Journal of Theoretical and Applied Information Technology
15th August 2022. Vol.100. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPARATIVE ANALYSIS OF THE PERFORMANCE OF GENERATING CRYPTOGRAPHIC CIPHERS ON THE CPU AND FPGA

[1]BEKBOLAT MEDETOV, [1]TANSAULE SERIKOV, [1]ARAY TOLEGENOVA, [2]ZHEXEBAY DAUREN

[1] Kazakh Agrotechnical University named after S. Seifullin, Nur-Sultan, Kazakhstan

[2] Nazarbayev University, Kabanbay Batyr Ave 53, Nur-Sultan, 010000, Kazakhstan

E-mail: [1]bekbolatmedetov@gmail.com, [1]tansaule_s@mail.ru, [1]arai82@bk.ru, [2]zhexebay92@gmail.com

ABSTRACT

With the rapid development of technologies for the use of artificial neural networks (ANN) to solve various complex and poorly algorithmized problems, recently there have been more and more ideas to use ANN to decrypt data encrypted by cryptographic protocols such as AES. As a rule, when solving a particular problem, ANNs require large amounts of data for their training. For example, in the task of data decryption, it is necessary to have a sufficiently large array of data that make up a pair: encrypted and original information. In the case of considering block data encryption using protocols of the AES family, the sizes of input and output data for ANN can be 128 bits. Thus, to implement an ANN capable of decoding data, first of all, it is necessary to develop a generator of training data, which is an array of size Nx128, where N is the amount of data. In this paper, the possibilities of implementing a training data generator for ANN using a CPU (central processing unit) and an FPGA (field-programmable logic integrated circuit) are considered. A comparative analysis of their performance in solving this problem is also carried out. Our experimental calculations show that the FPGA based oscillator has better performance than the CPU based oscillator. Moreover, the financial costs of building an FPGA generator turn out to be noticeably lower than the financial costs required to build a computer with a CPU. Thus, in this paper, we conclude that the implementation of the training data generator on the FPGA is more profitable both in terms of performance and cost.

The main result of the research is the experimental confirmation of the ability of AES-type encryption algorithms to be executed in parallel at a very high level. This statement is made on the basis of the implementation of encryption functions in the Verilog language for their execution on the FPGA, which perfectly supports parallel computing. We also inform you that we have developed a library written in the Verilog language, where all the functions for data encryption according to the AES protocols are implemented. And finally, this library was used to develop a high-speed encrypted data stream generator using AES encryption algorithms. In the future, this generator is planned to be used to generate a super-large amount of training data needed to train ANNs used to decrypt or search for vulnerabilities in encryption algorithms.

Keywords: *Programmable logic integrated circuit, Artificial neural networks, Algorithm, Encryption, Central processing unit, Protocol, Flow, Quality, Field-programmable logic integrated circuit, Pseudo-random number generator.*

1. INTRODUCTION

In our work, we touch upon some questions about the possibility of creating systems based on ANN and intended for decrypting data encrypted using protocols of the AES family. We were mainly interested in the question of choosing the most optimal way to generate training data for the ANN. As possible options, two options for the development of this generator were considered: 1) on the CPU and 2) on the FPGA. Since the problem under consideration assumes the need to generate a large amount of data, we pay special attention to the performance and cost of various methods for solving this problem.

It should be noted that ANNs paired with FPGAs are used to solve many problems similar to ours. For example, in [1], it is proposed to develop a vector-valued tree parity machine (VVTPM),

which is a generalized architecture of TPM models and can be more efficient and safer for real systems. In terms of efficiency and safety, it shows that the VTPM synchronization time is of the same order of magnitude as the baseline TPM model, and it may be safer than previous results with the same synapse depth. And in [2] it is reported that a pseudo-random number generator (PRNG) was developed based on artificial neural networks (ANN). This PRNG was used to develop a stream encryption system with high statistical randomness properties of its key sequence using ANN. The software simulation was built using MATLAB to firstly pass four well-known statistical tests that guarantee randomness characteristics. Secondly, such a stream encryption system must be implemented using FPGA technology, so the minimum hardware requirements must be considered. To ensure the performance of an FPGA-based PRNG compared to a MATLAB-based PRNG, the corresponding table lists the statistical properties of both the software and hardware versions of the PRNG.

The following work [3] presents the design of a neural network that performs the processes of encryption and decryption of the properties of a cipher with a symmetric key. The target from the network was selected according to the derivation of the AES Advanced Encryption Standard, which provides effective and recommended security. The results of computer simulation of the proposed NN cryptosystem show the closeness of the results achieved by the proposed NN-based AES cryptosystem to the results of conventional AES. A Field Programmable Lattice Array (FPGA) was chosen for high speed and high efficiency implementation. The neural network was implemented using a pipelined method to reduce the coupling between neurons. The final hardware was implemented by programming the Xilinx XCVirtex -E family with a 128-bit data bus, a maximum clock rate of 500 MHz, and a bandwidth of 4 Gbps for a single neural network. The tool used to implement the system is Foundation ISE 3.1i.

And in [4], a similar problem was considered as in [3], i.e., modern designs of neural network accelerators and common methods used. Based on the evaluation in this paper, when software and hardware are co-designed, an FPGA can provide more than 10 times better speed and power efficiency than a modern GPU. This shows that FPGA is a promising candidate for neural network acceleration. Techniques used to automate accelerator design are also reviewed, showing that current development methods can deliver both high performance and a network switch at runtime. But there is still a gap between ongoing projects and evaluation. On the one hand, quantization with an extremely narrow bit width is limited by the accuracy of the model, which requires further study of the algorithm. On the other hand, combining all methods requires more research in both software and hardware in order for them to work well together.

The following work [5] reports on the construction of an artificial neural network implemented on an FPGA with a small amount of hardware resources, while providing a high classification speed without information loss. To achieve this goal, they improved the MobileNet-V1 architecture. And also, the Ad-MobileNet model was studied using large databases and various applications to identify the disadvantages and advantages of this model.

And in [6], the use of a neural network in cryptography is considered, for example, the development of such a neural network that would be practically used in the field of cryptography. This article also includes an experimental demo. The purpose of cryptography is to make it impossible to use the cipher and reproduce the original plaintext without the corresponding key. With good cryptography, your messages are encrypted in such a way that brute force attacks against the algorithm or key are nearly impossible. Good cryptography ensures its security by using incredibly long keys and by using encryption algorithms that are resistant to other forms of attack.

Finally, in [7] the requirements are formed and the main stages of the synthesis of cryptological encryption and data decryption modules are explained. The table-algorithmic method for calculating the scalar product has been improved and is focused on working with floating-point operands. Structures of modules for cryptographic encryption/decryption of data have been developed. The development of specialized tools for modules of neuron-like cryptographic encryption and data decryption has been carried out. For this purpose, the VHDL programming language and the Quartus II development environment (version 13.1) were used. Estimated hardware costs and decryption time of the module of specialized decryption tools implemented on the FPGA.

The analysis of the scientific literature in this article shows that there are no real experimental measurements that would compare the

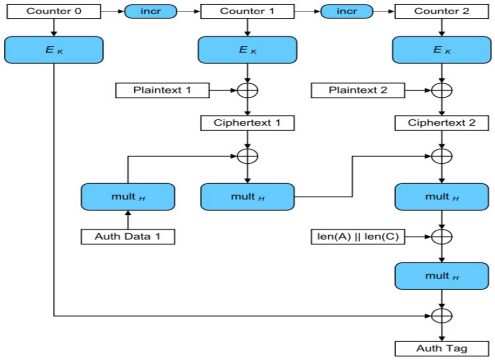
performance of FPGAs and CPUs when solving problems of organizing high-speed data encryption using AES protocols. There are also no studies evaluating the relative financial benefits of using FPGAs, CPUs, or other computing devices. This work aims to fill these gaps. We also believe that such an analysis, which was carried out in this work, can help in choosing the necessary computing device to speed up the execution of certain operations, for example, data encryption using AES protocols.

2. STATEMENT OF THE PROBLEM AND PREPARATION OF A NUMERICAL EXPERIMENT

The task was to conduct a comparative analysis of the performance of data encryption using protocols of the AES family on the CPU and FPGA. For this purpose, PCs with the following CPU specifications were used: Core i5-3230M 2.60 GHz, 4.0 GB RAM and Core i7-8700 3.20 GHz, 16.0 GB RAM. The model of this device xc7a100t-1csg324 was used as an FPGA.

Below is an algorithm for implementing AES-GCM-128 and AES-GCM-256 in the form of a flowchart and mathematical functions.

The figure shows the general structure of AES-GCM encryption.



GCM (Galois / Counter Mode) is a widely used mode of operation of symmetric block ciphers with high efficiency and performance. Authenticated encryption (AEAD) is a method that ensures the confidentiality and identification (integrity) of transmitted data.

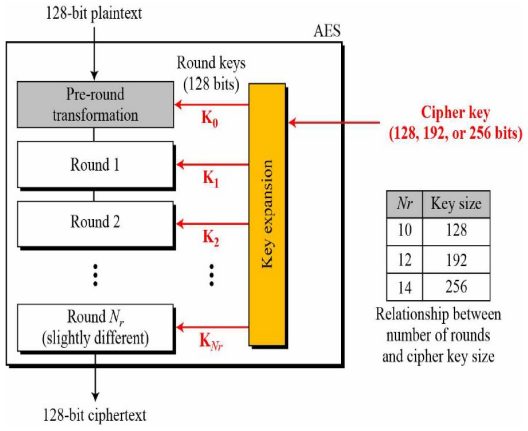
GCM mode is set for block codes with a block size of 128 bits. There is a GCM version called GMAC that only allows data to be verified and can be used as incremental code to verify messages. Both GCM and GMAC enter a starting vector of any length.

Due to the presence of a verification code (imitation), this approved encryption method allows the recipient to easily detect any changes (encrypted and added with explicitly sent data) before decrypting the message.

Encryption algorithm in GCM mode. A variant of one block of additional authenticated data "Auth Data 1" is presented. The initial value of the counter contains the vector IV (or is derived from it). The block receives two plaintext blocks 1 and 2. The E_K operation denotes encryption on the shared key K, the multH operation denotes multiplication in the GF field (2 ^ 128) by the hash key H, "incr" denotes the increment of the counter

In the usual CTR (counter) encryption mode, input blocks are numbered sequentially, the block number is encrypted with the E block algorithm (usually AES). The output of the encryption function is used in a plaintext XOR (exclusive or) operation to obtain the ciphertext. As with other counter-based modes, the scheme is a stream cipher, so it is imperative to use a unique initialization vector for each encrypted data stream.

GCM uses the Galois function "Mult" ("GHASH (H, A, C)"), which combines ciphertext blocks and an authentication code to produce an authentication tag. The input of the function is the hashing key H, which is the result of encrypting 128 zero bits on the key K, i.e. H = E (K, 0 ^ 128). The authentication tag is used to verify the integrity of the message. The channel transmits: the initialization vector [en] IV, ciphertext blocks, and the authentication code (16 bytes). GCM (GMAC) mode is similar to HMAC in its properties.



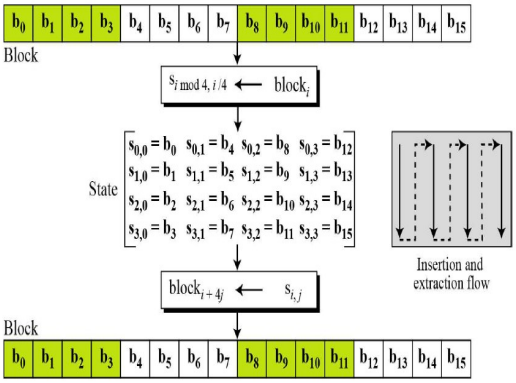
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

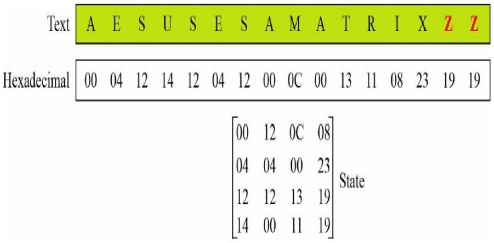
Below shows an algorithm how to transformation ciphertext-to-matrix. Block-to-state and state-to-block transformation.

Journal of Theoretical and Applied Information Technology
15th August 2022. Vol.100. No 15

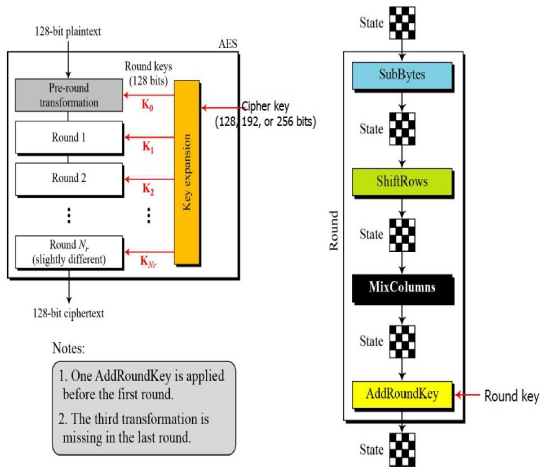
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



Changing plaintext to state

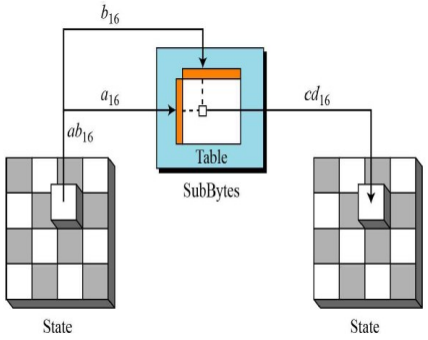


Structure of Each Round



The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.

The SubBytes operation involves 16 independent byte-to-byte transformations.



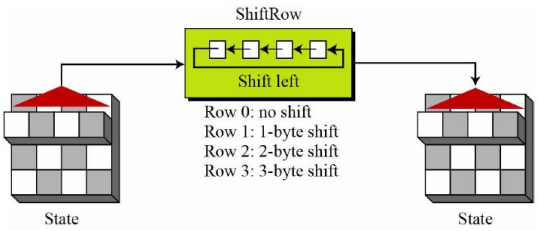
SubBytes transformation table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Another transformation found in a round is shifting, which permutes the bytes.

In the encryption, the transformation is called ShiftRows.

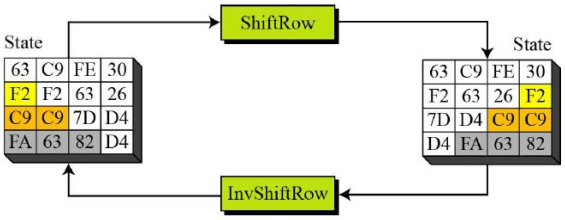
In the decryption, the transformation is called InvShiftRows and the shifting is to the right.



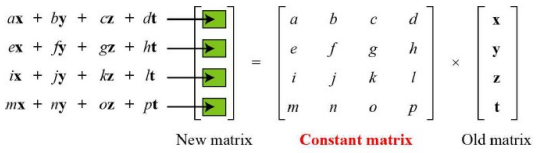
The figure shows how the state is transformed using the ShiftRow transformation. The figure also shows that the invshiftrows transformation creates the initial state.

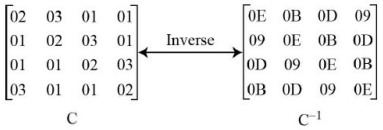
Journal of Theoretical and Applied Information Technology
15th August 2022. Vol.100. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

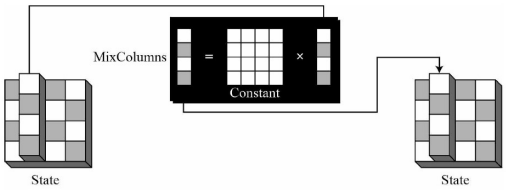


The figure shows how to mix bytes using matrix multiplication

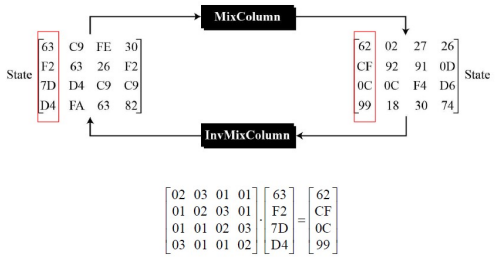




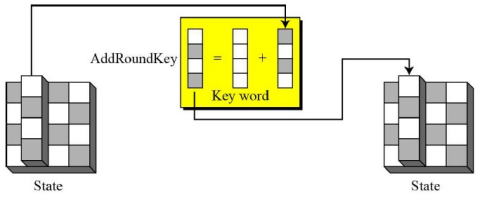
The MixColumns transformation operates at the column level; it transforms each column of the state to a new column. The InvMixColumns transformation is basically the same as the MixColumns transformation.



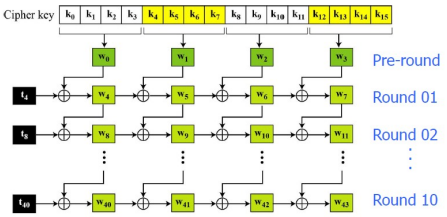
The figure shows how the state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.



AddRoundKey proceeds one column at a time. AddRoundKey adds a round key word with each state column matrix.



To create round keys for each round, AES uses a key-expansion process. If the number of rounds is N_r, the key-expansion routine creates $N_r + 1$ 128-bit round keys from one single 128-bit cipher key.



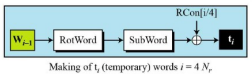
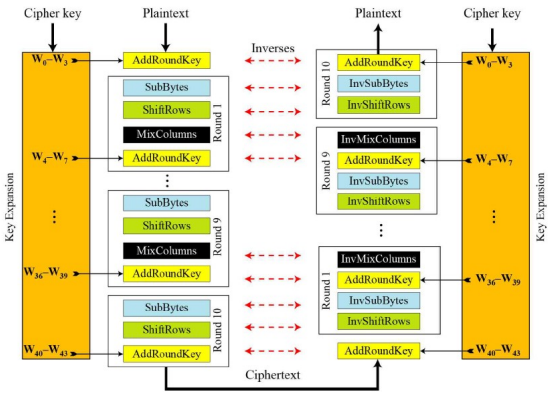


Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	(01 00 00 00)16	6	(20 00 00 00)16
2	(02 00 00 00)16	7	(40 00 00 00)16
3	(04 00 00 00)16	8	(80 00 00 00)16
4	(08 00 00 00)16	9	(1B 00 00 00)16
5	(10 00 00 00)16	10	(36 00 00 00)16

Below in table shows the example how the keys for each round are calculated assuming that the 128-bit cipher key is (24 75 A2 B3 34 75 56 88 31 E2 12 00 13 AA 54 87)16.

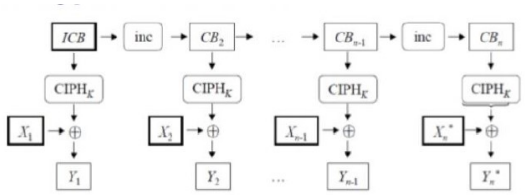
Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		w_{00} = 2475A2B3	w_{01} = 34755688	w_{02} = 31E21200	w_{03} = 13AA5487
1	AD20177D	w_{04} = 8955B5CE	w_{05} = BD20E346	w_{06} = 8CC2F146	w_{07} = 9F68A5C1
2	470678DB	w_{08} = CE53CD15	w_{09} = 73732E53	w_{10} = FFB1DF15	w_{11} = 60D97AD4
3	31DA48D0	w_{12} = FF8985C5	w_{13} = 8CFAAB96	w_{14} = 734B7483	w_{15} = 2475A2B3
4	47AB5B7D	w_{16} = B822deb8	w_{17} = 34D8752E	w_{18} = 479301AD	w_{19} = 54010FFA
5	6C762D20	w_{20} = D454F398	w_{21} = E08C86B6	w_{22} = A71F871B	w_{23} = F31E88E1
6	52C4F80D	w_{24} = 86900B95	w_{25} = 661C8D23	w_{26} = C1030A38	w_{27} = 321D82D9
7	E4133523	w_{28} = 62833EB6	w_{29} = 049FB395	w_{30} = C59CB9AD	w_{31} = F7813B74
8	8CE29268	w_{32} = EE61ACDE	w_{33} = EAFE1F4B	w_{34} = 2F62A6E6	w_{35} = D8E39D92
9	0A5E4F61	w_{36} = E43FE3BF	w_{37} = 0EC1FCF4	w_{38} = 21A35A12	w_{39} = F940C780
10	3FC6CD99	w_{40} = DBF92E26	w_{41} = D538D2D2	w_{42} = F49B88C0	w_{43} = 0DDB4F40



ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The following algorithm describes the operations of $GHASH$:

0. *If X is the empty str ,hen return the empty string as Y.*
1. *Let* n=$[len(X)128]$.
2. $LetX1,2,X3,\ldots\ldots,Xn-1,Xn$ *the unique sequence of bits such that*:$X=X1||X2||...||Xn-1||Xn$.
3. *Let* $CB1=ICB$.
4. *For* $i=2n$, $CBi=inc(CBi-1)$
5. *For* $i=1n-1$, $Yi=XiCIPHK(CBi)$
6. *Let* $Yn=XnMSB(X*n)(CIPHK(CBi))$
7. *Let* $Y=Y1||Y2||\ldots||Yn-1||Yn$ 8.*Return* Y.



In addition to comparing the performance of these devices, we also set the task to find out which option can provide the greatest benefit in terms of price / quality ratio. In our case, the quality of the device means the speed of performing encryption tasks, i.e. how much data is encrypted per unit of time. Accordingly, the unit of measure for the "quality" device parameter is byte/sec. In this case, the price / quality ratio will be determined by the formula:

$$E = \frac{Q}{C} \quad (1)$$

where C – is the cost of the device, Q – is the "quality" of the device (its performance).

Table 1 below shows the price list of devices used in our experiment. These prices were used to calculate the value for money ratio for these three devices.

Table 1. Price list of devices

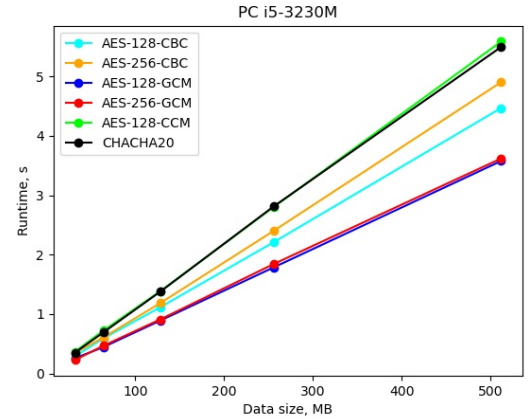
№	Device name	Price
1	Processor Core i5-3230M	$99.99 [8]
2	Processor Core i7-8700	$303.00 [9]
3	FPGA model xc7a100t-1csg324	$151.20 [10]

To determine the "quality" of these devices, various experimental measurements of the performance of these devices were carried out. At the same time, various data encryption protocols and algorithms for their implementation on FPGAs were considered. All data encryption functions on the FPGA were implemented in-house.

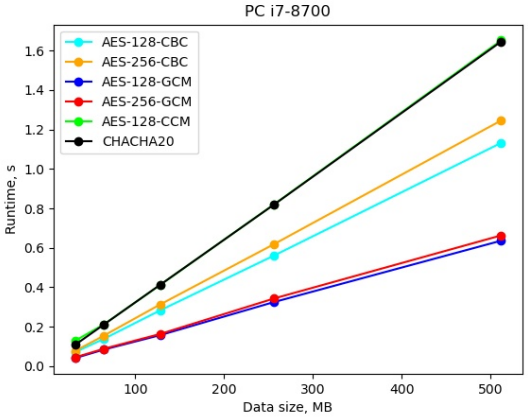
The main hypothesis in conducting these studies was built around the idea that despite block-by-block data processing, encryption algorithms using AES protocols can be performed in parallel at a high level. In this regard, the task of experimental confirmation of the correctness of this hypothesis was set. The correctness of the hypothesis is checked using a comparative analysis of the performance of encryption functions using the AES protocols on the FPGA and the CPU. The FPGA is used to implement parallel computing algorithms for data encryption. And the CPU is used to perform encryption functions using AES protocols without parallel computing. The CPU and FPGA are selected with comparable computing capabilities. In this case, if the computing performance on the FPGA is noticeably higher than on the CPU, then we can conclude that data encryption using the AES protocols is very well performed in parallel mode.

3. RESULTS OF NUMERICAL EXPERIMENTS

To experimentally determine the performance of devices, the following encryption protocols were considered: AES-CBC with a key size of 128 bits and 256 bits, AES-GCM with a key size of 128 bits and 256 bits, AES-CCM with a key size of 128 bits and Chacha20 with a key size of 256 bits . Initially, the speed of data encryption on the CPU for each type of AES protocol was studied. The result obtained is shown in Figure 1 as a dependence of the time for performing encryption operations on the amount of data. These calculations were carried out using the Python encryption library [11]. In this calculation, only the time of data generation was taken into account. As a result, from Figure 1 we see that encryption on the CPU using the Chacha20 and AES-CCM protocols takes more time than using the AES-GCM protocol.



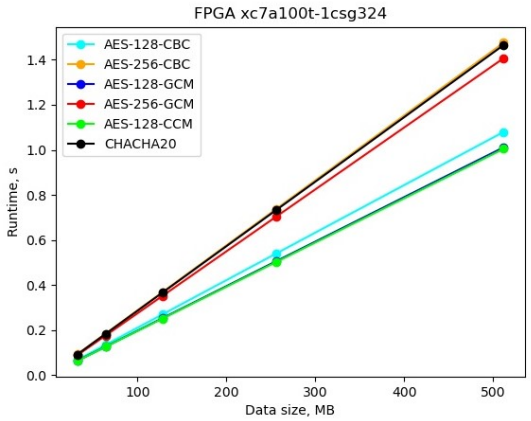
(a)



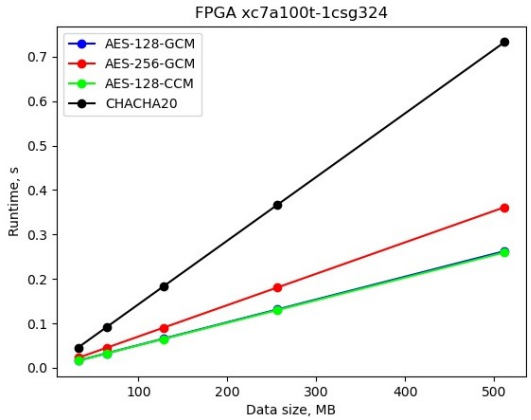
(b)

Figure 1: Dependence of time on the amount of encryption data for Core i5-3230M (a) and Core i7-8700 (b)

In the same way, numerical experiments were carried out to determine the speed of data encryption on the FPGA using different protocols. The results of these FPGA experiments are shown in Figure 2. During this experiment, it was found that the AES-CBC, AES-GCM and AES-CCM encryption algorithms take up only 15% of the FPGA resource with a 128-bit key, and 20% with 256 bit key. It turned out that the Chacha20 encryption algorithm requires more resources than AES (about 32% of all FPGA resources.) In AES-CBC encryption, the input block for each direct encryption operation (except the first one) depends on the result of the previous direct encryption operation, so the direct encryption operation does not can be performed in parallel [12]. Therefore, in the FPGA used in the numerical experiments, it is possible to parallelize AES-GCM and AES-CCM encryption into four streams, and Chacha20 – only into two streams. Of course, this limitation is related to the amount of resources used by the FPGA. Naturally, when using a more powerful FPGA, the number of threads will be greater.



(a)



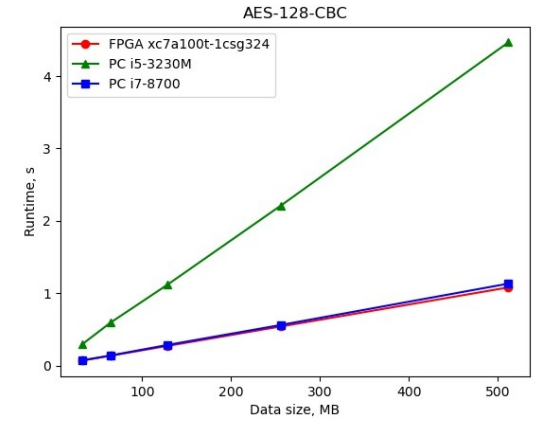
(b)

Figure 2: Dependence of time on the volume of data encryption for FPGAs: (a) without parallel processing and (b) with parallel processing

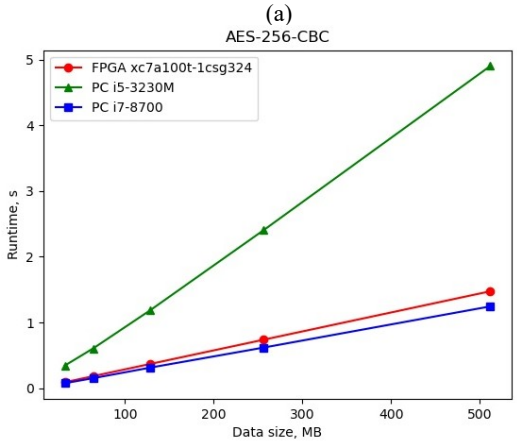
At the next stage, experimental measurements of the performance of AES-CBC encryption were carried out for the devices shown in Table 1. Figure 3 shows the result of these experiments in the form of three graphs of the dependence of the operation execution time on the amount of encrypted information. These results show that for the AES-CBC protocol, the i5-3230M processor has the lowest performance compared to other devices.

Journal of Theoretical and Applied Information Technology
15th August 2022. Vol.100. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



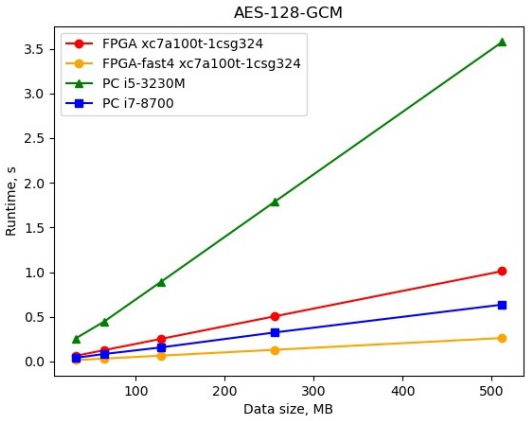
(a)



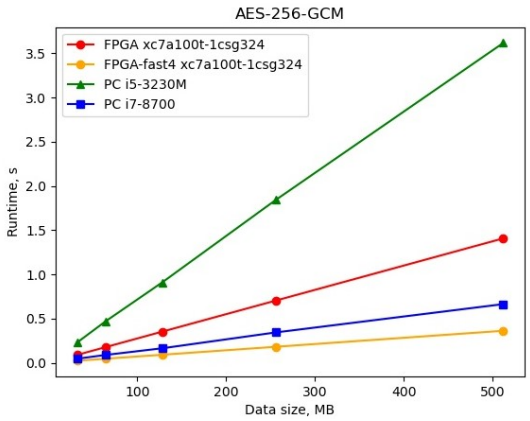
(b)

Figure 3. Comparison of AES-CBC encryption results: (a) AES-128-CBC and (b) AES-256-CBC

And Figure 4 shows the results of measuring the performance of AES-GCM protocol encryption. As you can see from this figure, AES-GCM encryption is faster on the i7-8700 CPU than on the FPGA, but the FPGA is faster than the i7-8700 when computing encryption blocks in parallel.



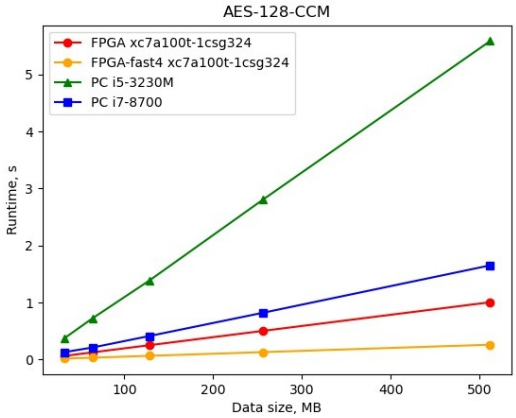
(a)



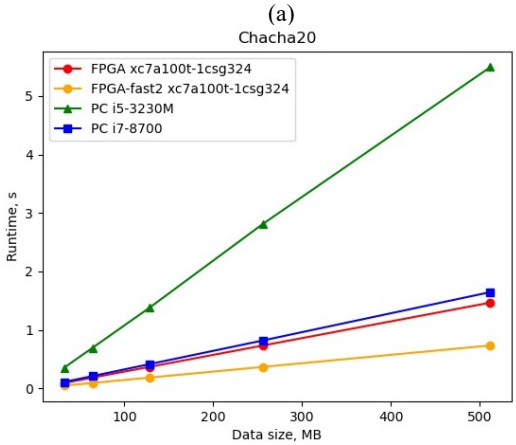
(b)

Figure 4. Comparison of AES-GCM encryption results: (a) AES-128-GCM and (b) AES-256-GCM

Further, the results of an experimental measurement of encryption performance using the AES-128-CCM and Chacha20 protocols are shown in Figure 5. From Figure 5, we see that the FPGA calculates encryption using these algorithms faster than all other devices. And it is also clear that with parallel processing on the FPGA, you can achieve higher performance than processing each block in one cycle.



(a)



(b)

Figure 5. Comparison of AES-128-CCM (a) and Chacha20 (b) encryption results

According to the graphs obtained, shown in Figures 1-5, we see that the execution time for data encryption depends on its volume in a linear manner. This means that this dependence can be represented as such a linear function:

$$T=a\cdot V+b \quad (2)$$

V - is the amount of data to be encrypted, and the coefficients a and b are empirical and are determined by the least squares method based on experimental data. By meaning, the parameter a in this formula means the inverse value of the encryption speed. In this case, the "quality" of the device, i.e. its performance can be determined by the formula:

$$Q=1/a \quad (3)$$

For each dependency graph shown in Figures 1-5, the corresponding approximating functions were determined according to formula (2). Also, for each type of encryption protocol and equipment, their "quality" (performance) was calculated. Tables 2 - 7 show the quality values and types of functions found from experimental measurements for each type of device and encryption protocols.

Table 2. Types of threats to information security

Device	Approximation function	Device quality (MB/Sec)
FPGA xc7a100t-1csg324	0.036V - 3.056e-06	27.78
PC i5-3230M	0.1481V + 0.0002056	6.75
PC i7-8700	0.03775V - 3.426e-05	26.49

Table 3. Types of threats to information security

Device	Approximation function	Device quality (MB/Sec)
FPGA xc7a100t-1csg324	0.0492V - 3.264e-06	20.33
PC i5-3230M	0.1628V - 8.265e-05	6.14
PC i7-8700	0.04153V - 2.143e-05	24.08

Table 4. Type of approximation function for the AES-128-GCM data encryption protocol

Device	Approximation function	Device quality (MB/Sec)
FPGA-fast4 xc7a100t-1csg324	0.008757V - 3.472e-07	114.19
PC i5-3230M	0.1186V + 0.0001986	8.43
PC i7-8700	0.02118V + 4.13e-05	47.21

Table 5. Type of approximation function for the AES-256-GCM data encryption protocol

Device	Approximation function	Device quality (MB/Sec)
FPGA-fast4 xc7a100t-1csg324	0.01205V + 1.389e-07	82.99
PC i5-3230M	0.1206V + 0.000191	8.29
PC i7-8700	0.02206V + 6.728e-05	45.33

Table 6. Type of approximation function for the AES-128-CCM data encryption protocol

Device	Approximation function	Device quality (MB/Sec)
FPGA-fast4 xc7a100t-1csg324	0.008664V + 9.697e-19	115.42
PC i5-3230M	0.1856V + 0.0002431	5.39
PC i7-8700	0.05449V + 0.000184	18.35

Table 7. Type of approximation function for the data encryption protocol Chacha20

Device	Approximation function	Device quality (MB/Sec)
FPGA-fast2 xc7a100t-1csg324	0.02443V + 1.939e-18	40.93
PC i5-3230M	0.1833V + 0.0002424	5.46
PC i7-8700	0.05467V + 5.632e-05	18.29

It is clear that in the tasks of data decryption using deep ANNs, it is required to use a large number of layers. Each layer can also contain a large number of neurons. All this leads to the need to calculate a huge number of ANN parameters. Accordingly, a large amount of training data is required. In order for ANNs to learn correctly, a certain relationship must be satisfied between the number of ANN parameters and the amount of training data [13]. According to [13], the minimum ratio is 1 to 10, i.e. the amount/volume of training data should be 10 times greater than the number of ANN parameters. But we believe that in order to achieve a higher level of ANN quality, it is necessary that this ratio be 1 to 1000.

Next, as an example, the volume and time required to generate training data for an ANN with ten layers was estimated. The parameters of this network are shown in Table 8 in the Tensorflow library format, where the total number of parameters is eventually determined to be about 9 million. To calculate the required amount of training data, we can use the following formula:

$$V = 1000 \cdot N_{in} \cdot \beta \quad (4)$$

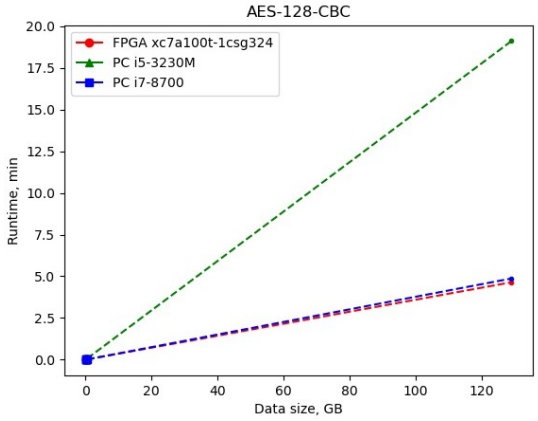
where N -_in is the size of the input data, which is 128 bits for protocols of the AES family [14] and 512 bits for the Chacha20 protocol [15], β - is the number of network parameters.

Table 8. The number of parameters of the neural network model with the sizes of the input and output layers 128 and 512

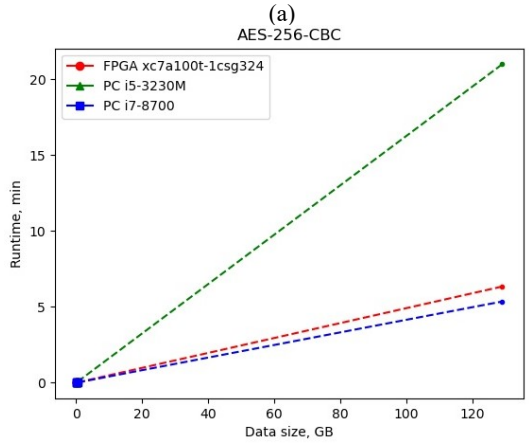
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	132096 / 525312
dense_1 (Dense)	(None, 1024)	1049600
dense_2 (Dense)	(None, 1024)	1049600
dense_3 (Dense)	(None, 1024)	1049600
dense_4 (Dense)	(None, 1024)	1049600
dense_5 (Dense)	(None, 1024)	1049600
dense_6 (Dense)	(None, 1024)	1049600
dense_7 (Dense)	(None, 1024)	1049600
dense_8 (Dense)	(None, 1024)	1049600
dense_9 (Dense)	(None, 128) / (None, 512)	131200 / 524800
Total params: 8,660,096 / 9,446,912		

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Thus, using formula (4) and the data from Table 8, for training a 10-layer ANN [16], we obtain the following amounts of necessary data: for protocols of the AES family - 129 GB, and for the Chacha20 protocol - 563 GB. As you can see, even training a small 10-layer ANN requires hundreds of gigabytes of data. We did not conduct appropriate experiments for such a large amount of data [17]. As can be seen from Figures 1-5, in experimental measurements, data volumes for encryption of no more than 600 MB were used [18]. But to calculate the encryption time for any amount of data, we can use the approximation functions given in tables 2-7. Figures 6, 7 and 8 show plots of all these approximation functions.

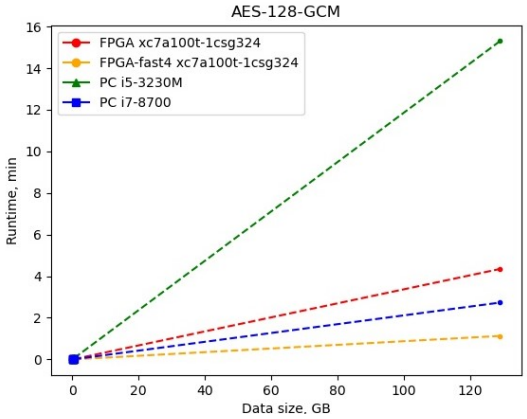


(a)

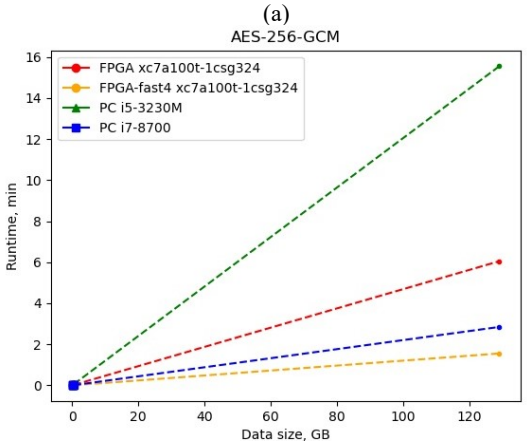


(b)

Figure 6. Comparison of AES-CBC encryption results for a data volume of 129 GB: (a) AES-128-CBC and (b) AES-256-CBC

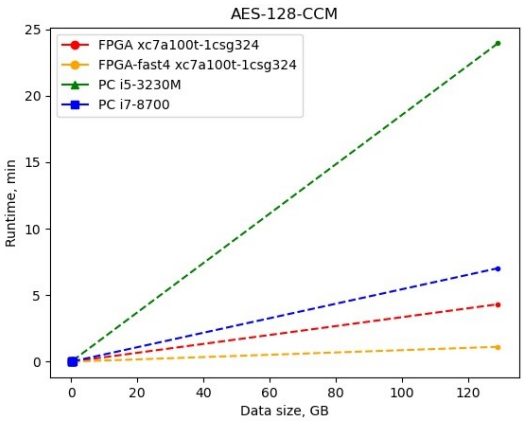


(a)

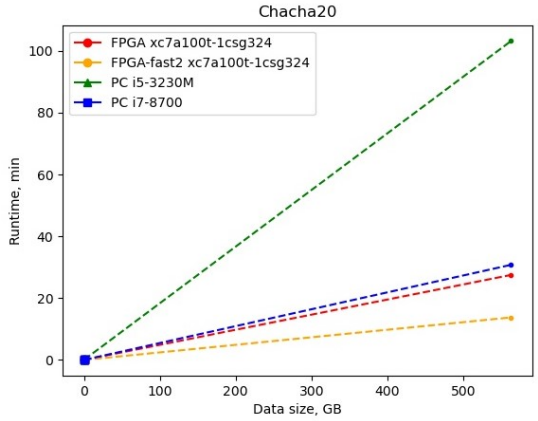


(b)

Figure 7. Comparison of AES-GCM encryption results for a data volume of 129 GB: (a) AES-128-GCM and (b) AES-256-GCM



(a)



(b)

Figure 8. Comparison of AES-128-CCM encryption results (a) for a data volume of 129 GB and Chacha20 (b) for a data volume of 563 GB

Finally, using formula (1), we can calculate the most important characteristic of devices: the price-quality ratio. The calculation results are shown in Table 9.

Table 9. Value for money devices

	Core i5-3230M	Core i7-8700	xc7a100t-1csg324
AES-128-CBC	0,0675	0,0874	0,1837
AES-256-CBC	0,0614	0,0795	0,1345
AES-128-GCM	0,0843	0,1558	0,7552
AES-256-GCM	0,0829	0,1496	0,5489
AES-128-CCM	0,0539	0,0605	0,7633
Chacha20	0,0546	0,0603	0,2707

4. CONCLUSIONS

The main results of this work are shown in tables 2-7 and 9. According to these results, we can conclude that the performance of the Core i7-8700 device turned out to be higher than the performance of the xc7a100t-1csg324 FPGA device, only for the AES-256-CBC protocol . In all other cases, we see that the FPGA in terms of data encryption speed has a huge advantage over both Core i5-3230M and Core i7-8700 processors. The relative poor performance of the FPGA for the AES-256-CBC protocol is due to the fact that this protocol is difficult to parallelize due to the fact that the encryption of the next data block uses the results of encryption of the previous data block.

In addition to comparing the absolute performance values of these devices, they were also compared according to the parameter: the price-quality ratio. Here, the FPGA turns out to be the undisputed and by a wide margin leader (see Table 9). In total, we conclude that the data generator, which is an array of pairs of original and encrypted information, is best implemented on an FPGA.

Thus, we come to the conclusion that for the implementation of high-speed data encryption using protocols of the AES family, it is best to use FPGAs. It is obvious that the FPGA gains a performance advantage over the CPU due to the ability to perform encryption operations using AES protocols in parallel mode at a high level. In addition, the positive effect of the use of FPGAs is achieved not only by obtaining a higher speed of execution of operations, but also due to lower cost compared to the CPU. It is clear that when using more expensive FPGA models, even higher performance can be achieved, but at the same time, the price-quality ratio will deteriorate slightly.

It should be noted that virtually all the conclusions of the article were obtained from the results of experimental studies, where specific computing devices were used. In this regard, all obtained graphs of the dependence of the calculation time on the amount of data and other numerical values given in tables 2-7 and 9 are correct only for these devices. In the case of comparing the performance of other computing devices, it is possible to obtain other results that do not coincide or contradict some of the conclusions of this work. Nevertheless, we believe that the main conclusion made in this work, that data encryption algorithms perform well in parallel mode, remains unchanged.

ACKNOWLEDGMENTS

This research has been funded by Nazarbayev University under research program #0054/ПЦФ-НС-19,PI Zhenisbek Assylbekov.

REFERENCES:

[1] 1.Sooyong Jeong, Cheolhee Park, Dowon Hong, Changho Seo, Namsu Jho, "Neural Cryptography Based on Generalized Tree Parity Machine for Real-Life Systems", Security and Communication Networks, vol. 2021, Article ID 6680782, 12 pages, 2021. https://doi.org/10.1155/2021/6680782

[2] Al-Jammas, Mohammed & Othman, Karam. (2011). Implementation of neural-cryptographic system using fpga. Journal of Engineering Science and Technology. 6. 411-428.

[3] Ameen, Siddeeq Y., Fadhil M. Almusailkh and Dr. Mohammed H. Al-Jammas. "FPGA Implementation of Neural Networks Based Symmetric Cryptosystem." 6th International

Conference: Sciences of Electronic, Technologies of Information and Telecommunications May 12-15, 2011.

[4] Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang and Huazhong Yang. 2017. [DL] A Survey of FPGA-Based Neural Network Inference Accelerator. ACM Trans. Recong. Technol. Syst. 9, 4, Article 11 (December 2017), 26 pages

[5] Bouguezzi, S.; Fredj, H.B.; Belabed, T.; Valderrama, C.; Faiedh, H.; Souani, C. An Efficient FPGA-Based Convolutional Neural Network for Classification: Ad-MobileNet. Electronics 2021, 10, 2272. https://doi.org/10.3390/electronics10182272

[6] Volna, E., Kotyrba, M., Kocian, V., & Janosek, M. (2012). Cryptography Based On Neural Network. ECMS 2012 Proceedings edited by: K. G. Troitzsch, M. Moehring, U. Lotzmann (pp. 386-391). European Council for Modeling and Simulation. doi:10.7148/2012-0386-0391

[7] I. Tsmots, V. Rabyk, O. Berezky, Y. Lukaschuk and V. Teslyuk, "Development Of Modules Of Neuro-Like Cryptographic Encryption And Decryption Of Data And Their Implementation On FPGA," 2021 IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), 2021, pp. 53-57, doi: 10.1109/CADSM52681.2021.9385228.

[8] Serikov, T., Zhetpisbayeva, A., Akhmediyarova, A., Mirzakulova, S., Kismanova, A., Tolegenova, A., W jcik, W. (2021). City Backbone Network Traffic Forecasting. International Journal of Electronics and Telecommunications, 67 (3), 319–324. doi: https://doi.org/ 10.24425/ijet.2021.135983

[9] Sarinova, A., Bekbayeva, A., Dunayev, P., Sarsikeyev, Y., Sansyzbay, K. Hyperspectral image compression algorithms for phytosanitary inspection of agricultural crops in aerospace photography // Journal of Theoretical and Applied Information Technologythis link is disabled, 2021, 99(24), стр. 6280–6290.

[10] https://www.intel.ru/content/www/ru/ru/products/sku/126686/intel-core-i78700-processor-12m-cache-up-to-4-60-ghz/specifications.html

[11] https://www.digikey.com/en/products/detail/xilinx-inc/XC7A100T-1CSG324C/3925803

[12] https://pycryptodome.readthedocs.io/en/latest/src/installation.html

[13] Dworkin M.J. Sp 800-38a 2001 edition. recommendation for block cipher modes of operation: Methods and techniques. – 2001.

[14] Vahid Nasir & Farrokh Sassani. A review on deep learning in machining and tool monitoring: methods, opportunities, and challenges//The International Journal of Advanced Manufacturing Technology, V.115. – 2021, pp 2683–2709. DOI: 10.1007/s00170-021-07325-7

[15] Dworkin M.J., Barker E.B., Nechvatal J.R., Foti J., Bassham L.E., Roback E., Dray Jr J.F. (Advanced encryption standard (AES). – 2001.

[16] Bernstein D.J. et al. ChaCha, a variant of Salsa20 //Workshop record of SASC. – 2008. – Vol. 8. – №. 1. – P. 3-5.

[17] Yakubova, M., Serikov, T. (2021) Development and imitating modeling in the developed network consisting of several knots removed among themselves on NetCracker 4.1// International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, EDMthis link is disabled, 2016, 2016-August, pp. 210-213, 7538726

[18] Manankova, O.A., Yakubov, B.M., Serikov, T.G., Yakubova, M.Z., Mukasheva, A.K.(2021) Analysis and research of the security of a wireless telecommunications network based on the IP PBX asterisk in an Opnet environment//Journal of Theoretical and Applied Information Technology, 2021, 99(14), pp. 3617–3630.