# A NEW SECURITY SOLUTION IMPLEMENTED BY THE USE OF THE MULTILAYERED STRUCTURAL DATA SECTORS SWITCHING ALGORITHM (MSDSSA)

*Alexandru Tăbuşcă*[1]

**Abstract**

*The present paper is a short extract from my PhD thesis and presents two different security field solutions: a conceptual one (an algorithm) and a practical one (a software application). I introduce the MSDSSA algorithm and a software application usable for encrypting different data files in the Microsoft Windows environment. The application was created as a new version of an application introduced at the time of my university graduation, upgraded to the second version for commercial use, upgraded to the third version at the time of some major modifications implementations in 2007 and now, finally, upgraded to its fourth version after the implementation of the MSDSSA algorithm.*

**Keywords: encryption key, cipher, code, methods and techniques for creating encryption results, attacks, cryptography, cryptanalysis, integration, security solutions, phishing**

This article is meant as a presentation of the use of a new algorithm for assuring a high level of security through the use of an encryption tool. This algorithm is capable of assuring the security of any type of data file used inside a computer network or on an independent computer, through its use at bit level.

For the first part of this paper I will present a short description of the MSDSSA algorithm works on a step-by-step approach.

The MSDSSA algorithm starts by decomposing the file chosen for encryption in a number of blocks, each with the same length as the others. I will use the letter B for marking the number of bits of such a block and the letter N for marking the number of different blocks made of B bits. In most real-life cases, there will be a "rest" of bits, comprising a number of bits smaller than the B number (this thing happens every time the file length, in bits, is not exactly a multiple of the chosen B number); this "rest" is called R and will remain unchanged throughout the entire encryption process that I propose. Taking into consideration the conditions the N number should be lower or equal to 2B. In order to obtain the encrypted file I have to generate the encrypted code for each block, code that is called "identifier". The main scheme used for generating such identifiers comprises for main stages:

1. Creation of individual identifiers for each block of B bits, the number of these identifiers being at most 2B;

[1] Romanian-American University, School of Computer Science for Business Management

2. Obtaining all possible combinations among the bits of a block, thus being able to have all the possible distinct blocks for the first iteration of the encryption process;
3. Generating new identifiers for the blocks of B bits created from the identifiers obtained before;
4. The generation of new identifiers for the blocks of B bits formed from identifiers obtained at the previous step takes place up to a L level, established by the one that initiates the encryption and saved within the encryption key.

**Comparison teste: msdssa – t-des**

In order to sustain the relevancy of the MSDSSA algorithm for the field of data security through encryption methods I consider that an important step is to undertake an analytical comparison between results obtained after the encryption of a set of different files with both MSDSSA and another relevant and widely used algorithm. One of the most relevant algorithms for such a comparison I considered the T-DES algorithm (also known as "triple-DES").

| *File to be encrypted* | *File size (bytes)* |
|---|---|
| *Text1.txt* | *14337* |
| *Text2.txt* | *73710* |
| *Test.java* | *48430* |
| *Test.class* | *131776* |
| *Test.cpp* | *9029* |
| *Test.com* | *18432* |
| *Honey.jpg* | *14952* |
| *Honey.bmp* | *114688* |

**Figure 1.** Specifications of the files chosen for comparison tests between MSDSSA and T-DES algorithms

The most suitable comparison I consider the one based on the real time necessary for the two algorithms to produce the result, the encrypted files. After carefully experimenting with all the files presented above, in Figure 1, I can summarize the following situation:

| *File to be encrypted* | *File size (bytes)* | *MSDSSA encryption time(seconds)* | *T-DES encryption time (seconds)* |
|---|---|---|---|
| *Text1.txt* | *14337* | *0,109* | *2,101* |
| *Text2.txt* | *73710* | *0,824* | *10,788* |
| *Test.java* | *48430* | *0,549* | *7,098* |
| *Test.class* | *131776* | *1,373* | *19,148* |
| *Test.cpp* | *9029* | *0,109* | *1,167* |
| *Test.com* | *18432* | *0,219* | *2,381* |
| *Honey.jpg* | *14952* | *0,164* | *1,914* |
| *Honey.bmp* | *114688* | *1,208* | *14,898* |

**Figure 2**. Encryption speed comparison between MSDSSA and T-DES algorithms

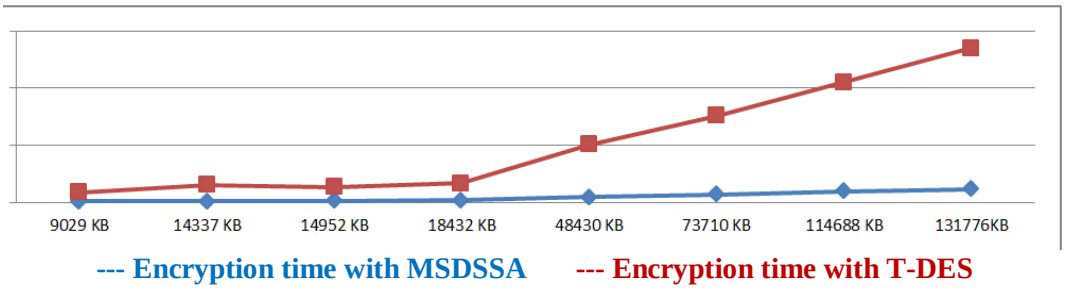


**Figure 3**. Graphical representation of the link between file size and encryption time for the MSDSSA and T-DES algorithms

In order to check the security level of the MSDSSA algorithm I have chosen a well-known and largely used test, specifically the verification of the homogeneity between the initial file and the final file obtained after the encryption process. The formula used to test this characteristic is the one known as the "chi-square test":

$$X^2 = \sum_{1}^{n} \frac{[(f]_0 - f_e)^2}{f_e}$$

For the specific case of the encryption algorithms, this mathematical formula uses the $f_e$ as the frequency of a character in the initial file and the $f_0$ value as the frequency of the same character inside the encrypted file. A greater X value is a good mark for the algorithm, showing a testimony of a more robust and powerful encryption solution. Taking into consideration the observations made for all the files tested above I can conclude that generally the chi-square value for the MSDSSA algorithm is higher, and thus better, than that of T-DES algorithm.

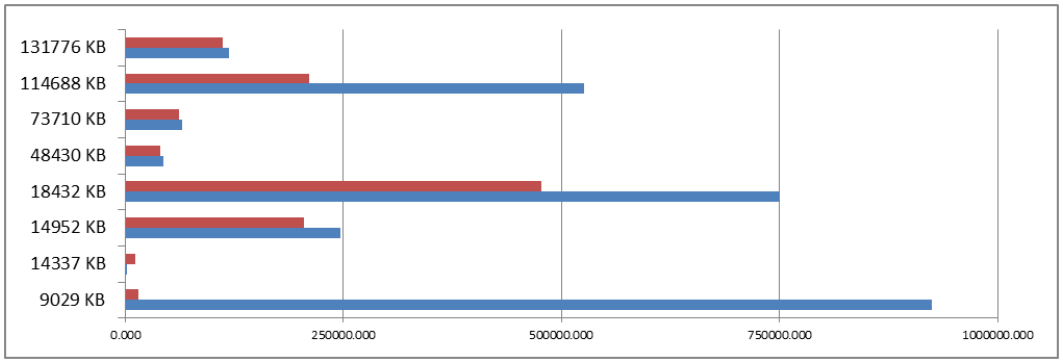


**Figure 4.** Graphical representation of the chi-square values calculated for both MSDSSA and T-DES algorithms

In conclusion, by using the new MSDSSA algorithm, we can encrypt any type of file, of any size, due to its implementation at bit level and we can do it by using a proven and secure way. Taking into consideration the fact that the number of iterations up to which the algorithm continues to generate new identifiers and the length in bits of the blocks are randomly chosen every time we start a new encryption session, we can conclude that the encryption key is practically unique for each encryption process.

## Implementation of an encryption solution based on the msdssa algorithm

### The EncryptDecrypt Application

The EncryptDecrypt application, as an integrated security solution that uses an implementation of the MSDSSA algorithm, is introduced and described in my PhD thesis[1]. For the development and implementation of the source code of this application I have chosen the Visual Basic development environment – introduced by Microsoft Corporation inside the Visual Studio Suite.

The EncryptDecrypt application can use as initial file that has to undergo the process of encryption any type of file, using its content in ASCII format as "clear text"; at decryption time the application practically rebuilds this ASCII content of the file, so that the internal format of the respective file is completely irrelevant for the encryption process. In case of an increased need for securing certain files the encryption solution can be used twice, using an already encrypted file as "clear text" for a new encryption process – a solution somehow similar with the T-DES case (triple DES). The EncryptDecrypt application has been created and designed to be a fast and efficient tools, capable of encrypting any message (data) by the use of a fast and secure algorithms.

### Final conclusion and future development directions for research and its practical applications of the results

Independent security applications, not related to the big developers of the market, with regular and often updates and with small area of usage have certain advantages and often provide a better and higher degree of protection against penetration risks, for short and medium periods of time in case of unclassified information protection. This fact is due not only to the solution's own quality, to the capabilities of the implemented algorithm, but also to the "failure" of the solution ☺. The less usage the solution has the less the risk of being targeted by a serious attack against it.

The next steps for my researches related to the problems presented inside this paper: the further development of the RSBS (Romanian Security Bulletin Solutions) application and the refining of the MSDSSA algorithm. The RSBS application will also comprise an online solution for security testing, the RSB-OSC (Romanian Security Bulletin – Online Security Checker). The RSB-OSC application applies, with the online user's specific approval, a questionnaire related to security issues and then launches a set of predetermined simulated attacks on the computer connected to the application's website. At the end of the simulated attacks the application compiles a report (with the possibility to have it emailed to the user) which first presents a "classification" of the user considering his answers at the questionnaire and then a statistical situation of the different security breaches that were detected or not during the simulated attack.
Another research topic for the future is the refining of the MSDSSA algorithm. First, I consider developing a new key distribution system because at this time, the encryption capabilities and the speed of the algorithm are competitive but MSDSSA is still tributary to an external mechanism for distributing the encryption/decryption keys.

**Bibliography**


1. Boneh D., Halevi S., Hamburg M., Ostrovsky R. – ”Circular-secure encryption from decision Diffie-Hellman”; published by: Springer - Proceedings of Crypto '08, 2008
2. Goldwasser S., Kalai Y.T., Rothblum G. – ”One-time programs”; published by: Springer - Proceedings of Crypto '08, 2008
3. Haitner I.; Holenstein T. – ”On the (im)possibility of key dependent encryption”; Published by: Springer - Proceedings of TCC '09, 2009
4. Hamalainen Antti, Tommiska Matti, Skytta Jorma – ”Gigabits per second implementation of the IDEA cryptographic algorithm”; published by: Springer - Verlag, 2002
5. Ishai Y., Paskin A. – ”Evaluating branching programs on encrypted data”; published by: Proceedings of TCC '07, 2007
6. Jarecki Stanislaw, Tsudik Gene – ”Public key cryptography – PKC 2009”; published by: Springer, 2009
7. Mogollon Manuel – ”Cryptography and security services: mechanisms and applications”; published by: Cybertech Publishing, 2008
8. Peltier R. Thomas – ”Information security policies and procedures: a practitioner's guide”; published by: CRC Press LLC, 2007
9. Stevenhagen P. – ”The arithmetic of number rings. Algorithmic number theory”; published by: MSRI Publications, volume 44, 2008
10. Tăbuşcă A. – ”Network security increase by using extended validation secure socket layer certificates for avoiding the phishing threats”; published by: DAAAM International - "Annals of DAAAM for 2009 & proceedings of 20th DAAAM international symposium 'Intelligent manufacturing & automation: theory, practice & education' ", ISI, 2009