Sampling methods for cryptographic tests

George Marinakis [*]

Abstract

Modern cryptographic algorithms have an enormous key diversity, so if we want to test their strength for all the keys, it will take practically an infinite time. To avoid this, we use the sampling method, in which we examine a much smaller number of keys *n* and then we make estimation for the total key population *N* with a predetermined sampling error. For the generation of the *n* cipher outputs (samples) with the *n* corresponding keys, the critical questions are how many samples we will test and how large must be the size of each sample. The general rule is that, the sampling error is reduced as we increase the number of the samples. But since the tests must be executed in an acceptable time, we must compromise the above rule with some additional factors, such as the type of the cryptographic cipher, the kind and the size of the plain information and of course the available computer power. In this study we examine the interrelations of all the above factors, and we propose applicable solutions.

Keywords: Cryptography, Data encryption, Communication security, Computer security, Data security, Information security.

1. Introduction

When we want to evaluate the cryptanalytic strength of cryptographic algorithms, we must perform some specific tests on their output sequences (keystreams) as well as on selected encrypted texts. These tests investigate the complexity and the non-linearity of the algorithms and are based on specific statistical and cryptanalytic methods, which test the randomness and the uniform distribution in the bits of the keystreams, as well as unwanted similarities between the different keystreams. They also check the independence between the clear and the encrypted text and they generally search for security weaknesses which can be used for cryptanalytic attacks. Some researchers have developed tests suites for random number generators and cryptographic algorithms, like these which are described in [1], [2], [3], [4]. Each of these test suites contains several statistical tests, which can investigate the existence of weaknesses in cryptographic algorithms.

For the generation of the output sequences, we must use a software simulation of the under-test algorithm, which should be able to generate binary unformatted outputs for all possible combinations of the cryptographic keys. The simulation software besides the capability of the selection and modification of the keys, must also enable the user to select the length of the cipher output bitstream.

[*] George Marinakis holds a MS in Electrical Engineering from University of Patras (Greece) and a PhD in Cryptography from National Technical University of Athens (NTUA). He is a former professor at Telecommunications and Electronics School of Signal Officers (Athens, Greece). He is currently instructor and scientific collaborator at Hellenic Army Academy. He can be reached at gmari@tee.gr.

The methods which we will propose are based on symmetric cryptographic algorithms, but similar methods can be applied in asymmetric cryptographic algorithms.

2. Testing all the keys

As we noted, for each test key of the cryptographic algorithm we have to produce an output sample and apply the tests to it. If we want to test all the keys, due to the very large size of cryptographic keys (which today varies between 2^{128} to 2^{256}), it is practically impossible to test all the combinations of the key. In order to show this, we constructed Table 1 in which we calculated the total time for the production of samples for all the keys, for the block cipher AES, for its three choices of key values (90, 128, and 256 bits).

According to [5] , if we use software implementation for a block cipher, the time needed for the production of one block is given by equation (1), where T_{MDL} is the time for a Main Decryption Loop (MDL), C_{MDL} is the CPU cycles for a MDL and F is the frequency of CPU clock. If in equation (1) we put C_{MDL} = 167 cycles/block which is the higher speed of AES software implementation that we found in literature [6] and F=3,6 GHZ as the currently higher CPU clock speed, we obtain that T_{MDL} = 46,388 nsec.

$$\mathrm{T_{MDL}} = \frac{\mathrm{C_{MDL}}}{\mathrm{F}} \quad \textbf{(1)}$$

Also according to [5] , if we use hardware simulation for a block cipher, the production time for one block, which is called latency, is given by equation (2). Where B_{size} is the number of the input block bits and $T_{hroughput}$ is the number of encrypted/decrypted bits/s (throughput) for single hardware implementations in FPGA or ASIC, which can reach 200 Gbps according to [7]. If in equation (2) we will put B_{size} = 128 bits and $T_{hroughput}$ = 200 Gbps, we will obtain that L_{atency} = 0,64 nsec.

$$L_{\mathrm{atency}} = \frac{\mathrm{B_{size}}}{\mathrm{T}_{hroughput}} \quad \textbf{(2)}$$

The above times refer to the production of only 128 bits (one block). However, in cryptographic tests we want larger samples in order to find any non-random patterns in the bitstream. If we produce samples of 1 Mbit (for the reasons that we explain in paragraph 5), we must multiply the above times by 7812,5 (10^6 bits = 128 • 7812,5). Therefore, we will need approximately 362 μsec with software simulation and 5 μs with hardware simulation. With these production times we calculated the third and fourth column of Table 1, which shows that it even with the fastest algorithm implementations in software or hardware, it will take an astronomical number of years to produce samples for all the keys.

Key size (bits)	**Key combinations (number of samples)**	**SAMPLE PRODUCTION TIME FOR ALL KEYS** (for 1 Mbit samples)	
		Software simulation (362 μsec / sample)	**Hardware simulation (5 μs / sample)**
90	2^{90}	$1,42 \bullet 10^{16}$ years	$1,96 \bullet 10^{14}$ years
128	2^{128}	$3,87 \bullet 10^{27}$ years	$5,35 \bullet 10^{25}$ years
256	2^{256}	$1,32 \bullet 10^{66}$ years	$1,83 \bullet 10^{64}$ years

Table 1. Total time for the production of algorithm samples for all keys

In addition to the practically impossible time which is needed in order to produce output samples for all the keys, we have to take into account and the time which is needed for the application of the statistical tests to each sample. This procedure takes much more time, as we will see in paragraph 6**.** From all these factors, it is obvious that during the cryptographic tests we are obliged to test a much smaller number of output samples (keys), with the use of the sampling method which is described below.

3. Sampling method

The sampling method is used when we want to measure the existence of some particular attributes on a very large number of *N* elements, therefore is very difficult and time consuming to test all the *N* elements one by one. To avoid this difficulty, we examine only a small number *n* of the total population *N* and then we make estimation for the total population, with a predetermined sampling error.

In our case, the procedure is that for each of the *n* sampling keys we generate an output sample of the algorithm and we store the *n* samples in corresponding files. Then, we submit all the output samples to the relevant statistical and cryptanalytic tests and we save the results of the tests, in order to make our estimations. The final decision on the cryptographic power of the algorithm (randomness, independence, unpredictability of outputs, etc.) is made based on the overall success rate of the tests in the samples. These checks are extremely time consuming, therefore if we want to have a reliable sampling (small sampling error) but also a practically feasible time to perform the checks, the main problems that arise are the following:

a. How many output samples we must check?

b. What should be the size of each sample?

c. How can we reduce the time of the tests?

d. What criteria should we use in order to select the sampling keys?

e. How do we rate the security (cryptographic strength) of the algorithm based on its test results?

In the present work we will analyze problems (a), (b), (c) and we will propose methods for solving them. The problems (d) and (e) will be addressed in a future study.

4. Number of the output samples

In our case, ***n*** is the number of cryptographic algorithm output samples (which corresponds to the number of tested keys) and ***N*** is the total number of the keys. Also, we have two possible attributes ***p*** and ***q***, because we want to investigate the existence of two possibilities:

a) Attribute ***p*** : The contents (bits) of the sample are random (strong algorithm).

b) Attribute ***q*** : The contents (bits) of the sample are non-random (weak algorithm).

In our case, we consider that one sample is random when it passes all the individual tests of the test suites which were mentioned in paragraph 1. If the sample fails at least one individual test, then it is considered non-random.

Theoretically, our case belongs to the binomial probability distribution (two attributes). However, in order to facilitate the calculations, we chose the normal probability distribution, which is very close to the binomial distribution for n > 100, according to [2] and [8]. Therefore, from [8] and [9], we have that for the normal probability distribution the number of required samples ***n*** , is derived from equation (3):

$n = \dfrac{\dfrac{z^2pq}{e^2}}{1 + \dfrac{1}{N}\left\{ \dfrac{z^2pq}{e^2} - 1 \right\}}$ **(3)**	where: N = total population (total number of keys) z = 1.96 (the value of the normal probability distribution curve for confidence level 95%) p = percentage of random samples $q = 1 - p$ = percentage of non-random samples e = sampling error for our estimation (level of precision)

Due to the fact that the number of cryptographic keys ***N*** is a very large number (usually between 2^{128} and 2^{256}) which practically tends to infinity, equation (3) can be simplified to (4):

$$n = \frac{z^2pq}{e^2} \qquad \textbf{(4)}$$

For our calculations it is better to take the worst case, in which ***n*** becomes maximum. This happens when the product pq is maximum. Since the sum of the probabilities p and q is constant (equal to 1), their product pq is maximum when $p = q = 0.5$ *(50%).* In other words, *n* is maximum when the percentage of random and non random samples are equal (maximum variability of the two attributes).

If we substitute in equation (4) the values which were described above (z=1.96 and p = q = 0.5), we can calculate the required number of the algorithm samples ***n*** for some characteristic values of the sampling error ***e.*** It is obvious that if we want to reduce the sampling error ***e***, we need to increase the number of samples ***n***. However, during the cryptographic checks, in addition to the number of samples, an important role plays the size of each sample, which we will examine in the next paragraph.

We must note that in the following paragraphs 5 and 6, our study is based on stream ciphers, but the final results and conclusions are similar for block ciphers. Moreover, we can test the block ciphers in the OFB mode operation, which simulates the function of a stream cipher.

5. Size of the output samples

If we take stream ciphers as an example, they mix their digital output sequences with the open information (using the XOR function) in order to produce the encrypted information. These sequences (key streams) must be as complex as possible, so that they can not be calculated and reproduced by an interceptor (i.e. they must be close to random). But in practice the cryptographic sequences of stream ciphers which use linear feedback registers (LFSR) are not completely random, but pseudo-random with a very long repetition period. Therefore, theoretically we would say that we need to test samples of the stream ciphers which are equal or greater in size than their minimum repetition period, in order to investigate the randomness along the entire length of the generated sequence.

Table 2 shows the minimum repetition periods of four simple stream ciphers. To simplify the calculations, we assumed that each stream cipher is consisted of 1 to 4 identical LFSRs of maximum length (shown in the columns of the table). For each stream cipher we set as alternative lengths of LFSR, the 40, 48 and 56 bits (rows of the table). If L is the length (in bits) of an LFSR, then its minimum repetition period is 2^{L} -1, while when we have a combination of more than one, the total period is equal to the product of their periods, according to [10]. Therefore, in Table 2 their minimum repetition period is calculated from the equation:

$$\text{Minimum repetition period} = (2^{L1} -1) \bullet (2^{L2} -1) \bullet (2^{L3} -1) \bullet (2^{L4} -1)$$

(where L1, L2, L3, L4 are the lengths of LFSR and L1 = L2 = L3 = L4)

In Table 2 for each L, in addition to the period, we also calculated the required transmission time of the bits in a communication channel with a speed of 10 Mbits/s (rows of the table with grey background). This is the time which is required by an interceptor to receive all the periodically repeated bits and analyze the algorithm.

As it is shown in Table 2 the interception of all the bits is impossible, due to the huge amount of time required. Therefore, we do not consider it necessary to test such long sequences of output samples for the evaluation of the algorithm.

Length L of LFSR (bits)	**MINIMUM PERIODS OF STREAM CIPHER** (bits) (the gray rows show their required transmission time with 10 Mbits/s)			
	with **1 LFSR**	with **2 LFSR**	with **3 LFSR**	with **4 LFSR**
40	$1{,}1*10^{12}$ bits	$1{,}2*10^{24}$ bits	$1{,}32*10^{36}$ bits	$1{,}46*10^{48}$ bits
	1,28 days	$3{,}8*10^{9}$ years	$4*10^{21}$ years	$4{,}48*10^{33}$ years
48	$2{,}8*10^{14}$ bits	$8*10^{28}$ bits	$2{,}23*10^{43}$ bits	$6{,}27*10^{57}$ bits
	321,2 days	$2{,}48*10^{14}$ years	$7{,}04*10^{28}$ years	$19{,}8*10^{42}$ years
56	$7{,}2*10^{16}$ bits	$5{,}2*10^{33}$ bits	$3{,}74*10^{50}$ bits	$2{,}7*10^{67}$ bits
	228,4 years	$16{,}8*10^{18}$ years	$11{,}8*10^{35}$ years	$4{,}32*10^{52}$ years

Table 2. Minimum repetition periods (keystreams) of four different stream ciphers

A more feasible but also substantial approach is to choose the size of the tested samples depending on the volume of information which will be encrypted. In particular, we are interested in the volume of information which is encrypted with the same key, in order to examine the robustness of the cryptosystem in practical cryptanalysis conditions. In most cryptosystems the key changes at least every day, so we are interested in the amount of information which is encrypted during a day. However, some cryptosystems (such as voice crypto) change the key with each new communication (session key), so in this case we are interested in the volume of information which is encrypted during a communication connection (session). As shown in Table 3, the size of the digital information varies greatly depending on the type and the duration of the information.

According to the above, in the case that we want to evaluate a stream cipher, the size of its output sequence (key stream) which we will test, must be at least equal to the maximum size of the information which will be encrypted (or alternatively the most usual size). For example, we see from Table 3 that in the case that we have 10 pages of text for encryption, we need an output sequence of only 100 Kbytes, while for 10 minutes of video, we need to produce an output sequence of 800.000 Kbytes. In the tests which are described in the next paragraph we had time constraints and also a restricted computing power, so we chose output samples of 132 Kbytes (1 Mbit in binary unformatted form). The size of 1Mbit is enough for simple texts up to 12 pages and for digitized voice up to 3.5 minutes. However, when we have a longer duration of telephone communication or when we want to encrypt photo or video, the sample size of 1Mbit is too small, so we may not discover some non-random patterns in the output of the stream cipher.

Information kind	**Duration / Resolution**	**Size (Kbytes)**	**Notes**
Simple text	10 pages	100	MS Word
Voice	5 minutes	1440	vocoder (4800 bits/s)
Picture or drawing	5 Mpixels	3000	compression JPEG
Music	3 minutes	6900	compression mp3 / 320 kbps
Music	3 minutes	2800	compression mp3 / 128 kbps
Video	10 minutes	800.000	compression MPEG

Table 3. Indicative size values of different kinds of digitized information

6. Required time

Based on what we have previously analyzed, in order to calculate the time required for the cryptographic tests, first we apply equation (4) of paragraph 4, in order to calculate the required number of samples for certain characteristic values of the sampling error. The results are shown in the first two columns of the comparative Table 4, from where we see that if we want to reduce the sampling error, we must greatly increase the number of the samples. For a sampling error of 3% we must test 1067 keys, for sampling error 2% we must test 2401 keys, while for sampling error 1% we must test 9604 keys. However, the larger number of samples means a much longer time for their production and testing procedure.

The total time required to test a sample of the algorithm is equal to the sum of the time required to produce the sample, plus the time required to conduct its randomness and cryptanalytic robustness tests. In practice, we measured that for an algorithm sample of 1Mbit, this total time is about 2 hours, using the statistical randomness tests of the Crypt-X software described in [3], which we ran on a commercial PC (3.6 GHz, with 4

GB RAM). Based on these tests, we created the second part of the comparative Table 4 (columns 3,4,5,6,7,8), which shows the total time required for the investigation of the *n* samples, if we share the samples production and testing to 1, 2, 3, 4, 5 and 6 PC.

Table 4 shows, on the one hand, how time-consuming it is to evaluate a cryptographic algorithm and on the other hand, how drastically the evaluation time can be reduced if we have the ability to use many computers for the algorithm evaluation. This is because we will distribute the samples production and testing to multiple computers which they will work in parallel, thus reducing the total time.

In practice, in order to make a more detailed and reliable evaluation, the sampling error should be less than 3%. In the case of a 3% sampling error, Table 4 shows that in order to complete the evaluation in a practically usable time of 45 days (1.5 months), we must have six parallel working computers. If we want a smaller sampling error or less days in order to complete the evaluation, then obviously we must use more computers.

SAMPLING ERROR (*e*)	**NUMBER OF SAMPLES** (*n*)	**REQUIRED TIME (days)** (with computers working 8 hours/day)					
		1 PC	**2 PC**	**3 PC**	**4 PC**	**5 PC**	**6 PC**
10 %	96	24	12	8	6	5	4
9 %	119	30	15	10	8	6	5
8 %	150	38	19	13	10	8	6
7 %	196	49	25	17	12	10	8
6 %	267	67	34	23	17	13	11
5 %	384	96	48	32	24	19	16
4 %	600	150	75	50	38	30	25
3 %	1067	267	134	89	67	53	45
2 %	2401	600	300	200	150	120	100
1 %	9604	2401	1200	800	600	480	400

Table 4. Required time of cryptographic tests based on the desired sampling error

It is therefore evident that, by using more numerous and more powerful computers, we are able to test a much larger number and size of the samples, and thus minimize the sampling error. The bottom line is that, if we want to make the evaluation of the cryptographic algorithms more thorough and more reliable, we must use as much as possible computing power.

And finally, it is obvious that for the final decision about the number of the samples, we must make a compromise between the reliability of the evaluation (small sampling error) and the available time and computer power.

7. Reduction of the testing time

There are some methods in order to reduce the extremely long time of the cryptographic tests which were described in the previous paragraph. These are:

a. Automation and parallelization: A very significant reduction of the testing time can be achieved with the use of appropriate software which will automate the main time-consuming processes, such as the production of the samples, the execution of the statistical tests, and the classification of the results. In addition, as we noted in paragraph 6, a very significant time reduction can be achieved if we share the total work to several computers. For the automation of all the testing processes, these computers must be connected in parallel through a LAN (Local Area Network) and they must be controlled through a parallel processing software.

As it is shown in Figure 1, if we have k computers connected in parallel, the n sampling keys of the algorithm will be evenly divided and distributed to them. This means that each computer will receive n/k keys and will work in parallel with the others, in order to produce its corresponding n/k output samples of the algorithm, apply the statistical tests to them and send the results back to the controller.

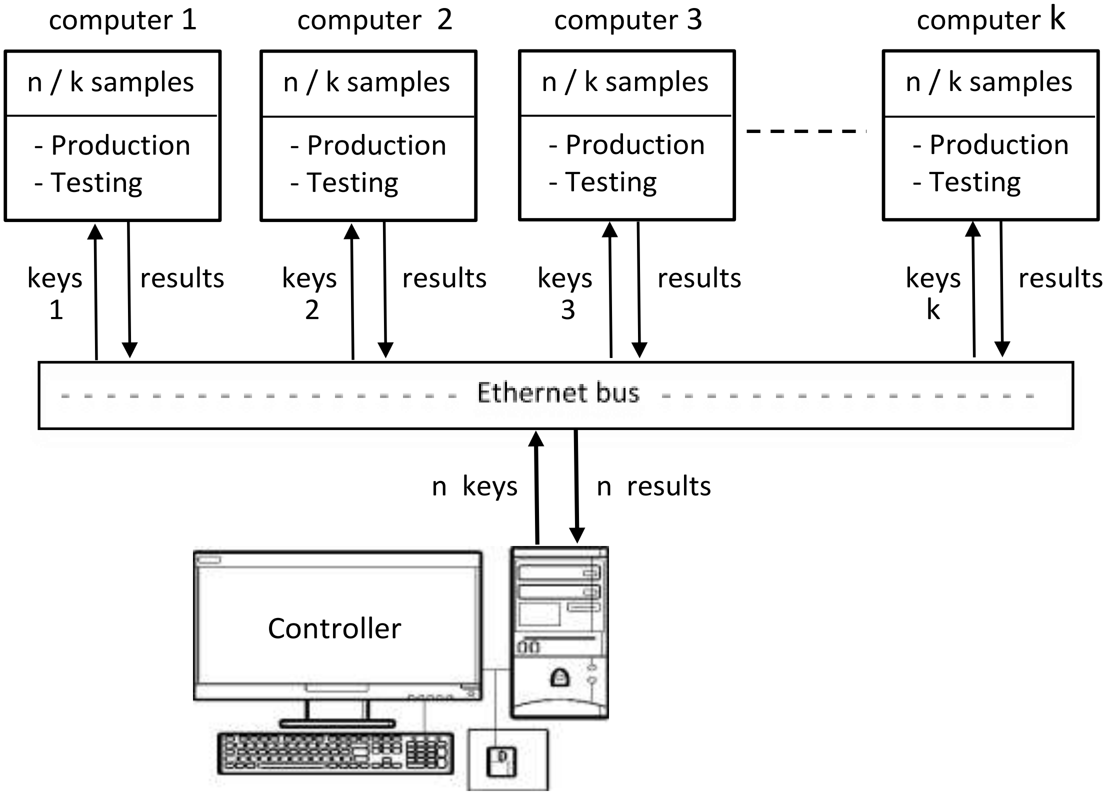

Figure 1. Reduction of the time for cryptographic tests, dividing the work through a parallel working LAN (keys 1 + keys 2 + keys 3 + ... + keys k = n).

The total time T_P for the production and testing of the samples in Figure 1 will be:

$T_P = T_1 / k$ (where T_1 is the time required with one computer).

This means that the time corresponding to the use of only one computer in the Table 4 of the previous paragraph, can be reduced to one tenth with the use of ten computers running in parallel.

b. Graphics Processing Unit (GPU): On modern computers the Graphics Processing Unit (GPU) has great capabilities of parallel processing (which is necessary for managing the pixels of the screen). Thus, many GPU manufacturers allow the users to use special software to program the GPU for their own specific applications, as it is described in [11], [12] and [13]. One such special application of the GPU can be its programming to speed up the cryptographic tests, in the context of what was mentioned in the previous subparagraph 7a.

c. Programmable hardware (FPGA or ASIC): For the purpose of the tests, the implementation of the cryptographic algorithm is easier when it will be done using a high-level programming language. However, for cryptanalytic and testing purposes, the execution time of the encryption can be dramatically accelerated if the algorithm is implemented in an especially programmable hardware FPGA (Field Programmable Gate Array) or ASIC (Application Specific Integrated Circuit), as it is described in [7] and [14]. The total time of the cryptographic tests can be reduced furthermore if we will also implement the statistical tests in FPGA or ASIC.

d. Overclocking: The technique of overclocking, concerns the increase of the nominal frequency of the computer clock (which has been determined by the manufacturer) in order to make the computer to operate faster than its original speed. However, this technique is controversial and we do not recommend it, because it can cause overheating, instability, reduction of the lifespan and even complete destruction of the CPU. Basic information and guidelines for overclocking can be found on [15], [16], [17], [18].

8. Conclusions

In order to reduce the extremely long time of cryptographic algorithm tests, we use the sampling method in which we examine a much smaller number *n* of the total key combinations *N*, and then we decide for the total key combinations with a predetermined sampling error. Due to the fact that *N* is extremely large, we ended up to a simplified equation which indicates that the number of sampling keys *n* practically is independent from *N* and it is inversely proportional to the sampling error *e*. Our calculations showed that for sampling error 3% we must test 1067 keys, for sampling error 2% we need 2401 keys, while for sampling error 1% we need 9604 keys. It is obvious that for the final decision about the number of samples, we must make a compromise between the reliability of the sampling (small sampling error) and the available time and computer power. Concerning the size of the tested samples, a practical and feasible rule is to choose a length equal to the maximum length of the information which will be encrypted.

The time which is needed for the cryptographic tests can be reduced with the automation of the main processes (production and testing of the samples, classification of the results etc.). Also, a significant time reduction method is the sharing of the total work to several computers working in parallel through a LAN. A further time reduction can be achieved with the implementation of the algorithm simulation and the statistical tests into special hardware (FPGA or ASIC). An alternate time reduction method is the implementation and parallelization of the above processes with the use of Graphics Processing Units (GPU).

References

[1]. Donald Knuth, "*The Art of Computer Programming-Volume 2 / Seminumerical Algorithms"*, Addison-Wesley 1998.

[2]. NIST Special Publication 800-22, "*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications"*, National Institute of Standards and Technology (NIST), April 2010.

[3]. Helen Gustafson et. al., "*A computer package for measuring strength of encryption algorithms"*, Journal of Computers & Security, 13(8), (1994), 687-697.

[4]. G. Marsaglia and G. Diehard, "*Battery of Tests of Randomness"*. Available at: http://stat.fsu.edu/pub/diehard/ (1996), http://stat.fsu.edu/pub/diehard/

[5]. George Marinakis, "Minimum key length for cryptographic security" http://www.scienpress.com/journal_focus.asp?main_id=57&Sub_id=IV&Issue=597

[6]. Daniel J. Bernstein and Peter Schwabe, "New AES software speed records", http://cr.yp.to/aes-speed/aesspeed-20080926.pdf

[7]. "Helion Giga AES cores" , https://www.heliontech.com/aes_giga.htm

[8]. Harrison M. Wadsworth, "*Handbook of Statistical Methods for Engineers and Scientists"*, Mc Graw-Hill, 1990.

[9]. Cochran W.G. *"Sampling Techniques",* 2nd Ed., New York: John Wiley and Sons, Inc.

[10]. Alfred Menezes, Paul C. van Oorschot, Scott A.Vanstone, "*Handbook of Applied Cryptography"*, CRC Press 1997.

[11]. J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, J. Phillips *"GPU Computing"* Proceedings of the IEEE, May 2008.

[12]. O. Harrison and J. Waldron, *"Practical Symmetric Key Cryptography on Modern Graphics Hardware"*. 17th USENIX Security Symposium. San Jose, CA. July 28 - August 1, 2008.

[13]. O. Harrison and J. Waldron, *"Efficient Acceleration of Asymmetric Cryptography on Graphics Hardware"*, AfricaCrypt 2009, Gammarth, Tunisia, June 21-25, 2009.

[14]. Kris Gaj and Pawel Chodowiec, *"FPGA and ASIC Implementations of AES",* http://teal.gmu.edu/courses/ECE746/project.

[15]. Avast, *"How to Safely Overclock Your CPU on Windows"* https://www.avast.com/c-how-to-overclock-cpu.

[16]. Intel, *"How to Overclock Your Unlocked Intel Core Processor"* https://www.intel.com/content/www/us/en/gaming/resources/how-to-overclock.html.

[17]. Intel, *"How to Overclock Your CPU from BIOS"* https://www.intel.com/content/www/us/en/gaming/resources/bios-overclocking.html.

[18]. AMD, *"Your tool to unlock AMD Ryzen™ Processors"* https://www.amd.com/en/technologies/ryzen-master.