Accelerating the AES Algorithm using OpenCL

Theodora Sanida
Department of Electrical & Computer Engineering
University of Western Macedonia
Kozani, 50131, Greece
thsanida@uowm.gr

Argyrios Sideris
Department of Electrical & Computer Engineering
University of Western Macedonia
Kozani, 50131, Greece
asideris@uowm.gr

Minas Dasygenis
Department of Electrical & Computer Engineering
University of Western Macedonia
Kozani, 50131, Greece
mdasyg@ieee.org

***Abstract*—Nowadays, cryptography plays an important role in the field of information security. The most common symmetric cryptographic algorithm is Advanced Encryption Standard (AES), which is based on the well-known Rijndael algorithm and is used worldwide in every domain. In this document, we present the implementation of the AES algorithm in two parallel modes of operation (CTR and XTS) with the OpenCL programming language. We used OpenCL because it is designed for parallel computing on heterogeneous platforms and ensures portability. Furthermore, we applied 128, 192 and 256 bit cryptographic key size and a file size ranging from 512B to 8MB to evaluate the performance of the kernel runtime and throughput (Gbps). The results have shown that, the performance of the CTR mode is better than the XTS mode. CTR mode speeds up the process of encryption with 128 bit key over 10.15%, 192 bit key over 10.09% and 256 bit key over 10.05%. The decryption process shows 128 bit key acceleration over 10.11%, 192 bit key over 10.05% and 256 bit key over 10.02%. Finally, comparing the results of our implementation to other similar parallel models, we have achieved better throughput performance.**

***Index Terms*—Cryptography, OpenCL, Advanced Encryption Standard (AES), Rijndael, CTR, XTS, Graphics Processing Unit (GPU), Block cipher.**

I. Introduction

Today, in order to increase performance, many cryptographic algorithms are implemented on GPUs. In recent years, the use of GPUs has increased significantly as they perform many more floating point operations than CPU, at the same time [1]. The AES, which encrypts and decrypts the data in blocks, has been adopted as a standard for symmetric cryptography by the National Institute of Standards and Technology (NIST) [2]. AES uses 128 bit blocks and its cryptographic key can be 128, 192 or 256 bits. Because GPUs perform many operations at the same time, the AES algorithm is a good application for this architecture [3].

The AES specification defines many modes of operation. The most common being Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), XEX-based tweaked-codebook mode with ciphertext stealing (XTS) and Counter mode (CTR) [4]. The ECB, CTR and XTS support parallel operations in both encryption and decryption and can benefit from a parallel GPU application. We used CTR and XTS mode, as they allow simultaneous execution of the encryption and decryption process.We did not use the ECB mode of operation because it lacks integrity.

In this paper, we present the findings of our research endeavor on implementing the algorithm AES in CTR and XTS mode with cryptographic keys 128, 192 and 256 bit in the GPU. The implementation of AES was performed with OpenCL programming language. We have used the OpenCL instead of other similar programming paradigms, because it is designed to accelerate parallel computing on heterogeneous platforms, while ensuring portability [5]. We compare the results of the three AES key sizes in the encryption and decryption process with file sizes ranging from 512B to 8MB and compare the throughput with other similar parallel implementations of other researchers, in Section V.

The paper structure is organized as follows. In Section II we give an overview of research work similar to ours. In section III we discuss the AES algorithm and modes of operation. Section IV provides an outline of the procedure followed for implementing AES in CTR and XTS mode on OpenCL. In section V, we present the results of our research. We conclude our article with the summary in Section VI.

II. Related work

In this section, we present research that is similar to ours. In [6], the authors develop the implementation of the symmetric key cryptography algorithm AES, named clAES, using the OpenCL standard. They present comparisons of the results obtained with clAES benchmarking across different multi-core architectures. They also introduce the basic concepts of AES and OpenCL in order to describe the details of clAES implementation. In their work they do not mention what mode of operation they are implementing.

Gilger *et al.* [7] present the capabilities and limitations of GPU accelerated block cryptography as implemented using CUDA and OpenCL in the form of an OpenSSL cryptographic library. Both GPU frameworks are capable of delivering high performance. They accelerate symmetric block cryptographers (AES, DES, Blowfish, Camellia, CAST5, IDEA) using GPUs compared to traditional CPU applications. In their work they implement the ECB and the CBC mode while we are implement the XTS mode.

In [8], authors propose a CUDA implementation of ECB mode encoding process and CBC mode decoding process on

GPU, in order to improve the efficiency of AES algorithm. Although, their implementation achieves high throughput of 60Gbps with use of the ECB and CBC modes, the algorithm's security is reduced compared to the CTR and XTS modes, where they provide maximum security for brute force attacks.

On [9] they implement an accelerated GPU framework in mobile devices using the XTS AES encryption algorithm. Their results showed that the performance of the proposed system can highly surpass any equivalent CPU based encryption system for mobile device users. In their work they do not implement CBC mode.

In contrast with these authors, we present our implementation of the AES algorithm for GPU using OpenCL for the two highest security modes of CTR and XTS. We performed many tests of the AES Algorithm in CTR and XTS mode using the same graphics card (Nvidia GeForce GTX 1060) and observed large performance differences. We achieved better throughput than the compared implementations of the other presented works.

III. AES Encryption and Decryption Algorithm

AES is the most widely used symmetric encryption algorithm and uses the same key in the encryption and decryption process. The plain text length is fixed at 128 bits, while the length of the cryptographic key varies and can be 128, 192 or 256 bits depending on the degree of security required.

The AES is an iterative algorithm and the total number of rounds required for the encryption/decryption process can be 10, 12 or 14 depending on the size of the cryptographic key used 128, 192 or 256 bits respectively. Each round consists of four transformations (SubBytes, ShiftRows, MixColumns and AddRoundKey). In the final round, the MixColumns transformation is ignored. Table I illustrates the relation between the total number of rounds and key's length.

TABLE I
KEY LENGTH AND TOTAL NUMBER OF ROUNDS (32-BIT WORDS)

AES	**Key Length** *(Nk Words)*	**Block Size** *(Nb Words)*	**Number of Rounds***(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

A mode of operation is an algorithm to provide authenticity or confidentiality. The AES defines many modes of operation, the most common with parallel processing in encryption and decryption are ECB, CTR and XTS [10]. A simple comparison of these parallel modes of operation is listed in Table II.

IV. Implementation

In this section, we present the application of the AES algorithm to CTR and XTS mode of operation using the OpenCL 1.2 in the three sizes of the cryptographic key.

The OpenCL is an open standard aimed to heterogeneous architectures. So, we decided to use the OpenCL language because has the advantage of portability and the same code can be compiled and executed in a variety of architectures, from CPU up to custom FPGA designs.

TABLE II
COMPARISON OF PARALLEL MODES OF OPERATION

Mode	**Description**
ECB	For a given key, the forward cipher function is applied independently and directly to each block of the plaintext. It is prone to cryptanalysis because there is a direct relationship between plaintext and cryptography.
CTR	The application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa.
XTS	The standard provides using a different key for the IV encryption than for the block encryption; this is contrary to the intent of XEX and seems to be rooted in a misinterpretation of the original XEX paper, but does not harm security.

The implementation of the AES algorithm of CTR and XTS mode of operation was performed with 128, 192 and 256 bits cryptographic keys with the C programming language. The implementation of AES in CTR and XTS mode of operation was tested using valid AES test samples and verified the results recommended in [4].

The AES block cipher can be implemented with different granularity of parallelism processing. The granularity of parallel processing can be 1, 4, 8 and 16 bytes per thread. We did several tests for 1, 4, 8 and 16 bytes per thread granularity and found that 16 bytes per thread had significantly higher performance. So our implementation uses 16 bytes per parallel granulation. That means that each thread is mapped to a 16 bytes AES plaintext block and the blocks are processed simultaneously. So, one thread handles an AES block and therefore no synchronization between the threads is required.

The CPU reads the input data file first. The input data file can be either encrypted or decrypted text. The parameters of the AES algorithm are the length of the 128,192 or 256 bit cryptographic key, the cryptographic key, the operating mode (CTR or XTS), and the procedure to be performed (encryption or decryption). All the data are stored in the host CPU main memory.

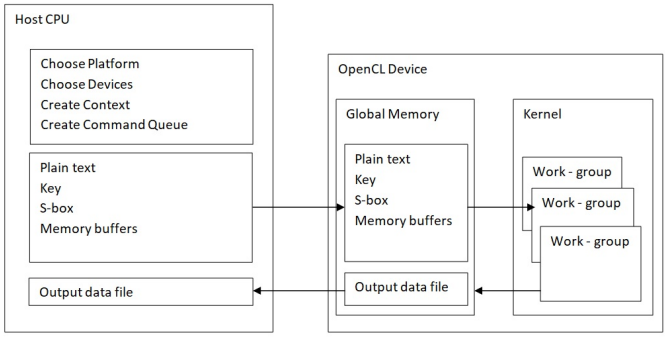

Fig. 1. Arcitecture of the AES with OpenCL

Subsequently we select the platform and device, by quering

the operating system for all the OpenCL devices and secting the one belonging to the GPU class. The parameters for the device are also adjusted by creating the context queue and commands. Here is the definition of the objects that are transferred to the kernel.

The objects that are transferred to the global memory of the GPU are the input data file, the output data file, the encryption key, the S-box as well as various memory buffers. Plaintext is initially transferred and stored on global memory. When the encryption or decryption process begins, plaintext is transferred into the shared memory to share the intermediate data.

We execute the kernel in the GPU. The encryption key and the S-box are calculated serially. The S-box is stored in the shared memory. The round keys are read only data and are shared by all threads, according to the constant memory characteristics. The kernel serially executes transformations of SubBytes, ShiftRows, MixColumns and AddRoundKey. The final round doesn't include the MixColumns. Finally, when the kernel completes its computations, the output data file is stored in the global memory of the GPU and transferred back to the CPU. Figure 1 shows our implementation architecture of the AES with OpenCL.

V. Experimental Results

In this section we will present the performance and compare the AES algorithm in CTR and XTS mode using the OpenCL 1.2 programming language with 128, 192 and 256 bit cryptographic keys. The plaintext or ciphertext file size ranges from 512MB to 8MB.

A. Evaluation Environment

In our experiments, we used Visual Studio 2019 software on Windows 10 Professional 64-bit, Intel Core i7 8750H (2.20GHz) processor, 16GB DDR4 RAM and Nvidia GeForce GTX 1060 graphics card. The Nvidia GeForce GTX 1060 has 1280 cores, GDDR5 memory type, 192 bit bus width, 1709 MHz Processor Clock and 2002 MHz Memory Clock. The detailed technical specifications of the hardware and software used in this work are given in Table III.

TABLE III
Hardware and software technical specifications

Component	Description
Hardware	Processor: Intel Core i7 8750H (2.20GHz), RAM Memory: 16GB DDR4 - 2666MHZ, Graphics Board: Nvidia GeForce GTX 1060, 6144MB
Software	Windows 10 Professional 64-bit, Visual Studio 2019
Drivers	Nvidia CUDA toolkit version: 10.1

B. Evaluation result

Time measurement was performed using the time.h library and specifically clock_gettime() function. Time measurement includes the kernel execution time (in ms) and does not include the time needed to transfer the data in the GPU.

Table IV show the execution times of the kernel (in ms) in CTR mode of operation against the encryption and decryption process at input file sizes from 512B to 8MB with 128, 192 and 256 bit cryptographic keys. Table V reports the execution times of the kernel (in ms) in XTS mode of operation against the encryption and decryption process at file sizes from 512B to 8MB with three cryptographic keys.

TABLE IV
Result of the AES kernel (in ms) in CTR mode of operation with three cryptographic keys

AES Implementation in CTR mode of operation						
	Encryption key length (ms)			Decryption key length (ms)		
File size	128 bit	192 bit	256 bit	128 bit	192 bit	256 bit
512B	0,09	0,12	0,16	0,11	0,14	0,16
1KB	0,14	0,17	0,20	0,16	0,19	0,22
2KB	0,29	0,33	0,36	0,28	0,30	0,33
8KB	0,72	0,73	0,77	0,72	0,76	0,80
32KB	1,90	2,18	2,34	1,81	2,10	2,26
128KB	6,34	8,13	10,44	6,98	8,99	10,91
512KB	26,37	30,37	34,41	28,21	30,32	36,41
2MB	116,01	119,84	124,34	117,04	120,47	124,87
8MB	768,99	777,06	782,42	773,47	782,14	787,14

CTR mode accelerates all key sizes in both encryption and decryption. The tables resulting that the CTR mode shows an accelerated process of 128 bit encryption over 10.15%, 192 bit over 10.09% and 256 bit over 10.05%. The decryption process shows 128 bit key acceleration over 10.11%, 192 bit key over 10.05% and 256 bit key over 10.02%. Figures 2 and 3 show the acceleration of the CTR mode over the XTS mode during the encryption/decryption process with the three key sizes and file size from 512B to 8MB.

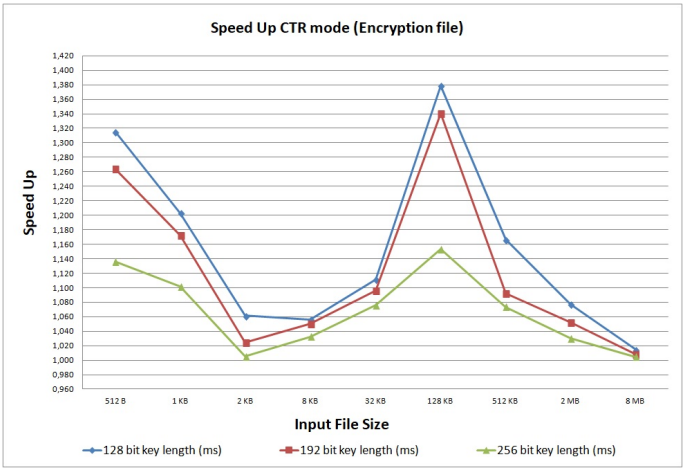

Fig. 2. Speed Up (times) in CTR mode (Encryption file)

Finally, we compare the AES in CTR and XTS mode of operation with similar parallel implementations on the GPU of other researchers. The most significant metric for comparing and evaluating the proposed implementation of the AES with other parallel implementations is throughput. The throughput is calculated using Equation 1.

TABLE V
RESULT OF THE AES KERNEL (IN MS) IN XTS MODE OF OPERATION WITH THREE CRYPTOGRAPHIC KEYS

AES Implementation in XTS mode						
	Encryption key length (ms)			Decryption key length (ms)		
File size	128 bit	192 bit	256 bit	128 bit	192 bit	256 bit
512B	0,12	0,15	0,18	0,15	0,16	0,18
1KB	0,17	0,20	0,22	0,18	0,21	0,25
2KB	0,31	0,33	0,36	0,32	0,34	0,37
8KB	0,76	0,77	0,80	0,77	0,80	0,82
32KB	2,11	2,39	2,51	2,20	2,47	2,41
128KB	8,74	10,89	12,04	9,73	11,74	13,14
512KB	30,75	33,17	36,94	33,73	35,02	37,90
2MB	124,99	126,07	128,11	125,88	127,36	129,22
8MB	780,14	783,74	786,14	782,03	785,93	788,41

$$\text{Throughput} = T_p/E_t \quad (MB/Sec) \tag{1}$$

In Equation 1, T_p is the total plaintext encrypted and E_t is the total encryption or decryption time. Table VI summarizes the results achieved by other parallel implementations in throughput. This was achieved because we significantly improved the performance of the kernel using constant memory and 16 bytes per parallel granulation.

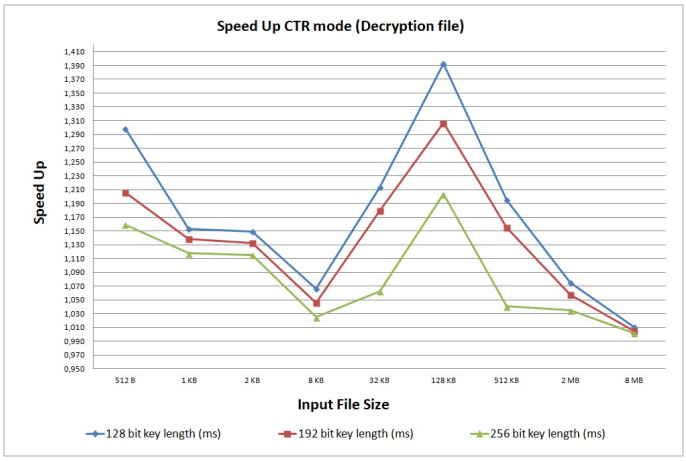

Fig. 3. Speed Up (times) in CTR mode of operation (Decryption file)

VI. CONCLUSIONS

In this work we present the effectiveness of modern graphics processing units in symmetric key cryptographic solutions. We implemented the AES algorithm in CTR and XTS mode of operation with 128, 192 and 256 bit key lengths with OpenCL programming language. We used the Nvidia GeForce GTX 1060 graphics card. The file size we used ranged from 512B to 8MB.

The results showed that CTR mode shows acceleration for all file sizes and for all thee key sizes. In the process of 128 bit encryption, acceleration is over 10.15%, with 192 bit over 10.09% and 256 bit over 10.05%. The decryption process shows 128 bit key acceleration over 10.11%, 192 bit key over 10.05% and 256 bit key over 10.02%.

Finally, the results show that our implementations achieve better throughput in XTS mode of at least 12.86% and in CTR mode of 14.71%. In our future work, we will optimize further the AES algorithm using block cipher modes of operation CTR and XTS, and we will evaluate the algorithm in FPGA architectures.

TABLE VI
THE COMPARISON OF OUR INPLEMENTATION WITH OTHER SIMIULAR IMPLEMENTATION OF THE AES IN CTR AND XTS MODE OF OPERATION

Paper	GPU Device	Language	Mode	Throughput (Gbps)
2014 [11]	ATI HD 7670M	OpenCL	CTR	5.04
2011 [12]	Nvidia GTX 285	OpenCL	XTS	8.59
2011 [12]	Nvidia GTX 285	CUDA	XTS	9.74
2017 [13]	Nvidia GT 555M	OpenCL	CTR	10.00
2009 [14]	Nvidia GT 8800	CUDA	CTR	12.50
Our work	Nvidia GTX 1060	OpenCL	XTS	12.53
Our work	Nvidia GTX 1060	OpenCL	CTR	14.71

REFERENCES

[1] K. Fatahalian and M. Houston, "A closer look at GPUs," *Communications of the ACM*, vol. 51, no. 10, pp. 50–57, 2008.

[2] NIST, "Advanced encryption standard (AES)," Nov 2001. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/197/final

[3] P. FIPS, "197: Specification for the advanced encryption standard, 2001," *http://www. csrc. nist. gov/publications/fips/fips197/fips-197. pdf*, vol. 4, pp. 17–18, 2009.

[4] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," National Inst of Standards and Technology Gaithersburg MD Computer security Div, Tech. Rep., 2001.

[5] "The OpenCL specification v1.2," Nov 2012. [Online]. Available: https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf

[6] O. Gervasi, D. Russo, and F. Vella, "The AES implantation based on OpenCL for multi/many core architecture," in *2010 International Conference on Computational Science and Its Applications*. IEEE, 2010, pp. 129–134.

[7] J. Gilger, J. Barnickel, and U. Meyer, "GPU-acceleration of block ciphers in the OpenSSL cryptographic library," in *International Conference on Information Security*. Springer, 2012, pp. 338–353.

[8] Q. Li, C. Zhong, K. Zhao, X. Mei, and X. Chu, "Implementation and analysis of AES encryption on GPU," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*. IEEE, 2012, pp. 843–848.

[9] M. A. Alomari and K. Samsudin, "A framework for GPU-accelerated AES-XTS encryption in mobile devices," in *TENCON 2011-2011 IEEE region 10 conference*. IEEE, 2011, pp. 144–148.

[10] M. Dworkin, "Recommendation for block cipher modes of operation: methods for format-preserving encryption," *NIST Special Publication*, vol. 800, p. 38G, 2016.

[11] Y. Yuan, Z. He, Z. Gong, and W. Qiu, "Acceleration of AES encryption with OpenCL," in *2014 Ninth Asia Joint Conference on Information Security*. IEEE, 2014, pp. 64–70.

[12] X. Wang, X. Li, M. Zou, and J. Zhou, "AES finalists implementation for GPU and multi-core CPU based on OpenCL," in *2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification*. IEEE, 2011, pp. 38–42.

[13] V. Conti and S. Vitabile, "Design exploration of AES accelerators on FPGAs and GPUs," *Journal of Telecommunications and Information Technology*, 2017.

[14] A. D. Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. IEEE Computer Society, 2009, pp. 1–8.