Research of algorithms for communication encryption

Milena N. Karova, Gergana E. Todorova, Mariana G. Todorova, Ivaylo P. Penev, and Ventsislav G. Nikolov

***Abstract*—**An application for research of algorithms for cryptographic secure data transmission by using the algorithms DES, TripleDES-128, TripleDES-192, AES-128, AES-192 и AES-256 is developed in C#. It offers to the user access to the transmitted data via password. Testing of the application is carried out to ensure trouble-free operation. The used resources and time required for encryption of groups of files with different length are studied. A comparative analysis of the algorithms is done and corresponding conclusions are made.

***Keywords*—**Communication encryption, Comparative analysis, Cryptography, Decryption.

I. Introduction

WITH the fast growth of universal electronic connectivity, people from around the world have a wide access to the information available from different sources. The need for confidential and security information increases. The data and resources must be protected from disclosure and security computer communication systems must be protected from network based attacks.

From ancient times different cryptography algorithms and methods have been employed to encrypt and hide valuable information from eavesdroppers [1]. The algorithms in cryptography are categorized into two classes: Symmetric and Asymmetric algorithms. In symmetric encryption both the sender and receiver share the same secret key [4]. The presented study compares symmetric algorithms. The security level of an encryption algorithm is measured by the size of key space [5]. Large key size means greater security but may decreases encryption or decryption speed. The larger the key space the more possible keys can be constructed. The strength of encryption schemes relies on the key secrecy, length of the key, the initialization string and how they all work together [2].

The main objective of presented work is evaluating the suitability of 6 symmetric popular algorithms. The implementation of algorithms is developed in the software application. It compares performance and resource requirements of encryption of communication and study of different secret key cryptographic algorithms.

The application is designed in C # for use cryptographic secure transmission of data using algorithms DES, TripleDES-128, TripleDES-192, AES-128, AES-192 and AES-256. It offers the user access to password information transmitted. To ensure trouble-free operation is carried out testing of the application and its components. As resources and time required for encryption algorithms groups of files with different length are used.

It needs to define the experimental topics on which it can evaluate and compare encryption schemes. The essential performance parameters are: average time for encrypting/decrypting, length of source file encrypting, size of secret key and resources used during file encryption [6, 8].

The developed application allows comparative analysis of various algorithms for communication encryption on various parameters. The analysis is depicted with a large number of figures, summarized data and tables.

II. Development Of Software Application For Research Of Algorithms For Communication Encryption

A. Review stage

The application for research of algorithms for cryptographic secure data transmission is developed using C# and .Net Framework version 4.5. The application is composed of two programs – server (Fig. 3) and client (Fig. 5). Their UML diagrams are shown in Fig. 1 and Fig. 2.

The work presented in this paper was supported within the Startup Scientific Project № 1, 2015, Technical University of Varna, "Research and development of algorithms for control of mobile robots under extreme conditions in virtual reality".

M. Karova is with the Department of Computer Science and Technology, Technical University of Varna, Bulgaria (e-mail: mkarova@ieee.bg).

G. Todorova is student at the Department of Computer Science and Technology, Technical University of Varna, Bulgaria.

M. Todorova is with the Department of Automation of Manufacturing, Technical University of Varna, Bulgaria (e-mail: mgtodorova@tu-varna.bg).

I. Penev is with the Department of Computer Science and Technology, Technical University of Varna, Bulgaria (e-mail: ivailo.penev@tu-varna.bg).

V. Nikolov is with the Department of Computer Science and Technology, Technical University of Varna, Bulgaria (e-mail: v.nikolov@tu-varna.bg).

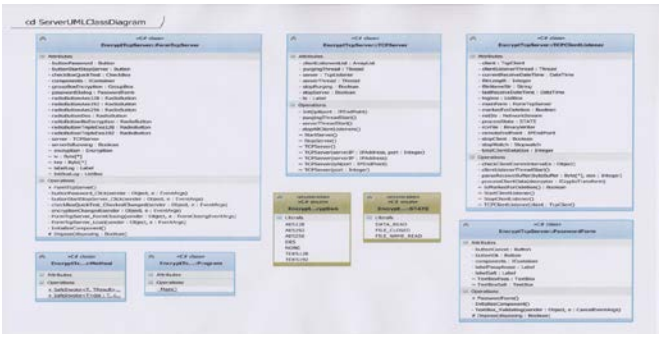

Fig. 1 server UML diagram

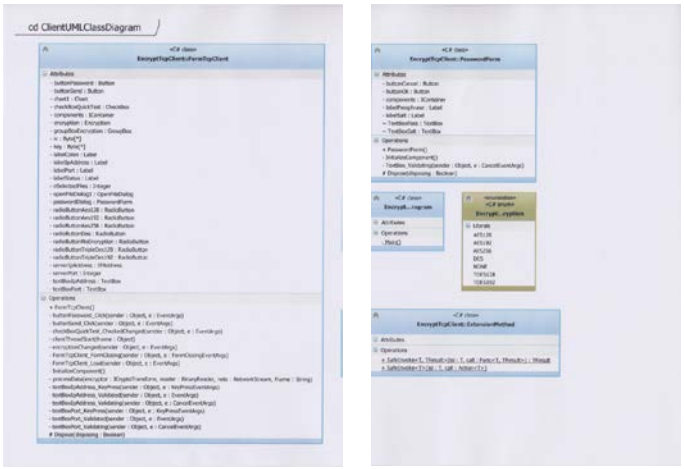

Fig. 2 client UML diagram

After starting the server passphrase and "salt" are set. They have to be strings with a length of at least 8 characters. A cryptographic salt is random data that is used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks versus a list of password hashes and against pre-computed rainbow table attacks. A new salt is randomly generated for each password. In a typical setting, the salt and the password are concatenated and processed with a cryptographic hash function, and the resulting output (but not the original password) is stored with the salt in a database. Hashing allows for later authentication while protecting the plaintext password in the event that the authentication data store is compromised. Cryptographic salts are broadly used in many modern computer systems, from Unix system credentials to Internet security.

From a security perspective, it is desirable for “salt” to contain uppercase and lowercase letters, numbers and symbols. There are built-in passphrase and “salt” that can be used for quick test of the application.

On the basis of the passphrase and the "salt" are derived a key and an initialization vector to be used for decryption of the received data. For this purpose, the class Rfc2898DeriveBytes of System.Security.Cryptography library is used, which implements PBKDF2 [6], by using SHA-1 [11] based generator.

There is a possibility of using algorithms AES (128-bit, 192-bit and 256-bit key), DES and Triple DES (128-bit and 192-bit key). After starting the server expects to receive files with encrypted data, decrypts and stores them in a folder on the disk. The file names, their length and the time for decryption are written in the server’s log. Methods of the Microsoft .NET System.Security.Cryptography library are used for decryption.

Upon receipt of a connection request from the client, the server starts a thread in which the receiving and decryption of data is done. After completion of the exchange of data, or after a period of idle time (15 seconds), the connection with the client is closed.

The client requires to be entered a passphrase and "salt" or to be used the built-in (for quick test). The method of encryption has to be the same as the one selected in the server.

B. Encrypted TCP server

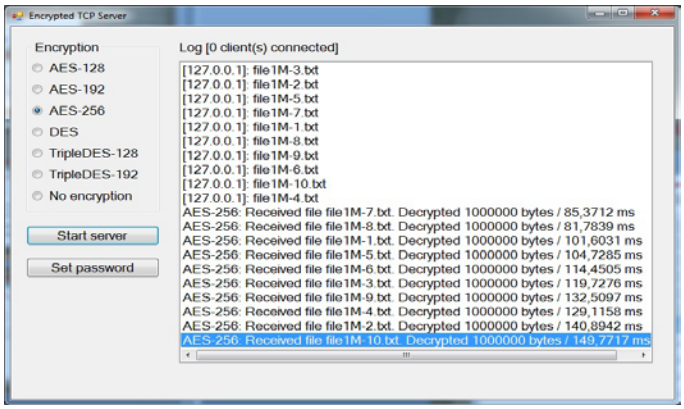

Fig. 3 server’s main window

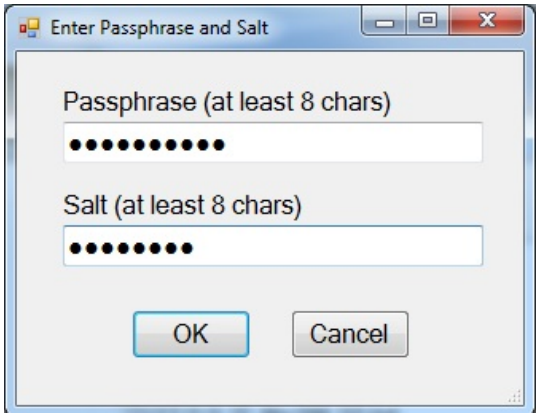

Fig. 4 dialog box for entering passphrase and “salt”

C. Encrypted TCP client

Pressing the button "Send file" opens a dialog box that allows you to choose one or several files. For each of the selected files a separate thread is started that connects to the server. Then the client sends to the server a block of 256 randomly generated bytes encrypted with the selected encryption algorithm and starts the encryption and transmission of the file. The purpose of the block of 256 randomly generated bytes is to increase security. Thus, even if the same file is transmitted repeatedly, every time the encrypted data will look different. After the transfer of all files, the screen displays a graph showing the time to encrypt each

file and the average time for encryption.

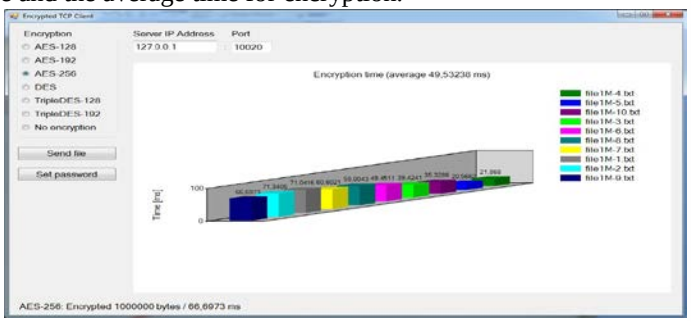

Fig. 5 client's main window

III. Research Of Algorithms For Communication Encryption

To compare the algorithms groups of files with four different lengths (10,000 bytes; 100,000 bytes; 1,000,000 bytes and 10,000,000 bytes) is done. The tests were performed with minimal use of computer resources from other processes.

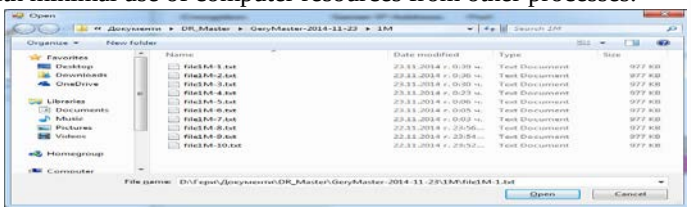

Fig. 6 test files with fixed size: 10K, 100K, 1M, 10M

The encryption is done by using each of the selected algorithms. Some of the obtained results are shown in the figures below.

A. Encryption time testing

The encryption time of 10 files with length 10,000 bytes using AES-128, AES-192 and AES-256 algorithms is shown on the following figures. The considered algorithms have similar average encryption time. The difference is about 0.02 ms.

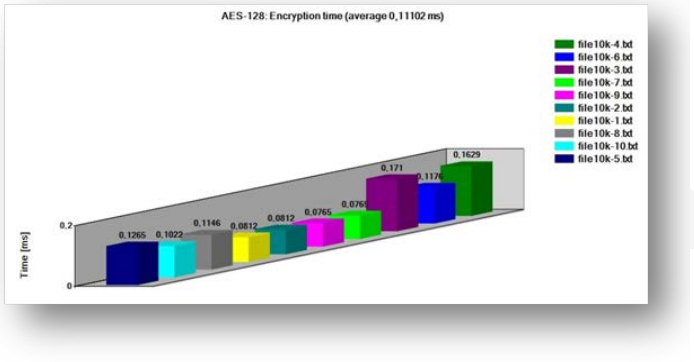

Fig. 7 AES – 128 Encryption time (average 0,11102 ms)

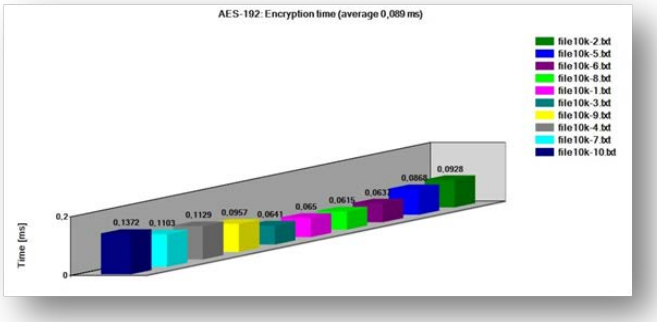

Fig. 8 AES – 192 Encryption time (average 0,089 ms)

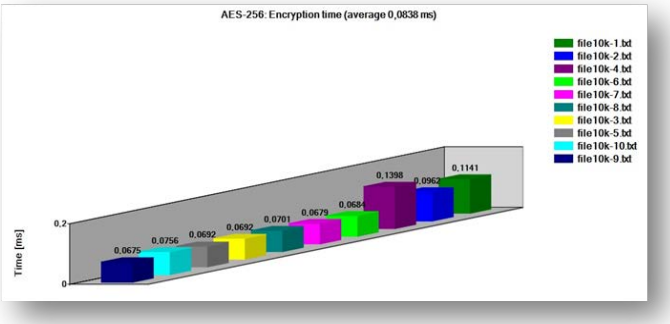

Fig. 9 AES – 256 Encryption time (average 0,0838 ms)

On Fig. 10 – Fig. 12 are shown the obtained results of encrypting the same files using DES, Triple DES -128 and Triple DES – 192. It makes an impression that the encryption speed is considerably slower than the speed of AES (about 9 times).

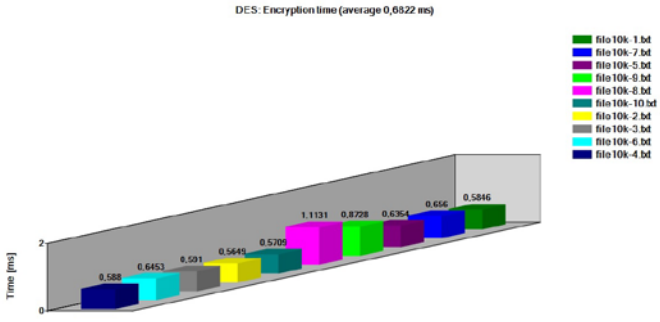

Fig. 10 DES Encryption time (average 0,6822 ms)

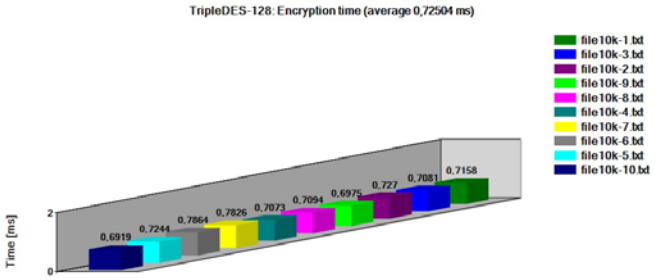

Fig. 11 TripleDES - 128 Encryption time (average 0,72504 ms)

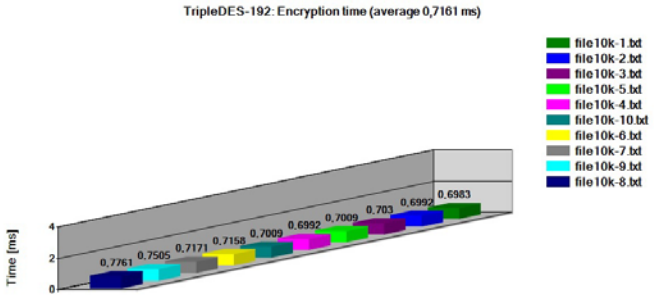

Fig. 12. TripleDES - 192 Encryption time (average 0,7161 ms)

A comparison of the encryption time of files of the same length using different algorithms (Fig. 13 – Fig. 16) and a comparison of the average time to encrypt files of different length in each of the algorithms (Fig. 17 – Fig. 22) is made. The obtained results are shown in the following figures:

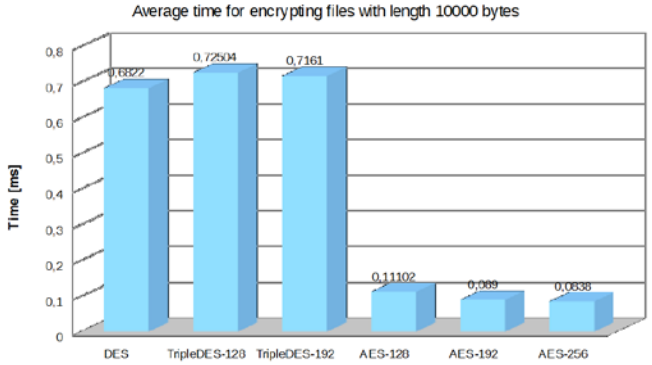

Fig. 13 average time for encrypting files with length 10,000 bytes

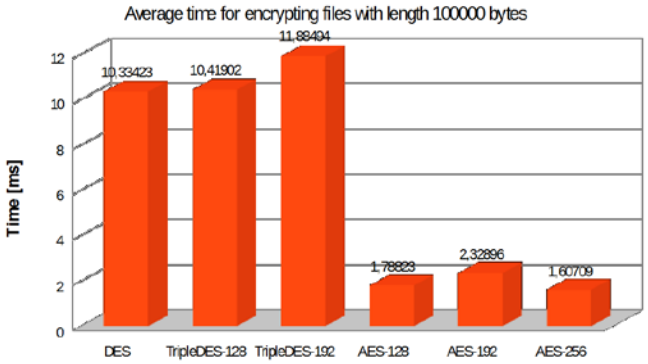

Fig. 14 average time for encrypting files with length 100,000 bytes

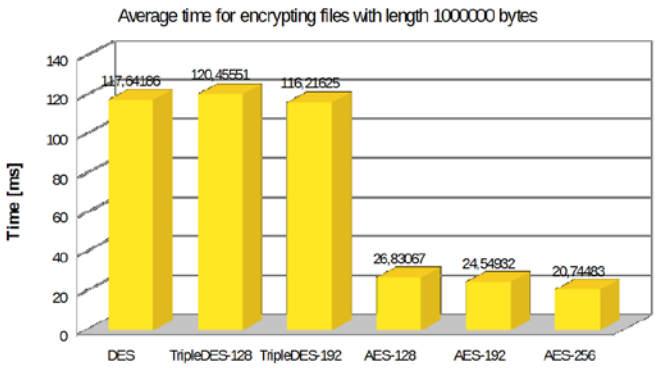

Fig. 15 average time for encrypting files with length 1,000,000 bytes

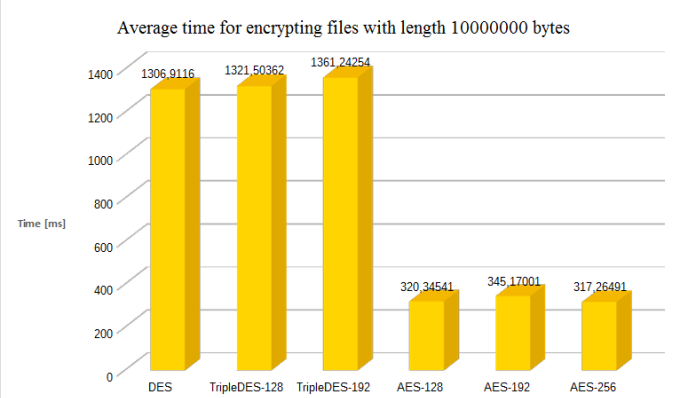

Fig. 16 average time for encrypting files with length 10,000,000 bytes

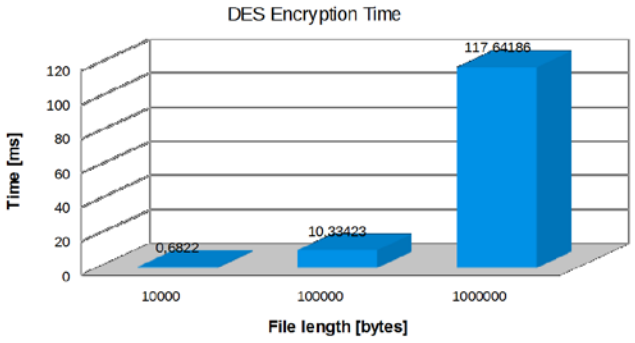

Fig. 17 DES Encryption time

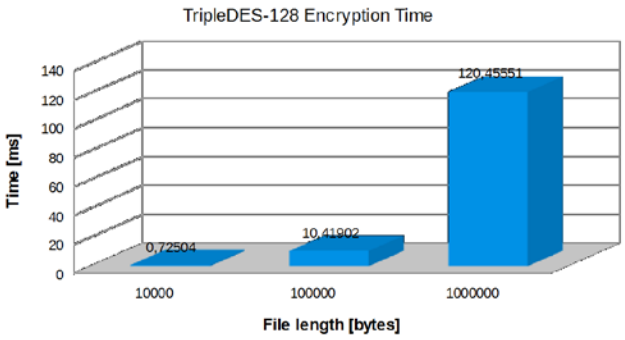

Fig. 18 TripleDES-128 Encryption time

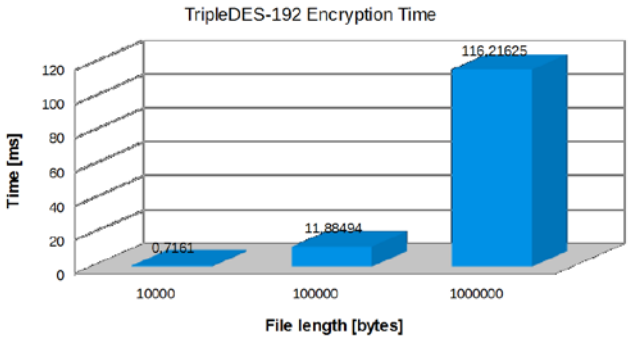

Fig. 19 TripleDES-192 Encryption time

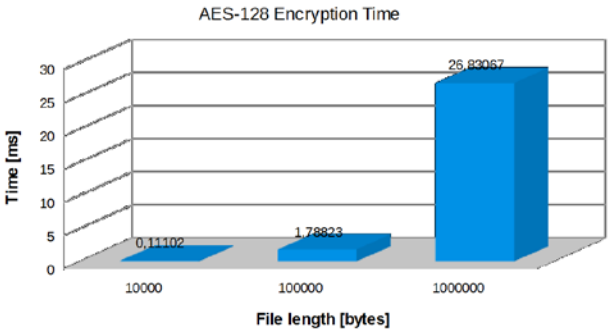

Fig. 20 AES-128 Encryption time

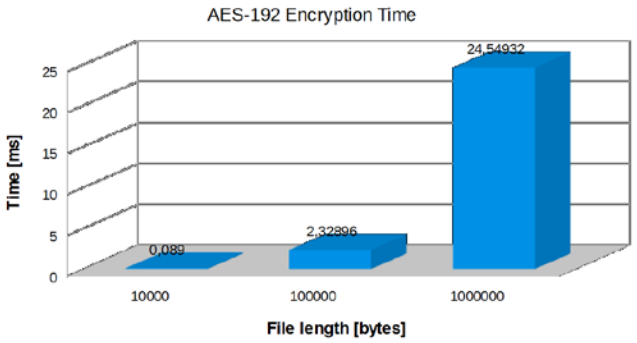

Fig. 21 AES-192 Encryption time

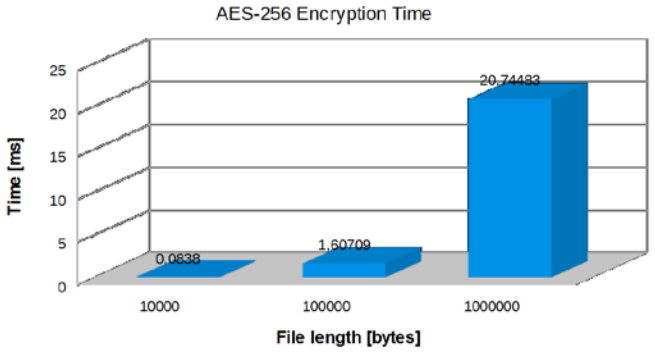

Fig. 22 AES-256 Encryption time

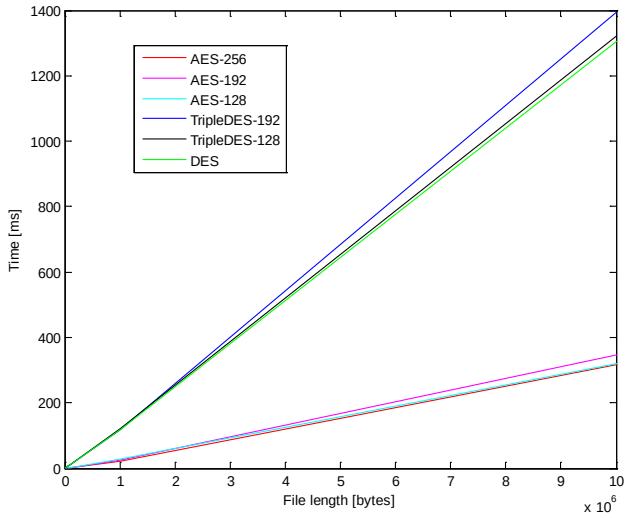

Fig. 23 influence of the file length and algorithm over the encryption time

The obtained results of encryption of 10K, 100K, 1M and 10M files using the considered algorithms are given on Table I. The ratio of the encryption time $K=t_{max}/t_{min}$ is shown in the last column, where t_{max} – maximum time for encryption, t_{min} – minimum time for encryption.

TABLE I. AVERAGE TIME FOR ENCRYPTION [MS]

	DES	Triple DES-128	Triple DES-192	AES-128	AES-192	AES-256	K
10K	0,6822	0,7250	0,7161	0,1110	0,0890	0,0838	8,6
100K	10,3342	10,4190	11,8849	1,7882	2,3289	1,6071	7,3
1M	117,6419	120,4555	116,2163	26,8307	24,5493	20,7448	5,6
10M	1306,91	1321,50	1396,24	320,34	345,17	317,26	4,4

Note: The least time for encryption is highlighted in red and the highest – in blue.

B. Used resources

System resources are monitored during the tests of groups of files with size 10 M. Summarized data achieved by Resource monitor are given below (Fig. 24 – Fig. 31).

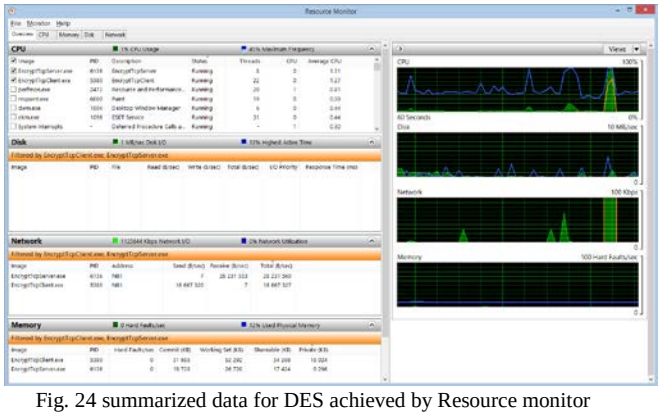

Fig. 24 summarized data for DES achieved by Resource monitor

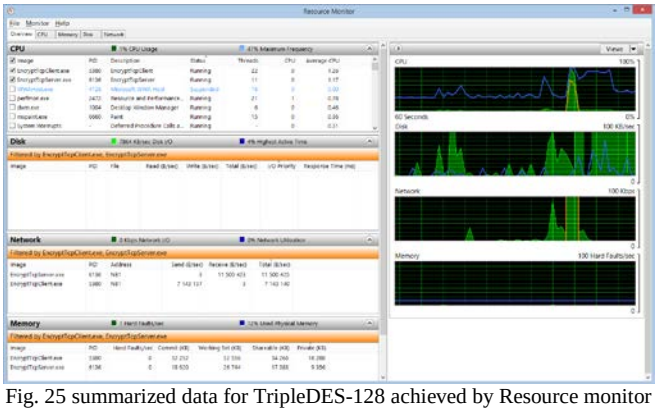

Fig. 25 summarized data for TripleDES-128 achieved by Resource monitor

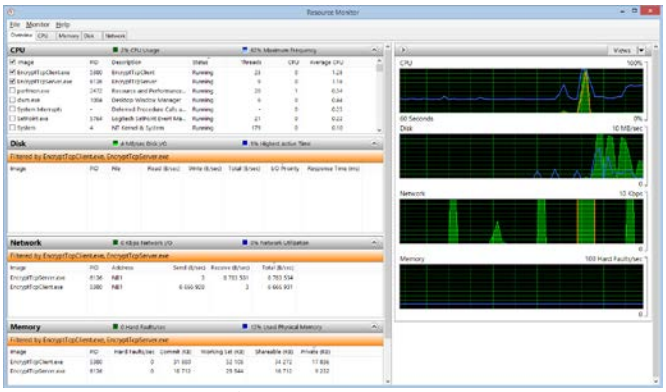

Fig. 26 summarized data for TripleDES-192 achieved by Resource monitor

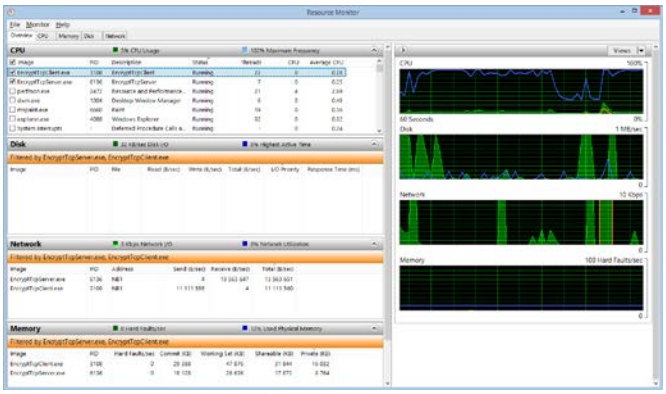

Fig. 27 summarized data for AES-128 achieved by Resource monitor

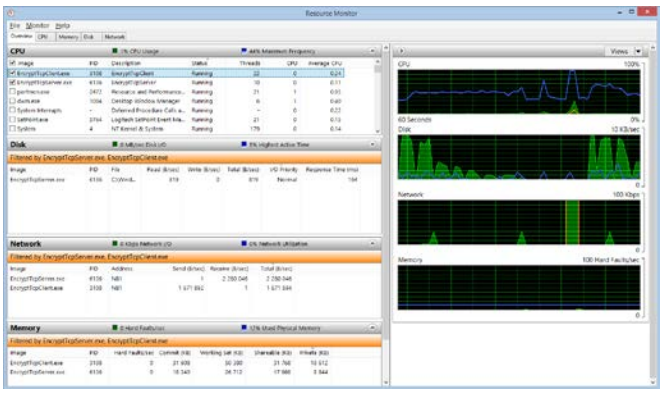

Fig. 28 summarized data for AES-192 achieved by Resource monitor

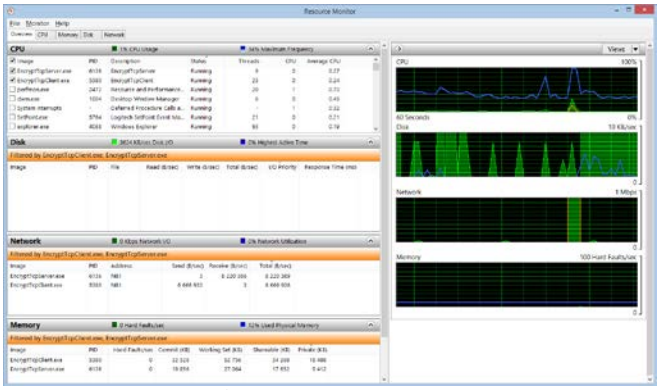

Fig. 29 summarized data for AES-256 achieved by Resource monitor

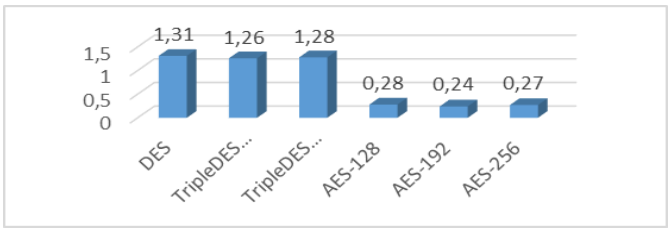

Fig. 30 average server CPU load

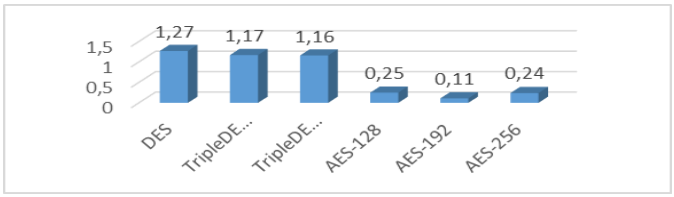

Fig. 31 average client CPU load

IV. Conclusions and Future Work

The developed application allows for testing and analyzing the considered algorithms. The flexible architecture of the application allows easily add new features that make it even more attractive to users.

In the paper, we aim to practically compare the working of the 6 algorithms mentioned above. We do so by comparing the used resources and time required for encryption of groups of files with different length.

As a result of the tests, the following conclusions can be drawn:

- AES-256 is the fastest encryption method, regardless of the file size. Slightly slower are AES-192 and AES-128.
- Triple DES has low encryption speed. Encryption of files is faster using DES than using Triple DES.
- It makes an impression that with increasing the file size, the speed of AES-256 relatively decreases comparing to the slowest algorithm (Triple DES). Table I shows that:

- Encryption of small files (10,000 bytes) is performed about 9 times faster using AES-256 than using Triple DES-128;
- Encryption of files of size 100,000 bytes is carried out about 7 times faster using AES-256 than using Triple DES-192;

- Encryption of files of size 1,000,000 bytes is performed about 5 times faster using AES-256 than using Triple DES-128;
- Encryption of large files (10M) is performed about 4 times faster using AES-256 than using Triple DES-192;
- The average CPU load of the server and the client is significantly lower (5-6 times) when using the AES algorithm as compared to that when using DES and TripleDES. Differences in CPU load when using DES and TripleDES are minor.

The considered problem offers a future scope on work related to:

- Testing with real time voice encryption;
- Data encryption using all methods in parallel.

REFERENCES

[1] A. J. Meneses, *Handbook of Applied Cryptography*, CRC Press, 1996.
[2] H. Manzoor, P. Mittal, V. Kumar, "A Comparative Study of Cryptographic Algorithms", International Journal of Computer Science and Network, vol. 3, Iss. 3, ISSN: 2277-5420, 2014.
[3] L. Chin-Feng, "C# Based EEG Encryption System Using Chaos Algorithm", 1 st WSEAS International Conference on Complex Systems and Chaos (COSC'13), Mathematics and Computers in Science and Engineering, Series 11, ISSN: 2227-4588, ISBN: 978-1-61804-178-4, 2013.
[4] P. Christof, J. Pelzl, "Understanding Cryptography", *A Textbook for Students and Practitioners*. Springer, ISBN 978-3-642-04100-6, 2010.
[5] A. C. Henk, *Encyclopedia of cryptography and security*. Eindhoven University of Technology The Netherlands, Springer, ISBN-13: (HB) 978-0387-23473-1, 2005.
[6] B. Kaliski, "PKCS #5: Password - Based Cryptography Specification Version 2.0", RSA Laboratories, 2000.
[7] A. Luma, R. Bujar, *Relationship between fibonacci and lucas sequences and their application in symmetric cryptosystems*, 4th International Conference on Circuits, Systems and Signals, pp. 146–150, WSEAS Press, ISBN 978-960-474-208-0, 2010.
[8] C. Pfleeger, *Security in computing*, ISBN 0-13-185794-0, Prentice Hall PTR Prentice - Hall, Inc, 1997.
[9] R. Robles, T. Kim, "Communication Security for SCADA in Smart Grid Environment", WSEAS Conference in Advances in Data Networks, Communications, Computers, 2010.
[10] B. Schneier, "Applied Cryptography: Protocols, Algorithms and Source Code in C", Printed in the USA, ISBN 0-471-11709-9, p. 657, 1996.
[11] http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf