2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems

# Distributed Platform for the Analysis of Cryptographic Algorithms

Vlad-Cosmin Ozunu, Cezar-Costin Pîrvu , Cătălin Leordeanu, Valentin Cristea
*Faculty of Automatic Control and Computers, University 'Politehnica' of Bucharest, Romania*
*Email: vlad.ozunu@cti.pub.ro, cezar.pirvu@cti.pub.ro catalin.leordeanu@cs.pub.ro, valentin.cristea@cs.pub.ro*

***Abstract*—Data protection and information security have always been intricate problems of the majority of software applications which have been deployed throughout the Internet. Consequently, a substantial effort has been put into the creation and development of a wide variety of solutions to tackle this very issue.The aim of this paper is to offer a means of performance measurement and security validation for some of the encryption algorithms which are extensively used in todays industry (DES, 3DES, AES, etc.), as well as some hash functions.Therefore, the evaluation platform takes on the above mentioned algorithms from two very divergent perspectives: one of them focuses mainly on CPU vs. GPU performance issues, whereas the latter tackles the problem of randomness of the encrypted results by comparison to several strict criteria. In order to achieve these goals, the platform provides a graphical user interface which eases interactions such as: tests selection, worker attachment or removal, events logging, input provision and output analysis. The proposed solution represents an evaluation platform that performs a wide range of tests on hash and symmetric key algorithms in order to deduce their performance and behavior on multiple architectures, as well as various NIST tests, on different environments.**

## I. Introduction

It is an undeniable fact that the IT industry has significantly evolved throughout the last couple of decades, not only due to the great increase in computational power of the machines, but also due to human ingenuity in developing software applications which have had a great impact on the very core of todays society by revolutionising even the way we interact. Nevertheless, in the context of an ever-changing industry, one point of emphasis which has persisted throughout this entire development period is represented by security, or to be more exact, data protection when being exchanged over the Internet. To solve this issue, the most usual solution is to use an encryption algorithm in order to try to prevent mischievous attacks from succeeding.

The aim of the proposed solution is to emphasise only on the characteristics of symmetric key algorithms. To be more specific, the evaluation platform focuses on providing a variety of tests which are based around two of the most widely encountered and intensively used algorithms of this type in the actual IT industry: 3DES(Triple DES) and AES. Since these algorithms are so significant in the context of the present industry, it is undeniable that such a platform, which has the clear purpose of not only evaluating their performances, but also validating the cryptograms which they produce, is extremely necessary.

Moreover, as mentioned above, a special emphasis has been put on the development of performance tests (CPU and GPU), which measure the time necessary for a machine (local or remote) to encrypt or decrypt a series of character sets. The platform takes into account not only the total encryption/decryption time, but also the individual time necessary for the conversion of a chunk of the input, feature which has offered the possibility of tracing charts for a graphical representation of the results. These graphs provide an insight on both the average run time and also the spikes registered throughout the alteration of the input. We compared different implementations of the same algorithms such as the Java libraries LibCrypto (javax.crypto)[9] and Bouncy Castle, as well as other more complex solutions such as John the Ripper.[8]

An evaluation platform of this type is necessary, as it can be used as a means of benchmarking the performances of encryption algorithms on both CPU and GPU architectures. Another role which can be successfully fulfilled by this platform is represented by the discovery of vulnerabilities present in some implementations of the algorithms, making them less prone to attacks in the nearby future. In other words, the evaluation platform can provide helpful observations through its results, which can set the stage for the next generation of cryptographic algorithms.

## II. Related work

Testing and evaluation of cryptographic algorithms is not a new field. In an era when there are many research groups dedicated to cryptology, this is a very dynamic field. The interval between the publication of a new cipher and the time when attacks are developed for it is becoming shorter and shorter. Also, organizations such as the National Institute of Standards and Technology (NIST) are helping by publishing standards and official test suites for such algorithms [4] [5].

A general characteristic of most block ciphers is the fact that most allow parallel execution with very little overhead. This lead to the search for different hardware platforms, in order to gain speed or energy efficiency. One such approach is COPACOBANA (Cost-Optimized Parallel COde Breaker) [6]. Their solution was based on a modular structure of FPGAs. Due to the fact that a cryptographic algorithm is usually based on the execution of multiple rounds, with little or no communication inside a single round, it can be mapped to a simple FPGA architecture, similar to a

Spartan3. Similarly to the solution proposed in this paper, the COPACOBANA architecture can be used for multiple applications for encryption or hash functions. It can be used for the evaluation of algorithms such as DES, AES or SHA-1 or the testing of different attacks on these algorithms [7]. However, their approach is focused on the use of FPGA-based architectures and it cannot be easily adapted for other types of hardware, such as GPUs.

Apart from this approach, there are other research papers oriented towards the use of cryptographic algorithms on different hardware platforms. AES has received a lot of attention, of course [1] [3]. This paper however is focused on the evaluation architecture, more than the cryptographic algorithms themselves.

## III. Solution architecture

This section is entirely dedicated to presenting the architecture which has been constructed in such a manner that it can support the functionality of an evaluation platform of the performances of several cryptosystems.

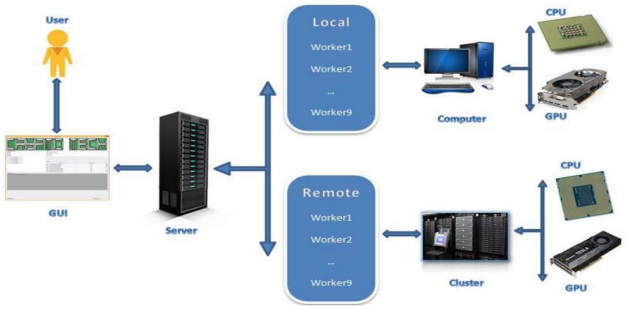


Figure 1. Architecture of the evaluation platform

### A. Architecture components

As it can be observed from figure 1, the entire architecture of the evaluation platform is based around the functionality of the server. The server is basically the central structural element as it mediates the communication between the user and all of the selected clients (workers). Therefore, it can be stated that the infrastructure of the evaluation platform relies on this client-server interaction, as it offers the possibility of both local and remote tests execution.

*1) Server:* The first module we will concentrate on is the server, as it represents the principal component of the entire architecture due to the functions it manages to carry out.

When a client tries to connect to the platform, the server creates a special structure entirely dedicated to handling all the communication aspects with the respective worker. The reason why there is a need for such a structure relies on the fact that the server must be able to simultaneously manage not only the newly detected inbound connections, but also the previously established ones.

Before being permitted to run tests on it, a client must first off be registered through a process which implies the transmission of some important information or relevant characteristics. The attributes provided by the client will become relevant in the context in which a test selection process is in order. The server also offers keep-alive mechanisms with both idle and busy clients, as it is extremely important to be able to determine whether a malfunction has appeared and the worker cannot be considered a viable candidate for test execution any longer.

In addition to all of these characteristics, the server is also tightly connected to the GUI (Graphical User Interface), as it takes the input offered by the user, processes it by detecting the tests and their inputs and transmits it only to the selected workers. Likewise, after the respective series of tests have been successfully executed, the server is the component responsible for updating the GUI, more specifically, announcing the graphical component that the workers have finished their tasks and they are now once again eligible for further analysis.

The server also offers the possibility of worker deregistration if the user decides that the respective client is no longer of use and no further testing is necessary. Another reason for the implementation of such a functionality may be represented by the situation in which the worker keeps producing results, but the measurements provided do not comply with certain, pre-verified, expectations.

Finally, another function worth mentioning refers to the fact that the server is the one which computes the MAC address of the local platform on which it is executed, meaning that it offers the support necessary for the other elements to determine whether they should fetch (through SCP transfer) the results from the remote workers or not.

*2) Client/Worker:* Despite having two types of performance tests (CPU-based and GPU-based), an important observation refers to the fact that the client remains generic, as it basically represents an abstraction of the underlying architecture it is executed on. Therefore, we cannot state that the clients are necessarily only composed out of CPUs or GPUs, but, in fact, they can be seen as standalone platforms of any type.

In terms of the offered functionality in the context of the evaluation platform, the clients are the other endpoint of the communication protocol and are mainly characterised by having a name, a type of architecture, a short description provided to the user and a chosen port. Moreover, a MAC address is also computed for the client, as it is the criteria for determining the location of the worker.

After a client is successfully registered to the evaluation platform, it awaits commands from the server, commands based on which the entire behaviour of the worker is modelled. For instance, when a client is selected to be a test subject, it immediately processes the command and passes the arguments to a work handler-type structure where

the tests are going to be executed. The reason why such an approach is necessary refers to the fact that the client must be able to continuously respond to the keep-alive messages transmitted by the server. This behaviour will be also presented in a more detailed manner in the following chapters, when more specific implementation details will be offered.

*3) Work handler:* An auxiliary work handler structure is needed in order to ensure the functionality of the evaluation platform. In the context of the platform architecture, this component is placed at client level and has the significant role of determining the selected tests and the arguments the user has provided for them.

After it has parsed these pieces of information, the handler takes each test, determines the type of the respective test (performance or validation) and then, executes it with the required set of input data. In other words, the work handler structure works as a dispatcher, as it only calls certain methods from several classes depending on the chosen test and the provided input. This mechanism is repeated for all of the selected tests and ends with a notice sent to the client, this notification propagating itself all the way back to the server which then updates the GUI accordingly.

### *B. Communication protocol*

The client-server communication protocol is basically the foundation of the entire infrastructure needed for the design of the evaluation platform. From a structural point of view, the entire communication protocol can be easily split up into three phases: registration, tests execution intertwined with the keep-alive mechanisms and lastly, deregistration. In the following paragraphs, these stages will be presented in great detail and also in the order in which a client would subsequently encounter them throughout all of the interactions with the evaluation platform.

First of all, when the client desires to connect to the platform, it sends a REGISTER request containing its name (identifier) to the server. However, before any registration can be realised, the server must also be informed about the port selected by the client. Based on the clients chosen port and name, the server generates its UID (User ID), which is then sent back to the respective worker. Finally, the client must confirm that it has received its server-level identifier by sending an acknowledgement also containing supplementary information (type, MAC address and a short description to be provided in the GUI to the user).

Furthermore, the presentation of the second step of the communication protocol must commence with the explanation of the keep-alive mechanisms, as they are ubiquitous throughout this entire stage regardless of the fact that the client is executing tests or not. The server has the role of initiating the process through a PING message, while the client must acknowledge its reception by sending an ACK. In order to prevent the network from flooding, a PING timeout

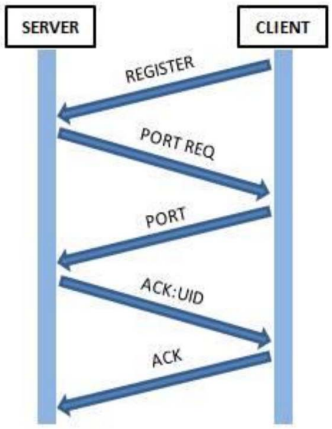


Figure 2. Registration phase

is used which, in this case, has been set to one second. However, if an ACK message (sent by the worker) is lost, the server will continue to dispatch PING messages up to a maximum of 20 retries (that have not been followed by a response). In this case, the client will be considered defective and therefore, strict measures are in order: the connection with the respective worker will be destroyed and the GUI will be updated so that the user will not submit tests to a faulty component. Yet, if the client manages to respond with an ACK before the number of maximum retries is reached, then the server will reset its timer and the process will be repeated after the timeout period has expired. These mechanisms are precisely depicted in figure 2.

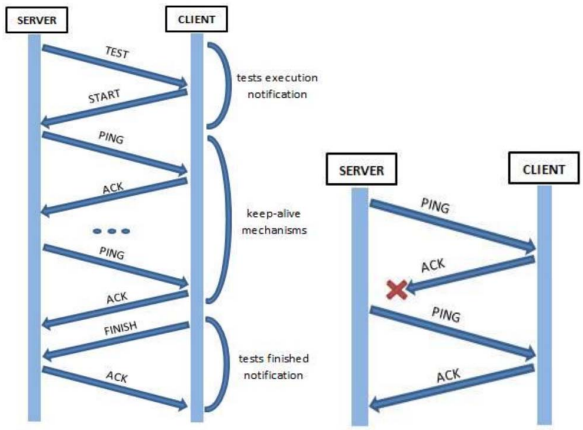


Figure 3. Tests execution and keep-alive mechanisms

Moreover, when a series of tests are selected from the GUI and launched into execution, the server immediately notifies all of the preferred clients by sending a TEST message containing the necessary information (test names and arguments). Next, the worker creates the work handler structure which will manage the correct execution of the

tests and if the initialisation of the respective structure is successful, then the client will reply with a START message to the server.

In a similar fashion, when all the tests are successfully executed by a client, it must inform the server about this situation and therefore, a FINISH message is used. When receiving this notice, the server will update the GUI, as it needs to represent the respective client as an available worker for further tests execution. After this step is successfully carried out, the server will then send an acknowledgement back to the respective client so that the worker will know that it can now proceed to destroying the work handler structure.

Lastly, the final component of the communication protocol, the deregistration phase, is always initiated by the server. More specifically, a client will be unregistered only when it is marked for removal by the user by means of the GUI. The case when the client stops responding to the PING messages and the maximum number of retries is reached is a different type of deregistration, as the server will not inform the worker about its actions. These different approaches (soft and hard deregistration) have been chosen in order to create a clear distinction between the case in which the client is removed as a result of the users interaction with the GUI and the situation when the worker is unregistered due to its inactivity (without user intervention).

## IV. Test scenarios

Throughout this section we will emphasise on the types of tests which have been integrated into the context of the entire evaluation platform without neglecting the results and observations obtained by means of their execution. Therefore, brief descriptions of the tests will be provided along with the manner in which certain measurements were computed. Moreover, where further analysis is needed, plots of the obtained results will be presented and observations will be offered based upon the respective graphs.

Performance tests are based on the idea of measuring the time needed for some cryptographic operations to be executed: encryption or decryption. Moreover, several implementations of these cryptosystems were used in the development of these tests, meaning that a great variety of results were obtained. These results will be presented in the following subchapters, by taking also into consideration the platform on which the experiments were performed (CPU or GPU).

The first performance tests we performed are dictionary based. The pieces of data that the experiments are required to convert come in the form of a dictionary, with each word submitted for alteration being placed on a separate line of the input file.Furthermore, it is important to state the manner in which such a test is executed at worker level. Each word is taken from the input file, a transformation is applied to it (encryption, decryption) and its duration is precisely measured. These individual measurements will be redirected to an output file, along with the results of the cryptographic conversions. Additionally, based on all of the computed durations, plots will be generated that will enable the observation of both the average time necessary for a transformation and also eventual load spikes which could be encountered.

Taking all of these initial pieces of information into consideration, the first sets of CPU tests which will be presented are based around the two previously described Java libraries: javax.crypto and Bouncy Castle. These experiments aimed to uncover the manner in which the provided implementations of the cryptosystems behaved when subjected to heavy load. Therefore, a substantial dictionary was used in this case, consisting of about 350 000 words, all of them being taken from the English language.

As it can be observed from the figures, the AES implementations provided by the two libraries have slightly different behaviours when subjected to the same input file and to similar execution conditions (the key size was also equal). What is more, the displayed tests were completed by using a local worker, but similar results were obtained when running the same experiments on the remote cluster.

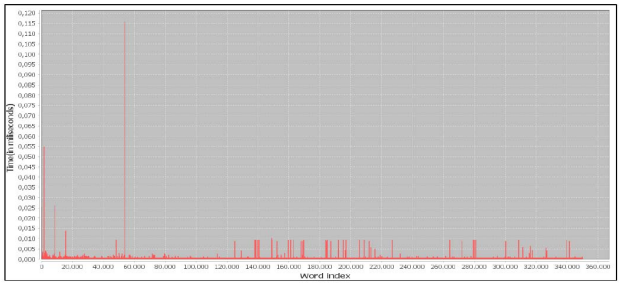


Figure 4. AES Bouncy Castle Dictionary test

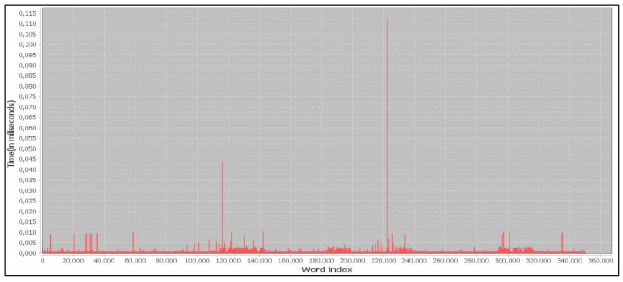


Figure 5. AES javax.crypto Dictionary test

At first glance, the Bouncy Castle implementation seems more stable, as in most circumstances, measurement spikes take the form of single irregularities and do not combine into clusters of inconsistent peaks. Stated differently, the measurement spikes are distributed in a scarcer manner than in the case of the javax.crypto. However, when it

comes to the case of the average encryption duration, the obtained results have shown that they are comparable, both implementations being able to surpass its counterpart in some experiments. It is also important to mention that these differences emerge only for sufficiently large test samples.

However, the 3DES implementations provided by the two libraries did not follow the same pattern as in the case of AES, more contrasting results being observed. In this situation, the Bouncy Castle library managed to obtain a significantly lower average encryption time. The explanation behind the 3DES results lies in the manner in which spikes are formed. Since the values of these peaks revolve around similar figures in both cases, the reason for its much higher performances must rely on the fact that fewer spikes are obtained. In other words, the Bouncy Castle implementation seems to display a higher resistance to this kind of irregularities, meaning that it is the more reliable structure out of the two.

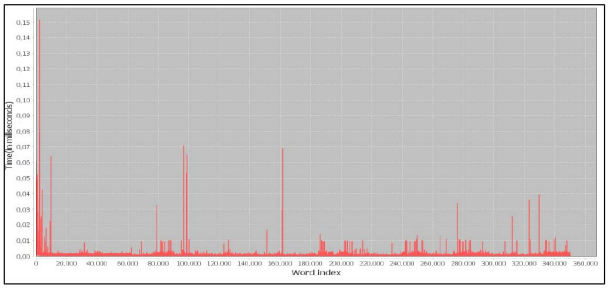


Figure 6. 3DES Bouncy Castle Dictionary test

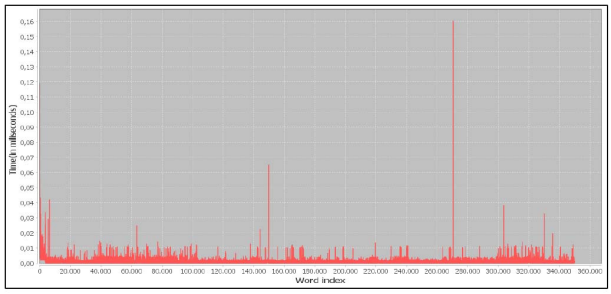


Figure 7. 3DES javax.crypto Dictionary test

The second set of performance tests which will be presented throughout this section are represented by the ones implemented by means of the John the Ripper project. These experiments will receive as input a list of DES-encrypted words and the measurements will try to depict the time markers when certain ciphers from the file are decrypted successfully. These guess durations will be computed in relation to the initial moment when the experiment was begun. Moreover, the tests are characterised by a lower precision when it comes to estimating each of the separate obtained results. Therefore, all measurements which are completed within a second will be considered to have been finished at the same time (at the beginning of the respective second). For instance, if several words are guessed between seconds one and two, then all of them will be redirected to the output file as having a time marker of one.

In addition to this, since all the encrypted words will be guessed by the project without the help of any external dictionary, the total runtime of an experiment can very easily reach significant durations. For this reason, the selection of input data cannot be as broad, vast as in the case of the previously presented performance tests.

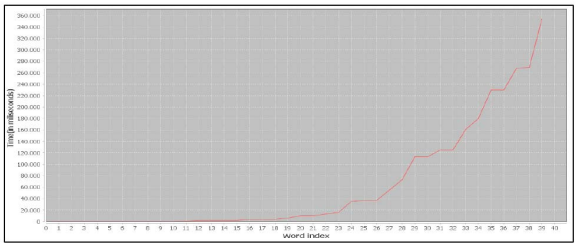


Figure 8. John the Ripper CPU results

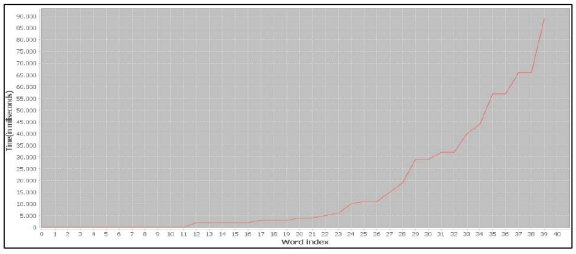


Figure 9. John the Ripper GPU results

The results presented in figures 8 and 9 were obtained by executing these type of performance tests on a remote cluster. Furthermore, a homogenous dictionary was utilised to generate these plots, as only six-letter long words were encrypted and used to build the input file. The words were also chosen so that they could cover as much of the permutation space as possible, enabling the observation of certain differences between the hardware platforms the experiments were executed on.

The first observation which can be made when looking at the obtained plots refers to the fact that both the CPU and GPU results seem to follow an almost identical behavioural pattern. The time markers illustrated throughout these figures depict proportional increases in duration, meaning that the graphs are very much alike. However, when comparing the total runtime of the experiments, a significant discrepancy arises. The GPU platform manages to guess all the encrypted words in the same time that it takes the CPU to correctly determine only about 60% of them.

Moreover, the processor seems to be able to size up to the performances of the graphical unit only at the beginning of the test. This observation conveys the idea that the differences between the two platforms will not be as significant for smaller experiments. Therefore, a conclusion emerges: the larger the runtimes of the experiments are, the greater the gap in duration will be obtained.

## V. Conclusions and future work

By taking into consideration all of the results which were presented in the previous chapter, it can be stated that the evaluation platform has definitely fulfilled its goal of offering a means of measuring, quantifying the characteristics of the cryptographic algorithms. Furthermore, since the experiments were an unquestionable point of focus in the development process of the platform, the measurements obtained by means of the application have supplied a wide range of perspectives on the behavioural patterns displayed by the cryptosystems when subjected to different execution conditions.

The performance tests which have been integrated into the functionality of the evaluation platform have illustrated that when handling and cryptographically processing sufficiently large amounts of data, the GPU architecture is the most suitable candidate for this exact situation. Otherwise, the time needed for the initialisation of the GPU makes it inapplicable for this type of tests and therefore, better (or comparable) results will be obtained by a standard execution on the processor.

Moreover, when comparing the performances of the two symmetric key algorithms which were selected for thorough analysis, an important observation has to be specified. The AES cryptosystem has proven that it is not only faster, but also more flexible than the 3DES encryption algorithm, meaning that this cipher is the most suitable out of the two when it comes to handling large volumes of data. Similar characteristics can be deduced when comparing the performances of the Bouncy Castle cryptographic project to the implementations of the javax.crypto library. In the case of the AES cryptosystem, the javax.crypto package seems to offer the most well-rounded implementation in terms of reliability and adaptability.

Due to the infrastructure which the entire application was built on, the evaluation platform can be very easily extended especially when it comes to the idea of adding new testing methods, or even modules. New experiments based on the already integrated cryptographic algorithms can be added in order to offer an even more accurate assessment of their characteristics. For instance, collision resistance experiments can be added to the functionality of the evaluation platform, therefore offering a more precise, thorough analysis of the security provided by the cryptosystems. Furthermore, new cryptographic algorithms can be linked into the project by means of the already incorporated Java-based libraries, which have been presented in great detail in the previous sections. By displaying such a great variety of test subjects, the evaluation platform can become an even more efficient analysis tool.Additionally, since the communication protocol is an intricate part of the functionality of the entire application, a point of focus should be represented by the improvement of this particular mechanism. An optimisation, could be obtained by reducing the number of messages which are exchanged between the server and the clients, meaning that the communication lines will seem less congested.

## Acknowledgment

The work has been supported by the "*Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds* " through the Financial Agreement POSDRU/159/1.5/S/ 134398 and the project Data4Water: Excellence in Smart Data and Services for Supporting Water Management, Project number 690900/H2020-TWINN-2015.

## References

[1] Hamalainen, Panu, Timo Alho, Marko Hannikainen, and Timo D. Hamalainen. "Design and implementation of low-area and low-power AES encryption hardware core." In Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on, pp. 577-583. IEEE, 2006.

[2] Moon, Dukjae, Kyungdeok Hwang, Wonil Lee, Sangjin Lee, and Jongin Lim. "Impossible differential cryptanalysis of reduced round XTEA and TEA." In Fast Software Encryption, pp. 49-60. Springer Berlin Heidelberg, 2002.

[3] Feldhofer, Martin, Johannes Wolkerstorfer, and Vincent Rijmen. "AES implementation on a grain of sand." IEE Proceedings-Information Security 152, no. 1 (2005): 13-20.

[4] Murphy, Sean. "The power of NISTs statistical testing of AES candidates." Preprint. January 17 (2000).

[5] Soto, Juan. "Randomness testing of the AES candidate algorithms." NIST. Available via csrc. nist. gov (1999).

[6] Kumar, Sandeep, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. "Breaking ciphers with COPACOBANAa cost-optimized parallel code breaker." In Cryptographic Hardware and Embedded Systems-CHES 2006, pp. 101-118. Springer Berlin Heidelberg, 2006.

[7] Gneysu, Tim, Gerd Pfeiffer, Christof Paar, and Manfred Schimmler. "Three Years of Evolution: Cryptanalysis with COPACOBANA." In Workshop record of SHARCS. 2009.

[8] Lim, Ryan. "Parallelization of John the Ripper (JtR) using MPI." Nebraska: University of Nebraska (2004).

[9] Nambiar, Vishnu P., Mohamed Khalil-Hani, and Muhammad M. Zabidi. "Accelerating the AES encryption function in OpenSSL for embedded systems." International Journal of Information and Communication Technology 2.1-2 (2009): 83-93.