On GPU Implementations of Encryption Algorithms

Adrian MATEI, Cristian LUPAȘCU, and Ion BICA

***Abstract*—With current evolution in technology and digitization process we see more often that data breaches are a real problem. There is a gap in data privacy because by using digital information processing we have three big zones of interest: data at rest, data in use and data in transit. The objective of this paper is to emphasize the importance of privacy and how data at rest and data in transit can be protected using cryptographic primitives. By using graphical processing units (GPUs) and special processor instructions (AES-NI) we accelerated some of the mainly used symmetric encryption schemes. By comparing the results with classical CPU implementations, we obtained for AES-128 a speedup of approximately x346 in OpenCL implementation and x123 by using CUDA framework. When having compared the same result with the AES-NI instruction set version, we got an improvement factor of 1.6. We also accelerated 3DES using the same frameworks and we managed to quicken the encryption with a speedup of x1101 using CUDA, respectively x2365 for OpenCL.**

***Index Terms*—encryption; hardware acceleration; CUDA; OpenCL; AES-NI.**

I. Introduction

In an environment where personal data protection and privacy preserving operations are more and more emphasized, cryptography has become a general tool to keep those goals achieved. In addition to cryptography, there are several other methods like data anonymization, masking, grouping or removal of personal information, which can be used to ensure that there is no leaking of personal data.

Data migration and data processing in the cloud is a major trend to acquire portability and redundancy of information, but this is affecting all sensitive data that is not properly protected. The most secure way to take care of data sets is to encrypt them in such a way that only the right owner (or someone with the approval of the owner) could decrypt data and use it for different services. It is worth mentioning that in the past few years the community of researchers in cryptography had developed encryption schemes that could allow manipulation on encrypted data without decrypting it or to know the secret (or private) key.

Sensitive information could be subject to regulations, directives and other legislative acts such as General Data Protection Regulation (GDPR) [1] in European Union or Health Insurance Portability and Accountability Act (HIPPA) in the United States of America which establishes a list of 18 indicators [2] that must be taken into account when processing medical data.

Encryption and signature schemes make use of heavy computations which can reduce the performance and throughput in favor of data security. These operations can be accelerated by using parallel computing architectures like Graphics Processing Units (GPUs), Field Programmable Gate Arrays (FPGA) or only by using specific schemes with implementations in Application Specific Integrated Circuits (ASIC).

Along with the fast technological growth we can observe that most of workstations, even portable devices, are equipped with a GPU which is mainly used for graphical processing. By using those GPU's there could be a significant improvement in speeding up the encryption and decryption of data, which takes place for every packet that is going inbound or outbound.

This paper focuses on analyzing the main theoretical benefits brought by utilizing the GPU in securing users' personal data. We compared raw performance achieved by using traditional implementation based on CPU technologies with implementations developed using CUDA and OpenCL technologies, which are the most widespread frameworks that harness GPU's capabilities.

The rest of this paper is organized as follows: first, we present in Section 2 the related work on this area of research. Then, Section 3 introduces the implementations details of the proposed solution, and in Section 4 we present an analysis of the experimental results. Conclusions about the methods of accelerating different encryption algorithms end our paper.

II. Related Work

There are several papers regarding hardware acceleration of different encryption algorithms using GPU technologies.

In his paper [3], Manavski manages to outperform traditional AES implementations based on CPU and OpenGL by harnessing the state of the art CUDA technology using accessible hardware (Nvidia Geforce 8800). He focuses on accelerating the main cryptography algorithm using the GPU, while keeping the key scheduling process solely for the CPU. The performance gained by using GPU capabilities is rather impressive: a peak throughput rate of 8.28 Gbit/s, making the GPU almost 20 times faster than the CPU. As he concludes, his "effort has to be considered as the proof that the modern unified GPU architecture can perform as an efficient cryptographic acceleration board".

One effective use case of the GPGPU technologies is the integration of accelerated algorithms in IPsec VPNs [4]. The authors achieved very good results using CUDA technology in AES-CTR implementation, which is highly parallelizable,

A. MATEI is with the Military Technical Academy „Ferdinand I", Bucharest, Romania (e-mail: adimatei94@gmail.com).

C. LUPAȘCU is with the Military Technical Academy „Ferdinand I", Bucharest, Romania (e-mail: clupascu8@gmail.com).

I. BICA is with the Military Technical Academy „Ferdinand I", Bucharest, Romania (e-mail: ion.bica@mta.ro).

and conclude with a strong recommendation towards this approach as it manages to "highly secure transfer of confidential or sensitive data over IPsec VPN networks".

Another approach on this subject is taken by Ottesen [5] in his master thesis. He developed two different implementations of AES based on CUDA technology. He manages to increase the performance by parallelizing the operations that are done for every block from the input data and also by using lookup tables to combine the SubBytes, ShiftRows, and MixColumns operations that take place into one AES round.

Authenticated encryption used in ciphers like AES is a bit more complicated to parallelize, but the authors of paper [6] separated the AES-GCM implementation in a CPU-GPU manner to precompute counters in order to use them in encryption process. Their proposed technique should allow Gigabit throughput and at the same time minimize the CPU load.

A great work [7] was done by Keisuke Iwai et. al. to elucidate the relation between memory allocation style of AES variables and granularity when using GPUs to accelerate process. Results of their experiments showed that 16 bytes / thread granularity had the highest performance, achieving approximately 35 Gbps throughput. They also demonstrated that granularity and memory allocation differences effects reach around 2%–30% in performance.

III. Solution Design and Implementation

For our project, we took a different approach. We chose not only to compare one encryption algorithm using a single CPU implementation and one GPU technology (CUDA), but we developed an application that uses both CUDA and OpenCL technologies, as well as AES-NI (AES-New Instructions) for Intel processors. We also developed modules for two of the most widespread encryption algorithms: AES and 3DES. Both algorithms are implemented using Counter Mode operation, as it is highly parallelizable, as shown in Table I.

Table I. Parallelization Properties of Different Block Cipher Modes of Operation

Block cipher mode	Parallelizable operation(s)
ECB	Encryption/Decryption
CBC	Decryption
CFB	Decryption
OFB	Not parallelizable
CTR	Encryption/Decryption

From an architectural point of view, the application is quite modular and can be easily modified to be integrated with encryption tools like Veracrypt [8], Dokany [9] and CppCryptFS [10]. As it has been developed in Windows environment using Visual Studio IDE, it is easier to set it up as a library component developed specifically for Windows OS. However, because of the cross-platform advantages brought by CUDA technology and C development language, the source code that can be used for compiling to other operating system is almost identical. The only difference consists of using the appropriate compiling tools for the desired distribution.

Besides the diversity of technologies and implemented encryption algorithms, we have also chosen a wide range of input file sizes so we can precisely decide which implementation is the better choice in a given situation.

Unlike other implementations, ours could offer the best choice in which approach should be used, either the AES-NI (if CPU is capable to execute those instructions) or a framework (CUDA or OpenCL) which accelerates the operation on the graphical processing unit (if available). For the CUDA framework we used the CUDA Toolkit 10.1 provided by NVIDIA, and for the OpenCL we used the Intel SDK for OpenCL™ applications. For the AES-NI implementations we don't need extra frameworks since the CPU instruction set has specific instructions.

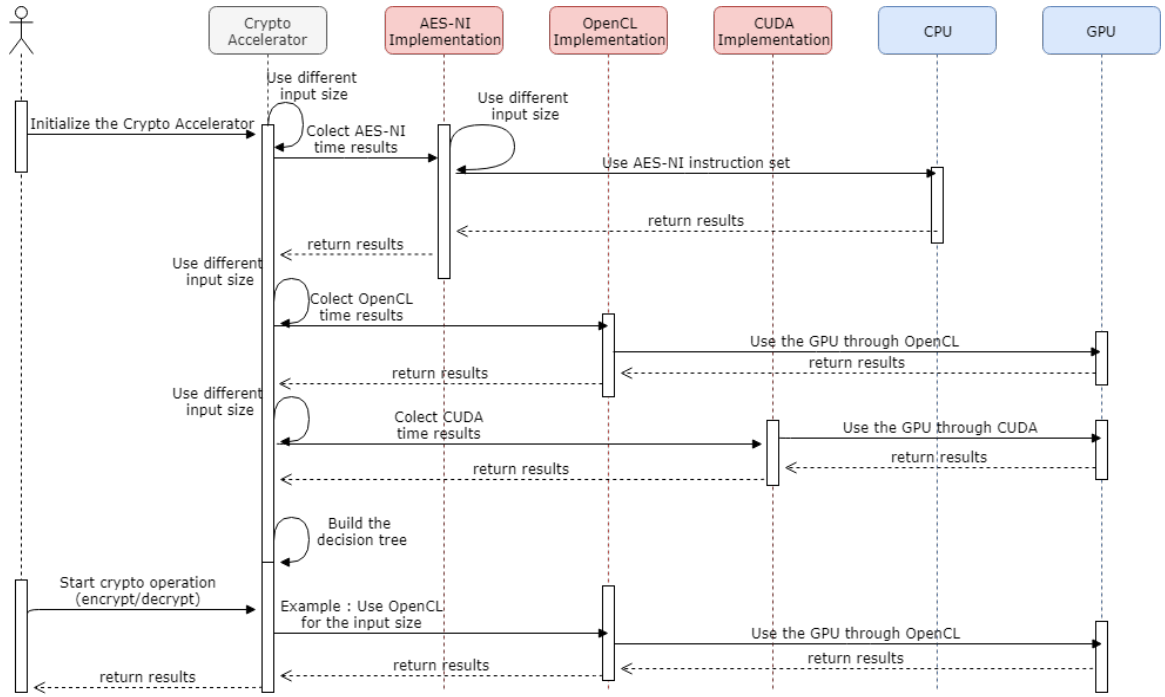

Figure 1. Sequence diagram for initial flow

 Digital Object Identifier 10.32754/JMT.2019.2.04

In order to obtain the maximum speed of encryption/ decryption, we must first use all the implementations on different input sizes to build a decision tree. The way this approach works for an AES operation can be observed in Figure 1.

In terms of architecture and software components, we developed three main pluggable components: OpenCL acceleration API, CUDA acceleration API and for the AES algorithm the AES-NI acceleration API. On top of these components we have the management module for decision making based on input size, this is also the interface to the user.

Beyond those first 3 components we have the frameworks which could be found on both platforms Windows and Linux, meaning: OpenCL & CUDA software development kits and AES-NI instruction set (for Intel processors). Figure 2 presents an overview of this architecture.

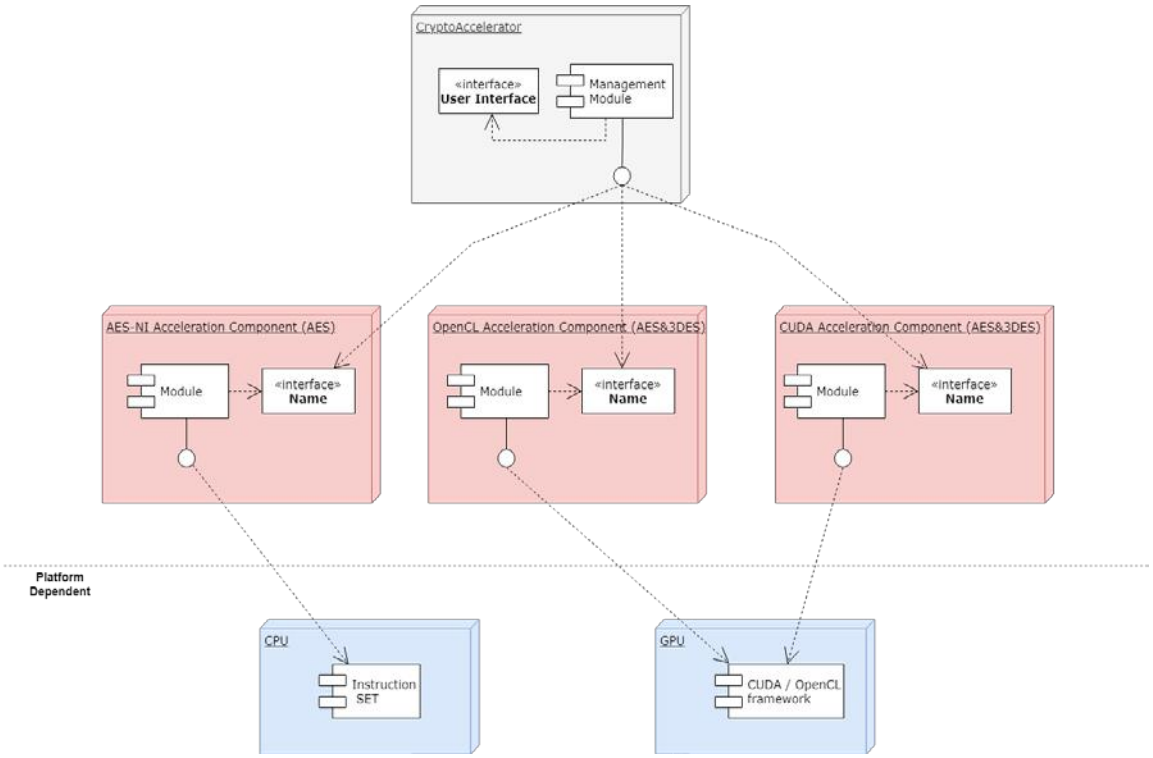

Figure 2. Software components overview

IV. Experimental Results and Analysis

Regarding the platform used for the experiments, its specifications are illustrated in Table II.

Table II. Platform Used for Conducting the Experiments

Specification name	Specification value
CPU	Intel Core i7 6700HQ 2,6 GHz
RAM	24 GB
GPU	Nvidia GeForce GTX 960M
OS	Windows 10 x 64

Due to the fact our approach is to properly and objectively compare the performance gains that GPU technologies can provide in real life situations, we chose a broad range of input file sizes for our tests. The file names and sizes are shown in Table III.

Table III. File Names and Exact Sizes (In Bytes)

File name and size	File size in bytes
100 kB	102,400
500 kB	512,000
6 MB	6,291,456
17 MB	17,825,792
57 MB	59,768,832
120 MB	125,829,120
209 MB	219,152,384

For the results shown in the subsequent tables and figures, the following legend is applied:

- Test files which are named after their sizes in kB or MB (e.g.: 100 kB, 6 MB);
- TIME: the time needed for a test file to be encrypted/decrypted by the algorithm;
- PERF: the performance of the cipher algorithm measured in MB/s, where 1 MB = 1048576 Bytes;
- Median performance: median values of the corresponding computed parameters.

A. AES Implementation results

In Tables III-VI are illustrated the results obtained using the following techniques: CPU implementation (standard implementation, without any optimization - Table IV), CPU implementation with AES-NI technology which uses special SIMD instructions developed by Intel for its microprocessors (Table V), GPU-CUDA (Table VI) and GPU-OpenCL implementations (Table VII).

Digital Object Identifier 10.32754/JMT.2019.2.04

TABLE IV. AES-CTR PERFORMANCE WITH CPU IMPLEMENTATION (NO PARALLELIZATION)

AES Standard							
Test file names and dimensions	Performance parameters	AES-CTR - Key size					
		128		192		256	
		Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
100 kB	TIME (s)	0.04222	0.04977	0.05564	0.06163	0.07350	0.06316
	PERF (MB/s)	2.31319	1.96231	1.75510	1.58459	1.32873	1.54613
500 kB	TIME (s)	0.24788	0.24721	0.27778	0.29127	0.30530	0.33078
	PERF (MB/s)	1.96986	1.97516	1.75778	1.67638	1.59935	1.47615
6 MB	TIME (s)	2.45835	2.40232	3.84219	3.13233	3.65402	3.44482
	PERF (MB/s)	2.44066	2.49759	1.56161	1.91551	1.64203	1.74175
17 MB	TIME (s)	6.82761	6.82092	8.87800	9.01599	10.59189	9.88474
	PERF (MB/s)	2.48989	2.49233	1.91485	1.88554	1.60500	1.71982
57 MB	TIME (s)	24.12111	23.76123	31.78072	31.41706	33.49443	34.41996
	PERF (MB/s)	2.36308	2.39887	1.79354	1.81430	1.70178	1.65602
120 MB	TIME (s)	50.30353	49.94292	62.55041	62.27724	71.54107	71.59196
	PERF (MB/s)	2.38552	2.40274	1.91845	1.92687	1.67736	1.67617
209 MB	TIME (s)	86.37251	86.27355	106.64259	105.77429	125.01828	123.22364
	PERF (MB/s)	2.41975	2.42253	1.95982	1.97591	1.67176	1.69610
Conclusion	Median PERF (MB/s)	2.34028	2.30736	1.80873	1.82558	1.60371	1.64459

TABLE V. AES-CTR PERFORMANCE WITH CPU IMPLEMENTATION (AES-NI SIMD INSTRUCTIONS)

AES-NI							
Test file names and dimensions	Performance parameters	AES-CTR - Key size					
		128		192		256	
		Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
100 kB	TIME (s)	0.00017	0.00023	0.00036	0.00028	0.00034	0.00028
	PERF (MB/s)	585.76411	432.15511	268.97954	346.20791	284.78350	346.20791
500 kB	TIME (s)	0.00130	0.00119	0.00107	0.00157	0.00122	0.00164
	PERF (MB/s)	376.93240	411.16482	454.23075	311.09019	398.95453	297.82200
6 MB	TIME (s)	0.01113	0.01229	0.01281	0.01316	0.01497	0.01703
	PERF (MB/s)	539.13718	488.03001	468.33062	455.82082	400.86292	352.22279
17 MB	TIME (s)	0.03563	0.04328	0.03874	0.03560	0.05171	0.04443
	PERF (MB/s)	477.13873	392.78918	438.80317	477.51465	328.77884	382.60188
57 MB	TIME (s)	0.11358	0.10490	0.14338	0.11945	0.15516	0.14336
	PERF (MB/s)	501.83716	543.39105	397.55099	477.19294	367.37263	397.59372
120 MB	TIME (s)	0.21701	0.21215	0.25709	0.25489	0.30740	0.29472
	PERF (MB/s)	552.97026	565.65066	466.76391	470.79720	390.37032	407.17142
209 MB	TIME (s)	0.37160	0.36894	0.44567	0.44250	0.52155	0.51220
	PERF (MB/s)	562.43160	566.48050	468.95259	472.31203	400.73070	408.04208
Conclusion	Median PERF (MB/s)	513.74449	485.66590	423.37308	430.13368	367.40764	370.23740

TABLE VI. AES-CTR PERFORMANCE WITH GPU-CUDA IMPLEMENTATION

AES CUDA							
Test file names and dimensions	Performance parameters	AES-CTR - Key size					
		128		192		256	
		Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
100 kB	TIME (s)	0.00046	0.00041	0.00052	0.00059	0.00054	0.00054
	PERF (MB/s)	214.62912	235.88466	187.44002	166.64889	181.18043	181.85521
500 kB	TIME (s)	0.00183	0.00182	0.00210	0.00208	0.00229	0.00233
	PERF (MB/s)	267.11228	267.99190	232.84752	234.97654	213.22325	209.47287
6 MB	TIME (s)	0.02086	0.02087	0.02333	0.02333	0.02585	0.0258
	PERF (MB/s)	287.65941	287.56290	257.21267	257.17960	232.14424	232.54011
17 MB	TIME (s)	0.05892	0.05891	0.06599	0.06599	0.07290	0.07300
	PERF (MB/s)	288.52682	288.56600	257.62650	257.62260	233.19296	232.87033
57 MB	TIME (s)	0.19755	0.18008	0.22103	0.20162	0.24335	0.22292
	PERF (MB/s)	288.53163	316.53126	257.88938	282.71706	234.23534	255.69711
120 MB	TIME (s)	0.37898	0.37904	0.42429	0.42429	0.46931	0.46922
	PERF (MB/s)	316.64107	316.58677	282.82609	282.82743	255.69235	255.74412
209 MB	TIME (s)	0.67453	0.66001	0.75497	0.73888	0.83295	0.81726
	PERF (MB/s)	309.84354	316.66331	276.83146	282.86131	250.91482	255.73382
Conclusion	Median PERF (MB/s)	281.84912	289.96954	250.38195	252.11906	228.65477	231.98765

TABLE VII. AES-CTR PERFORMANCE WITH GPU-OPENCL IMPLEMENTATION

AES OpenCL							
Test file names and dimensions	Performance parameters	AES-CTR - Key size					
		128		192		256	
		Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
100 kB	TIME (s)	0.00029	0.00025	0.00030	0.00030	0.00040	0.00035
	PERF (MB/s)	340.26568	384.47343	324.43937	328.80892	242.32320	278.22293
500 kB	TIME (s)	0.00075	0.00073	0.00086	0.00088	0.00100	0.00099
	PERF (MB/s)	649.31017	672.56371	571.08918	553.60686	486.33591	491.72331
6 MB	TIME (s)	0.00650	0.00657	0.00792	0.00784	0.00894	0.00922
	PERF (MB/s)	922.50923	913.24201	757.76711	764.91586	670.91580	650.75922
17 MB	TIME (s)	0.01828	0.01879	0.02194	0.02208	0.02491	0.02511
	PERF (MB/s)	930.02900	904.64027	774.98177	769.96241	682.42945	677.02111
57 MB	TIME (s)	0.05988	0.06138	0.07306	0.07092	0.08305	0.08252
	PERF (MB/s)	951.96740	928.67151	780.22339	803.72250	686.32527	690.74164
120 MB	TIME (s)	0.13236	0.12727	0.14991	0.15267	0.17438	0.17533
	PERF (MB/s)	906.59092	942.88476	800.48563	786.02448	688.15231	684.43147
209 MB	TIME (s)	0.22606	0.22873	0.26165	0.26578	0.30158	0.30698
	PERF (MB/s)	924.52922	913.72511	798.78615	786.36762	693.00759	680.83055
Con-clusion	Median PERF (MB/s)	803.60023	808.60011	686.82466	684.77267	592.78422	593.39003

The results obtained from these tables can be summarized in two smaller tables that show the performance increase recorded by using GPU technologies instead of traditional CPU implementations (Tables VIII and IX):

TABLE VIII. AES GPU PERFORMANCE INCREASE OVER THE AES STANDARD IMPLEMENTATION

Implementation	Average performance increase (in Times)
AES-CUDA-128	x123.034
AES-CUDA-192	x138.266
AES-CUDA-256	x141.810
AES-OpenCL-128	x346.886
AES-OpenCL-192	x377.402
AES-OpenCL-256	x365.167

TABLE IX. GPU PERFORMANCE INCREASE OVER THE AES-NI IMPLEMENTATION

Implementation	Average performance increase (in Times)
AES-CUDA-128	x0.572
AES-CUDA-192	x0.589
AES-CUDA-256	x0.624
AES-OpenCL-128	x1.613
AES-OpenCL-192	x1.607
AES-OpenCL-256	x1.608

It must be noted that the average performance increase was calculated using the formula:

(AES_GPU_Encryption_Median_Perf + AES_GPU_Decryption_Median_Perf) / (AES_Serial_Encryption_Median_Perf + AES_Serial_Decryption_Median_Perf),

where AES_Serial term implies AES Standard implementation for Table VIII and AES-NI implementation for Table IX.

 Digital Object Identifier 10.32754/JMT.2019.2.04

B. 3DES implementation results

In Tables X-XII are illustrated the results obtained using the following techniques: CPU implementation (standard implementation, without any optimization - Table X), GPU-CUDA (Table XI) and GPU-OpenCL implementations (Table XII).

TABLE X. 3DES-CTR PERFORMANCE WITH CPU IMPLEMENTATION

3DES			
Test file names and dimensions	Performance parameters	3DES-CTR	
		192	
		Encryption	Decryption
100 kB	TIME (s)	1.10226	1.03509
	PERF (MB/s)	0.08860	0.09435
500 kB	TIME (s)	5.10073	5.12299
	PERF (MB/s)	0.09573	0.09531
6 MB	TIME (s)	63.38538	63.44384
	PERF (MB/s)	0.09466	0.09457
17 MB	TIME (s)	180.51405	179.15294
	PERF (MB/s)	0.09418	0.09489
57 MB	TIME (s)	584.55898	581.86910
	PERF (MB/s)	0.09751	0.09796
120 MB	TIME (s)	1236.52006	1232.26014
	PERF (MB/s)	0.09705	0.09738
209 MB	TIME (s)	2173.57387	2182.60639
	PERF (MB/s)	0.09616	0.09576
Conclusion	Median PERF (MB/s)	0.09484	0.09575

TABLE XI. 3DES-CTR PERFORMANCE WITH GPU-CUDA IMPLEMENTATION

3DES CUDA			
Test file names and dimensions	Performance parameters	3DES-CTR	
		192	
		Encryption	Decryption
100 kB	TIME (s)	0.00111	0.00109
	PERF (MB/s)	88.29679	89.67516
500 kB	TIME (s)	0.00480	0.00484
	PERF(MB/s)	101.74646	100.86372
6 MB	TIME (s)	0.05884	0.05889
	PERF (MB/s)	101.98011	101.88487
17 MB	TIME (s)	0.16441	0.15246
	PERF (MB/s)	103.40192	111.50466
57 MB	TIME (s)	0.51064	0.51112
	PERF (MB/s)	111.62397	111.52089
120 MB	TIME (s)	1.07564	1.07554
	PERF (MB/s)	111.56128	111.57176
209 MB	TIME (s)	1.87275	1.87233
	PERF (MB/s)	111.60077	111.62556
Conclusion	Median PERF (MB/s)	104.31590	105.52094

TABLE XII. 3DES-CTR PERFORMANCE WITH GPU-OPENCL IMPLEMENTATION

3DES OpenCL			
Test file names and dimensions	Performance parameters	3DES-CTR	
		192	
		Encryption	Decryption
100 kB	TIME (s)	0.00045	0.00046
	PERF(MB/s)	216.05365	212.75871
500 kB	TIME (s)	0.00217	0.00217
	PERF (MB/s)	225.32591	225.32591
6 MB	TIME (s)	0.02631	0.02636
	PERF (MB/s)	228.05017	227.58307
17 MB	TIME (s)	0.07459	0.07494
	PERF (MB/s)	227.90648	226.85117
57 MB	TIME (s)	0.25056	0.25112
	PERF (MB/s)	227.49315	226.98221
120 MB	TIME (s)	0.52587	0.52530
	PERF (MB/s)	228.19458	228.44176
209 MB	TIME (s)	0.91240	0.92480
	PERF (MB/s)	229.06745	225.99505
Conclusion	Median PERF (MB/s)	226.01306	224.84827

The summary of obtained results regarding 3DES GPU performance increase over CPU implementation is shown in Table XIII.

TABLE XIII. 3DES GPU PERFORMANCE INCREASE OVER THE 3DES STANDARD IMPLEMENTATION

Implementation	**Average performance increase (in Times)**
3DES-CUDA-192	x1101.019
3DES-OpenCL-192	x2365.681

The average performance increase is calculated similarly to the formula used for AES implementations:
(3DES_GPU_Encryption_Median_Perf + 3DES_GPU_Decryption_Median_Perf) / (3DES_Standard_Encryption_Median_Perf + 3DES_Standard_Decryption_Median_Perf).

V. CONCLUSIONS AND FUTURE WORK

The main contribution of this paper is to evaluate different approaches in order to compare encryption/decryption times for the widely used symmetric algorithms. Because protection of data is very important we tried to emphasize that there are some potentially solutions to shorten the delays related to cryptographic operations when information is being encrypted (decrypted). The results clearly underline the fact that even with a desktop or a laptop one could use the GPU to accelerate the long taking task of encryption. Using OpenCL framework, the AES encryption/decryption is approximately 360 times faster than sequential CPU implementation and around 140 times faster with CUDA framework.

As a future work we will try to parallelize the Galois Counter Mode (GCM) and XEX-based tweaked-codebook mode with ciphertext stealing (XTS) used in data at rest encryption. Secondly, we plan to integrate this work into live hard disk encryption solutions such as VeraCrypt [8], Dokany [9] and CppCryptFS [10]. The integration could be made by writing a custom driver using the main interface of our implementation, since our solution is pluggable.

Last but not least we will integrate all the developed methods into a solid implementation which could choose, depending on the input size and other computer characteristics, what method is more efficient to be used in order to get the lowest encryption/decryption times.

REFERENCES

[1] Regulation 679/2016 of The European Parliament and of the Council of 27 April 2016, GDPR Regulation, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN

[2] HIPAA PHI: List of 18 Identifiers and Definition of PHI, https://www.atlanta.va.gov/Docs/HIPAA_Identifiers.pdf

[3] Svetlin A. Manavski, CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography; 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007). doi: 10.1109/ICSPC.2007.4728256

[4] Heinemann, Colleen, Sai Shankar Chaduvu, Adam Byerly and Alexander Uskov. "OpenCL and CUDA software implementations of encryption/decryption algorithms for IPsec VPNs." 2016 IEEE International Conference on Electro Information Technology (EIT) (2016): 0765-0770. doi: 10.1109/EIT.2016.7535336

[5] Alexander Ottesen, Efficient parallelisation techniques for applications running on GPUs using the CUDA framework.

[6] Georg Schönberger, Jürgen Fuß, GPU-Assisted AES Encryption Using GCM, Bart Decker; Jorn Lapon; Vincent Naessens; Andreas Uhl. 12th Communications and Multimedia Security (CMS), Oct 2011, Ghent, Belgium. Springer, Lecture Notes in Computer Science, LNCS-7025, pp.178-185, 2011, Communications and Multimedia Security. doi: 10.1007/978-3-642-24712-5_16

[7] Keisuke Iwai, Naoki Nishikawa and Takakazu Kurokawa, Acceleration of AES encryption on CUDA GPU, International Journal of Networking and Computing, Volume 2, Number 1, pages 131–145, January 2012. doi: 10.15803/ijnc.2.1_131

[8] https://www.veracrypt.fr/code/

[9] https://github.com/dokan-dev/dokany.

[10] https://github.com/bailey27/cppcryptfs.