient one. The proposed AES-XTS architecture is depicted in figure 2. Our core is capable of processing all inputs as being given in original AES-XTS approved standard.

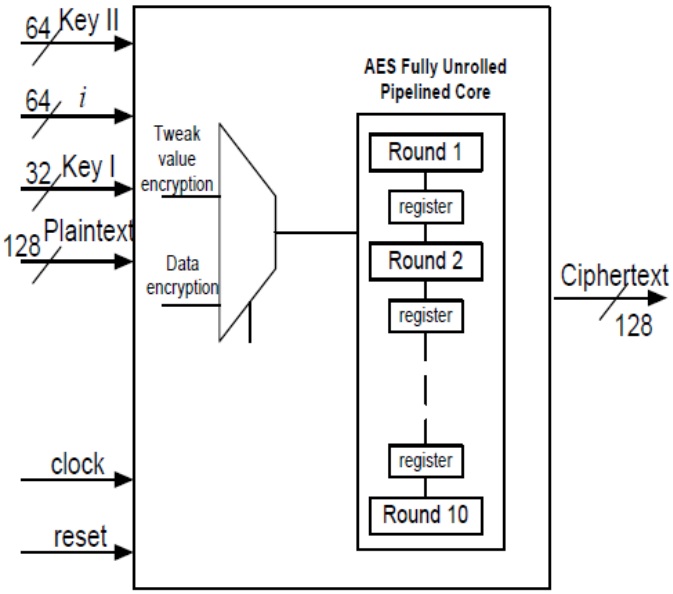

Fig. 2. XTS-AES Pipelined Implementation Block Diagram
Following considerations were taken into account for the core to operate

1) The key_2 and i has been taken input in the chunk of 64 bits. The reason of this width selection is due to our FPGA I/O constraint as our selected FPGA has limited I/O pins. Therefore a trade off was done for the selection of width of no of input and output pins. As data encryption can not start until tweak value encryption therefore we have kept the width of these two pins at 64 bits.
2) The width of key_1 is set to 32 bits as this is required for data encryption/decryption and this process can not start until tweak value is encrypted so we can load this in 4 clock cycles parallel to the other processes being done.
3) The width of plaintext and ciphertext is set to 128-bits each so our core could be able to produce 128-bits output at every clock cycle.

The AES algorithm processes 128 bits data in a pipelined fashion to give the encrypted tweak value or cipher-text in time multiplexed method. The only encountered delay is due to tweak value encryption. After this initial delay, at every clock cycle, the data is being encrypted or decrypted.

The implementation uses fully unrolled pipelined design that incorporates outer round pipelined technique. During the tweak value encryption the data being provided need to be buffered. Therefore we have used a buffer of 1920 bits to store the data. Key I is loaded in the 32 bits form since it is required for data encryption and it will not consume any cycles because

during tweak value encryption it is loading in parallel. Our core is generating ciphertext of 128 bits so there is an initial delay of data loading cycles and tweak value encryption. Once the pipeline is full, successive blocks are encrypted one after another and generate cipher-text of 128 bits at every clock cycle. Since tweak value need to be encrypted before data encryption, therefore we have used time multiplexed method to use the same pipelined core to do the both thus providing a significant savings of hardware area.

A. Tweak Value Calculation

One of the significant features of our XTS-AES implementation is calculation of tweak value. Each time, encrypted tweak value is multiplied by α^j in a recursive manner. Our proposed design calculates this value in parallel to the data encryption or decryption thus minimizing the computing cycles of the output to be calculated. The initial 26 cycles includes 1 initialization cycle, 2 loading cycles for key 2 and I, 10 processing cycles for tweak value encryption and 10 for data encryption and 2 for exclusive or processes and one for output. Figure 3 shows diagrammatic representation of the no of bits versus no of processing cycles. The figure shows that after certain period of time, the number of clock cycles is only a function of no of bits being processed.

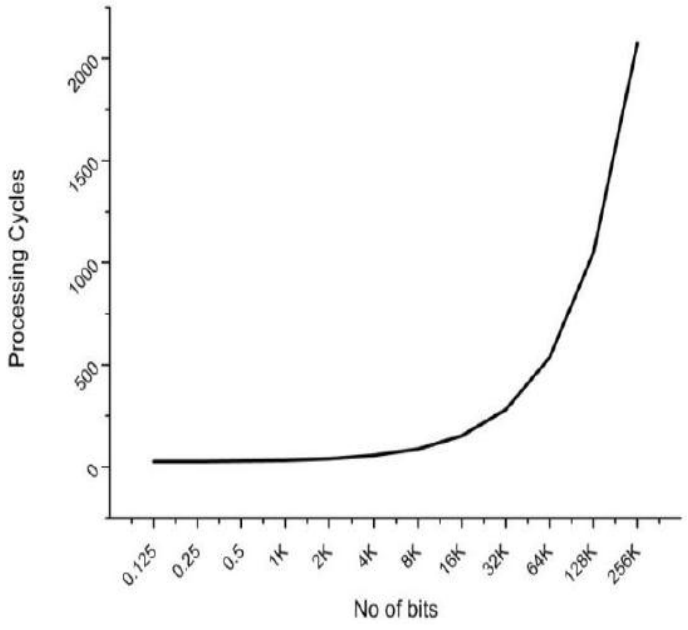

Fig. 3. No of Cycles versus No of bits processed

In the same way our measured throughput reaches at 35.8 Gbps as the number of bits processed is 256K. After this point our measured throughput settles down at 35.8 Gbps. Figure 4 gives a diagrammatic representation of Throughput versus no of bits processed.

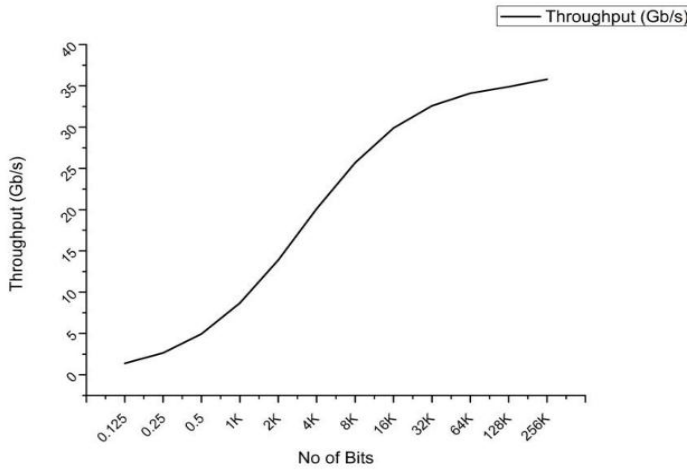

Fig. 4. Throughput versus No of bits processed

B. Key Expansion Module

The key expansion module was used to generate the sub-keys for the data encryption and decryption. The round keys can be generated either using pre-computation method or on-the-fly method. For the tweak value encryption we generated each round key on-the-fly parallel to the encryption process. For data encryption, the sub-keys were calculated during the tweak value encryption process. In the case if key 1 is changed then its sub-keys can also be generated on-the-fly. For data decryption, all sub-keys need to be calculated prior to the start of the data decryption process. For this purpose we have taken an advantage of the algorithm such that all sub-keys of key 1 were calculated during the tweak encryption process. However during the data decryption process, if key 1 is changed then the throughput may be limited but this chance is unavoidable as the same key is used to decrypt the data that was used to encrypt it. The use of Substitution Boxes is shared for round key generation and data encryption thus minimizing the area requirements for the overall algorithm.

V. RESULTS AND DISCUSSION

The design was developed using Verilog HDL and simulated using Xilinx ISE Webpack 12.3. This section shows the results after successful synthesis, map and place and route. The validation and verification was done using Virtex 5 FPGA with ff676 package, speed grade -3 (Virtex V- XC5vlx50-3ff676). The results are also being compared with the existing other AES-XTS results to date. Our design was tested using the test vectors available in NIST P1619 [16]. Several design parameters were considered for an effective comparison.

These parameters include 1) Throughput 2) Efficiency (Mbps/slice) and (Mbps/Area). The throughput is being calculated using Eq (1).

Throughput = (Data_block_size * Clock Frequency)/Clock cycles (1)

Table II and Table III shows the complete hardware resource summary for the XTS-AES encryption and XTS-AES decryption after place and route report.

TABLE II
AES-XTS PIPELINED ENCRYPTION RESOURCES SUMMARY

Number of Slice LUTs	12,172
Number of occupied Slices	4,258
Number of LUT Flip Flop pairs used	12,458
Number of bonded IOBs	418
Number of BlockRAM/FIFO	27
Frequency	279.822MHz
Throughput	35.8 Gb/sec

A. AES-XTS Decryption Results

Using the same technique used for the pipelined encryption we have implemented AES-XTS pipelined decryption. The major difference between both implementations is due to that AES-XTS decryption uses tweak value encryption as well as data decryption therefore it has resulted in lower throughput with more number of slices. The implementation of fully unrolled pipelined decryption design resulted in a maximum throughput of 30 Gb/sec with 28 Block RAMs and 5,117 slices. The Efficiency of the system that is throughput/slice can be calculated as Efficiency= 30 Gb/sec/5,117=5.86 Mb/sec/slice

Table III. shows the hardware resource summary of the XTS-AES pipelined decryption implementation.

TABLE III
AES-XTS PIPELINED DECRYPTION RESOURCES SUMMARY

Number of Slice LUTs	14,152
Number of occupied Slices	5,117
Number of LUT Flip Flop pairs used	15,085
Number of bonded IOBs	418
Number of BlockRAM/FIFO	28
Frequency	234.709MHz
Throughput	30 Gb/sec

VI. Comparative Analysis

There are numerous AES FPGA pipelined implementations with different modes. From Table I, it seems that one of the designs given by Drimer [10] has got the highest throughput in all of the designs but the efficiency of the design is reflected if measured in terms of Mbps/Area. In our proposed design we have calculated efficiency but most of the designs present in literature do not provide this information. The throughput and efficiency achieved by [10] is the highest one but with the expense of high number of BRAMs that are 80. Another design given by [15] has got better efficiency but it also includes high number of BRAMs. Our design has got the second highest efficient design. But after inclusion of throughput/area parameter that includes BRAMs also (1 BRAM=128 slices), our design has got the best throughput/ area result. Therefore in conclusion we can say that the Drimer design has got the highest Mbps/slice while our design has got the highest throughput/area. In calculation of Efficiency (Mbps/slice), we have listed two efficiencies for our design and the design given in [10] since they are implemented on Virtex 5 and in this device a slice is made up of 4 LUTs instead of 2 for previous Xilinx devices.

VII. AES-XTS Decryption Comparison with AES Decryption Implementations

Table IV compares our AES-XTS decryption implementations to the AES decryption implementations to date. In the existing literature there are only few decryption implementations. The implementation given by S.M. Yoo (Yoo, Kotturi et al. 2005) has got an efficiency of 4.3 Mbps/slice with an inclusion of 200 BRAMs. Also this implementation may not include key expansion. This implementation uses the pipelined methodology. In comparison to our pipelined implementation our design has got 5.86 Mbps/slice which is relatively better. Other decryption implementations based on iterative method has got better throughput than that of us but our design has got best throughput/area relative to other two designs

TABLE IV
COMPARISON WITH AES DECRYPTION IMPLEMENTATIONS

Ref	Mode	CLB Slices	Device	Throughput (Gpbs)	Block RAM	Eff (Mbps/ slice)	Eff (Mbps/ Area)
[13]	Dec	5677	XCV 2000 E	4.121	0	0.72	0.72
[9]	Dec Pipelined	6541	Virtex-II Pro	28.44	200	4.34	0.88
[11]	Dec Iterative	2121	XCV600E	0.557	18	0.26	0.13
Ours	Dec	5117/ 10234	Virtex V- XC5vlx 50-3ff676	30	28	5.86/ 2.93	3.44

Table V shows the comparison of our XTS-AES implementations to existing AES-XTS implementations to date. According the results shown, there are only few implementations of XTS-AES exist. These designs are low throughput designs. The design given in [18] reports the throughput of 2.8 Gbps while the same author in a modified implementation reports the throughput as 3.5 Gbps. These two designs are low throughput design and can match only the speed of SATA I data transfer rate. SATA III standard provides data transfer rate upto 6 Gbps therefore these two designs given in [7] do not incorporate it. The design given by Elliptic [19] has got the 10 Gbps but all other parameters are not given in the product data sheet so that efficiency of the design could not be truly interpreted. Our design has got the highest throughput as well as highest Mbps/slice and Mbps/Area.

TABLE V
COMPARISON WITH XTS-AES IMPLEMENTATIONS

Device	Slices	Frequency (MHz)	Throughput $(Gbps)$	BRAMs	Efficiency Throughput/ Slice (Mbps/Slice)	Efficiency Throughput/ Area (Mbps/Area)	Ref
Virtex 5	1470	209	2.8	0	1.9	1.9	[18]
Virtex 5	1616	-	3.5	0	2.16	2.16	[18]
-	-	-	10	-	-	-	[19]
Virtex 5-3	4,258	279.822	35.8	27	8.40	4.64	This work

VIII. CONCLUSIONS

In this paper a high efficient AES-XTS FPGA core implementation is presented. The design incorporated several techniques that included outer round pipelining, use of single core to do the tweak encryption as well as data encryption and calculation of tweak in parallel to the data encryption. After all these optimizations, our design occupied 4258 slices only with a throughput of 35.8 Gb/sec. Our design not only provided an increased throughput compared to the other implementations but it is also the efficient in terms of throughput/slice which is 8.4 Mbps/Slice. Future work includes optimization of the algorithm to further reduce the slice count as well as increasing the throughput.

REFERENCES

[1] C. Laird, "Taking a hard-line approach to encryption," Computer, vol. 40, pp. 13-15, 2007.

[2] K. Scarfone, et al., "Guide to storage encryption technologies for end user devices," NIST Special Publication, vol. 800, p. 111, 2007.

[3] L. Hars, "Discryption: Internal hard-disk encryption for secure storage," Computer, vol. 40, pp. 103-105, 2007.

[4] A. J. Elbirt, et al., "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," in Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, 2000.

[5] G. Saggese, et al., "An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm," Field Programmable Logic and Application, pp. 292-302, 2003.

[6] F. X. Standaert, et al., "Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements and design tradeoffs," Cryptographic Hardware and Embedded Systems-CHES 2003, pp. 334-350, 2003.

[7] K. Jarvinen, et al., "Comparative survey of high-performance cryptographic algorithm implementations on FPGAs," IEE Proceedings on Information Security, vol. 152, pp. 3-12, 2005.

[8] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined AES processor on FPGA," in Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on, 2004, pp. 308-309.

[9] S. M. Yoo, et al., "An AES crypto chip using a high-speed parallel pipelined architecture," Microprocessors and Microsystems, vol. 29, pp. 317-326, 2005.

[10] S. Drimer, et al., "DSPs, BRAMs, and a pinch of logic: Extended recipes for AES on FPGAs," ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol. 3, p. 3, 2010.

[11] M. McLoone and J. V. McCanny, "Rijndael FPGA implementations utilising look-up tables," The Journal of VLSI Signal Processing, vol. 34, pp. 261-275, 2003.

[12] K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays," Topics in Cryptology—CT-RSA 2001, pp. 84-99, 2001.

[13] N. A. Saqib, et al., "AES algorithm implementation-an efficient approach for sequential and pipeline architectures," in Computer Science, 2003. ENC 2003. Proceedings of the Fourth Mexican International Conference on, 2003, pp. 126-130.

[14] J. Zambreno, et al., "Exploring area/delay tradeoffs in an AES FPGA implementation," Field Programmable Logic and Application, pp. 575-585, 2004.

[15] J. M. Granado-Criado, et al., "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," INTEGRATION, the VLSI journal, vol. 43, pp. 72-80, 2010.

[16] I. S. i. S. W. Group, "IEEE P1619/D19: Draft standard for cryptographic protection of data on block-oriented storage devices," ed: July, 2007.

[17] P. FIPS, "197," Advanced Encryption Standard (AES), vol. 26, 2001.

[18] E. Hatzidimitriou, et al., "Exploration and enhancement of P1619-based crypto-cores for efficient performance," in Consumer Electronics (ICCE), 2011 IEEE International Conference on, 2011, pp. 361-362.

[19] Elliptic, "XTS-AES Core," ed, 2009.