DOI: https://doi.org/10.32070/ec.v3i47.89

Oleh Polihenko
PhD student,
National Aviation University
Ukraine, Kyiv
o.poligenko@ukr.net
ORCID ID: 0000-0002-2427-4976

Roman Odarchenko
Doctor of Science (Engineering),
Chair of Telecommunications and Radioelectronics Academic Department,
National Aviation University
Ukraine, Kyiv
odarchenko.r.s@ukr.net
ORCID ID: 0000-0002-7130-1375

Sergiy Gnatyuk
Doctor of Science (Engineering),
Vice dean of the Faculty of cybersecurity, computer and software engineering,
National Aviation University
Ukraine, Kyiv
s.gnatyuk@nau.edu.ua
ORCID ID: 0000-0003-4992-0564

ENTERPRISE INFORMATION SECURITY MANAGEMENT SYSTEM BASED ON THE MODERN OBFUSCATION TECHNIQUE FOR MOBILE NETWORK OPERATORS

Abstract. In today's realities, with constantly developing information technologies (IoT, 5G, Big Data, Cloud technologies, etc.), software protection is an urgent issue in the field of information security for each small, medium, or big enterprise. Also, software protection is a very important task for such enterprises, as mobile network operators, which, in order to ensure leadership in the market, produce a large number of modern unique software products for their own needs. Moreover, the software is the intellectual property of the enterprises, which developed it. The software is the intellectual property of both large corporations and small companies. The low reliability of software protection for enterprises is associated with a rather complex and time-consuming process, as well as with a number of technical limitations, which contributes to the thriving of computer piracy, inflicts colossal losses on IT companies and, of course, the state as a whole.

Therefore, the development of a new effective method of software protection, for the moment, is a priority in the field of information security, and new methods and techniques of software protection are needed for all specialized companies, which develop paid software. Nowadays there are many different approaches to solving this problem. These are encryption, watermarking, etc., but no one gives guaranteed results. That's why modern companies engaged in software development, should provide their customers with a more secure information product.

In this paper, the authors have provided the improved, more effective obfuscation method, based on a new sequence of obfuscation transformations. This method allows providing software protection of the enterprises from reverse engineering. To ensure the effectiveness of the proposed method, the authors have developed a special software product, based on cycles of operation and the creation of pseudocode to protect other software products. In the process of writing this article, studies were conducted that showed the following results. The product has become approximately 1.4 times more secure, and the obfuscation rate has increased by 10 percent. Based on the foregoing, the developed method can be followed to complicate the decoding process of existing software products used in various enterprises.

In the future, it is planned to implement additional obfuscation transformations, as well as a comparative analysis with existing obfuscation programs.

Keywords: enterprise, security management system, information security, obfuscation, software, secure coding

Formulas: 9, fig.: 2, tabl.: 2, bibl.: 27

JEL Classification: C02, L96

Introduction. Mobile network operators (MNO) become one of the most rapidly developing enterprises. Now we can observe the emergence of the first 5G commercial networks. 5G wireless technology is another game-changer for software development. More than any generation of wireless technology before it, 5G is a revolutionary upgrade. Things like autonomous vehicles, virtual and augmented reality, games, remote control devices and Internet of Things (IoT) deployment will force to use of novel approaches in software development. In this situation, it is also necessary to ensure the highest level of software product protection.

Software security against computer pirates and unauthorized users is actual problem during few decades. It makes serious damage to software engineering industry. This problem is amplified by speed multimedia and Internet technologies development because the quantity of ways to get non-license content grows every day. Modern world characterized by break of software every year, month and day. It costs billions of US dollars. The most serious attacks connected with code study and its hidden vulnerabilities detection which are common during software tools creation.

In the core of every software tools there is intellectual property of its developers. Software protection against unauthorized using, modifying and copying is the most important issue in modern information and communication systems. Computer piracy and illegal software using cause big damage for state economy particularly in hi-tech sector.

Development of effective security methods for codes is background and wall on the way of non-license products expansion. It is one of the main tasks for companies-developers as well as for state policy in IT sector.

Today there are many approaches to this problem solving. These are encryption, watermarking etc., but no one gives guaranteed result. Also, there are many obfuscation transformations in scientific and technical literature [Wang, Wu, Chen, Wei 2018] but detail obfuscation security methods description is absent. But this technology is one of prospect method to make hard illegal code study and modification. From this viewpoint, the development of new approaches and modification of existed obfuscation technologies is

actual task directed to growing efficiency of secure coding and protection against reverse engineering.

Modern companies, which develop a big number of software products (i.e. mobile network operators), should provide high-quality and secure information product to their customers. An important aspect of software development is to guarantee its reliability and integrity. Illegal intruders try to get source code and bypass the licensing stage. Therefore, it is necessary to protect not only the software overall as well as the source code. Obfuscation protection technique could be used to confuse program code, complicate the analysis and comprehension of operation algorithms furthermore preserving the functional program features. One of the main problems of software reliability is absence of the developed and implemented software protection methods, especially absence of software code protection from reverse engineering [Wang, Wu, Chen, Wei 2018; Stepanenko, Kinzeryavyy, Nagi, Lozinskyi 2016; Kuang, Tang, Gong, Fang, Chena, Wang 2018].

Literature review and the problem statement. Experience in recent years has shown that many IT companies around the world are facing challenges which require involvement of cybersecurity departments [Anderson, Choobineh 2008; Mayadunne, Park 2016; Yadegari, B., Johannesmeyer, Whitely, Debray 2015; Foket, De Bosschere, De Sutter 2019; Uchenna, Ani, He, Tiwari 2017]. Indirect attacks by cybercriminals spread pretty rapidly in terms of their number and threat scale, thus, affecting the quality of the product and sometimes leading to a complete shutdown of some enterprises, including government ones. This affects the reputation of the company or enterprise, and also leads to colossal losses of financial resources [Yadegari, B., Johannesmeyer, Whitely, Debray 2015; Foket, De Bosschere, De Sutter 2019; Shariati, Bahmani, Shams 2011]. In this case, the cybersecurity departments of companies / enterprises must possess the latest tools, understand the algorithm of these crimes and react immediately, prior to damage is caused [Sari 2015; Hu, Z., Gnatyuk, Sydorenko, Odarchenko, Gnatyuk 2016]. For example, there is a progressive project Industry 4.0 [Cafasso, Calabrese, Casella, Bottani, Murino 2020; Dechow, Granlund, Mouritsen 2006; De Smit, Elhabashy, Wells, Camelio 2016; Dzwigol, Dzwigol-Barosz, Miskiewicz, Kwilinski 2020; Dźwigoł, Shcherbak, Semikina, Vinichenko, Vasiuta 2019; Lu 2017; Miśkiewicz 2019; Miśkiewicz, Wolniak 2020], which helps enterprises to automate production in order to get the maximum productivity with minimal involvement of human labor [Wang, Wu, Chen, Wei 2018]. This software requires financial investments from the production side and constant maintenance. Of course, many will try to get such software for free, which will lead to the loss of the reputation of the IT company which produces an unprotected product, as well as lead to financial losses in general. In this case, this can be prevented by ultra-precise encryption, which will simply be impossible for a cybercriminal to bypass using conventional algorithms. Software security significantly increases the confidence of enterprises and gives the company an impeccable reputation at all possible levels [Rangel 2019; Zeng, Koutny 2019; Granlund, Mouritsen 2003]. Such an IT company will have better financial capabilities with those advantages for the development of further modern technologies and software.

The analysis of modern approaches to obfuscation was carried out [Stepanenko, Kinzeryavyy, Nagi, Lozinskyi 2016; Kuang, K., Tang, Gong, Fang, Chena, Wang 2018] and it has showed a lot of attention from researchers to the basic obfuscation process requirements, described categories of obfuscation distribution transformations and presented obfuscation protection methods.

Modern obfuscation methods for secure coding were analyzed in review papers [Stepanenko, Kinzeryavyy, Nagi, Lozinskyi 2016; Jeet, Dhir 2016] and paper contains effective software security techniques [Kaur, Tomar 2018; Merhi, Ahluwalia 2019] with some elements and procedures of code obfuscation.

However, a complex mechanism for source code protection that uses most of the known obfuscation transformations is absent.

Consequently, to decrease the probability of the reverse engineering process implementation it is important to develop a reliable technique of obfuscation protection of executable software files.

The main purpose of research is to create a reliable obfuscation technique for software security of the MNO enterprises that provides program code protection against the reverse engineering process.

To achieve purpose the following tasks were completed:

- new obfuscation technique of enterprises software protection was developed;
- software tool for source code protection was created;
- experimental study of created software tool was carried out.

Next parts of paper contain theoretical background and experimental study of proposed obfuscation technique.

Research results. The obfuscation technique called StiK was developed based on a new sequence of obfuscation transformations to solve mentioned problems [Stepanenko, Kinzeryavyy, Nagi, Lozinskyi 2016].

The input data for this technique is:

- Source Code *A*.
- Obfuscation Code Structure Transformation $S=(S_1,....,S_6)$, S_i is one of the transformations: the restructuring of the arrays; the clone method; modification of loop conditions; the dead code method; use of mark «goto»; the parallel code method $i=\overline{1,6}$
- Obfuscation Transformation of Variables $V=(V_1,....,V_4)$, S_j, is one of the transformations: inheritance relations modification, splitting or merging of variables, huge variables, converting static data to procedure, $j=\overline{1,4}$.
- Obfuscation Punctuation Transformation P=(P1,P2), Pk, is one of the transformations: inversion of code elements, token removing $k=\overline{1,2}$.

In this technique, was determined the obfuscation sequence which must be accomplished to provide effective software protection from the reverse engineering process, to complicate the analysis process and comprehension of the program code algorithm.

Code Obfuscation Technique Description:

Stage 1. First, the source code A is loaded into a memory fragment. This code is divided into logical structures A=(A1,....,An), (Ax – the logical structure of the code A, $n\in N$, $x=\overline{1,n}$). The amount of the logical structures is determined by the program code size. Then logic units A_x, $x=\overline{1,n}$, are being transformed using some obfuscation structure transformations S_i, $i=\overline{1,6}$. The number of transformations for each logical structure is determined accidentally. The obfuscation transformation number for each A_x should be at least three. The final part is to combine all the elements into the code and to test its availability.

Stage 2. Next, the resulting code *B* was divide into logical structures B=(B1,....,Bm) (By– the logical structure of the code B, $m\in N$, $y=\overline{1,m}$). Logical structures By, $y=\overline{1,m}$ are being processed by obfuscation variable transformation V_j, $j=\overline{1,4}$ (the obfuscation transformation

number for each B_y should be at least two). Obtained transformed structures were combined into program code *C* and its availability was tested.

Stage 3. The resulting code *C* is being divided into logical structures $C=(C_1,....,C_g)$ (C_z – the logical structure of the code C, g∈N, z=$\overline{1,g}$). Next, logical structures Cz, z=$\overline{1,g}$ are being transformed by obfuscation punctuation transformations P_k, k= $\overline{1,2}$. As a result of these stages, the software code *D* is created and its availability is tested.

To provide more reliable software protection mechanism was proposed using more marks «goto» after each obfuscation stage of the StiK method. This item increases the stability index of the software code and the average difference index between the transformation code and the source code. To implement it the violation of program logic was used and development of the unreadable tangled code. The output software code is completely modified however the program preserves a whole functionality of the source code, presented on Figure 1. A pseudocode was developed for the StiK protection method. The basic procedures according to the scheme is presented on Figure 2.

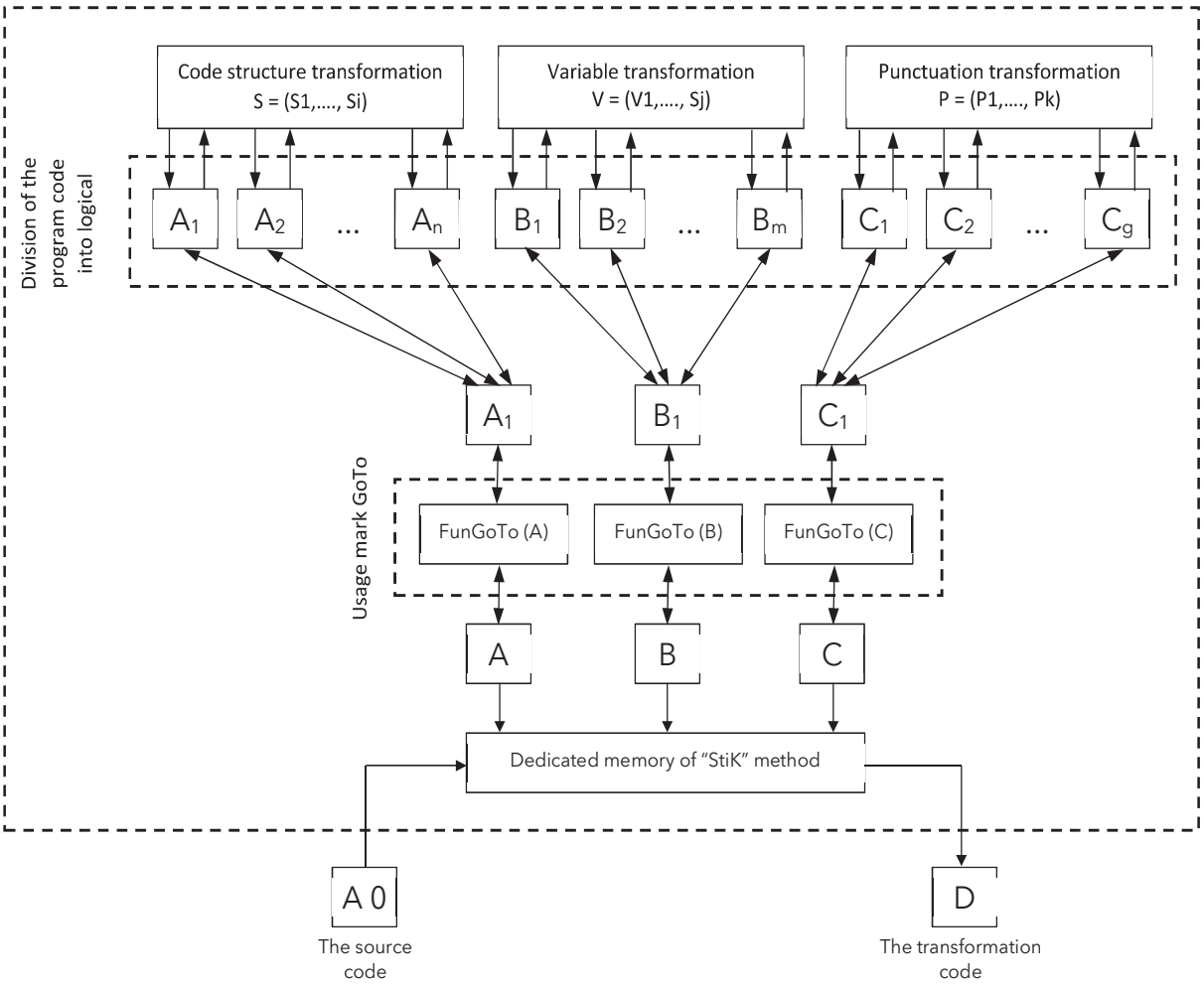

Figure 1 – Scheme of the obfuscation protection method
Source: *developed by the authors*

A procedure *OpenFile()* is used to download the program code. Apply to its input the file name that will be transformed.

A *FunRand()* is applied to generate a pseudorandom sequence whereby the corresponding indexes was obtained that will be used for the code transformation.

A procedure *DivFunction()* is used to separate the program code into logical structure.

CodeStructure(), *VariableFun()*, *PunctuationFun()* procedures are applied for each stage of obfuscation transformation. Data about logical structures and generated obfuscation transformation indices in appropriate procedures were entered.

A *FunGoTo()* is an unconditional branch instruction, which is applied several times in each transformation.

A procedure *AssociationF()* is used to combine separate logical structures into a single code.

A procedure *Cheking()* is applied for sanity check of the generated code.

A procedure *WriteFile()* saves the transformed code as a new file.

Input: *NameFile* is the file name with the source code, transformations *S, P, V.*
Output: NameFileNew is the file name with the transformation code.

1. A=OpenFile(NameFile);
2. $\{A_x\}$=*DivFunction(A),* $A=(A_1,....,A_n)$, A_x is the logical structure *A,* n∈N, x=$\overline{1,n}$
3. *for(x=1;x≤n;x++)*
   3.1 $\{A_x\}$=*FunGoTo(*$\{A_x\}$*)*
   3.2 for(x_1*=1;*x_1*≤3;*x_1*++)*
      3.2.1. {*i*}=*FunRand(S);*
      3.2.2. A_x=*CodeStructure(*A_x*,S,i),* $i\in\overline{1,6}$;
   3.3. $\{A_x\}$=*FunGoTo(*$\{A_x\}$)
   3.4. B=*AssociationF(*$\{A_x\}$*);*
4. $\{B_y\}$=*DivFunction(B),* $B=(B_1,....,B_m)$, B_y is the logical structure *B,* m∈N, y=$\overline{1,m}$;
5. *for(y=1;y≤m;y++)*
   5.1 $\{B_y\}$=*FunGoTo(*$\{B_y\}$*)*
   5.2 for(y_1*=1;*y_1*≤2;*y_1*++)*
      5.2.1. {*j*}=*FunRand(V);*
      5.2.2. B_y=*VariableFun(*B_y*,V,j),* $j\in\overline{1,4}$;
   5.3. $\{B_y\}$=*FunGoTo(*$\{B_y\}$)
   5.4. *C=AssociationF(*$\{B_y\}$*);*
6. $\{C_z\}$=*DivFunction(C),* $C=(C_1,....,C_g)$, C_g is the logical structure *C,* g∈N, z=$\overline{1,g}$);
7. *for(z=1;z≤g;z++)*
   7.1. $\{C_g\}$= *FunGoTo(*$\{C_g\}$)
   7.2. for(z_1*=1;*z_1*≤1;*z_1*++)*
      7.2.1. C_z= *PunctuationFun(*C_z*,P,z);*
   7.3. $\{C_g\}$=*FunGoTo(*$\{C_g\}$)
   7.4. *D=AssociationF(*$\{C_z\}$*);*
8. *Cheking(D)*
9. *WriteFile(D, NameFileNew)*

Figure 2 – Pseudocode for the obfuscation protection method
Source: *developed by the authors*

Experimental Study and Discussion. The software tool was developed based on the submitted sequence of operations and created pseudocode for protection method. The

StiK software tool was created using C++ programming language, in the Visual Studio 2013 programming environment. Obfuscation transformations were defined from each category: the dead code method, usage of mark «goto», huge variables, token removing.

Experimental study was conducted to determine the speed characteristics of the obfuscation process and the stability of the software code to the reverse engineering process. The experimental two-part methodology was developed. The input data for these experiments was 10 files with source code, they were transformed using the developed software tool. Using this console tool and two-part experimental methodology the efficiency of StiK was studied in following manner:

1. Obfuscation rate was assessed (Experiment 1).
2. Security of code against reverse engineering was assessed (Experiment 2).

Experimental studies were fulfilled using computer system with processor Intel (R) Core (TM) i3-6100, 3.7GHz and 4 GB RAM based on 64-bit operation system Windows 7 Service Pack 1. Also all results were compared with similar results of well-known and effective SmartAssembly obfuscator (see Table 2).

Experiment 1. To realize this experiment input files size and transformed files size as well as time for obfuscation were fixed. Obfuscation rate was assessed using expression $v_i = O_i/t_i, i = \overline{1,10}$, where vi is obfuscation rate for i-th file; Oi is transformed i-th file size, ti is time for i-th file obfuscation.

For all assessed obfuscators the parameter of middle rate was calculated using expression $vcep = \sum_{i=1}^{n} v_i / n$. In Table 1 results of Experiment 1 are presented.

Table1 - Results of obfuscation rate assessing (Experiment 1)

File	1	2	3	4	5	6	7	8	9	10	Middle rate
File size, KB	4.09	0.90	5.18	2.68	3.08	1.34	1.51	1.18	0.72	3.55	
t, s Smart Assembly	0.1	0.008	0.14	0.012	0.029	0.042	0.028	0.016	0.038	0.027	
t, s StiK	0.103	0.038	0.12	0.058	0.061	0.033	0041	0.022	0.017	0.058	
File size, KB Smart Assembly	4.95	1.86	10.0	3.08	3.42	3.86	3.01	3.56	1.31	3.68	
File size, KB StiK	8.23	3.75	14.7	6.97	5.86	4.71	4.96	5.27	3.95	5.33	
Rate, KB/s Smart Assembly	49.5	202.5	75.5	256.6	118.9	91.7	107.3	205.6	34.1	136.3	127.8
Rate, KB/s StiK	82.9	99.7	132.6	138.1	96.1	162.7	125.9	239.5	232.4	95.9	140.6

Source: *developed by the authors*

In accordance to Table 1 StiK obfuscator is 10% faster than SmartAssembly.

Experiment 2. During this study 10 executed files were analyzed and these files were protected using SmartAssembly and StiK obfuscators. Coefficient of code growing was calculated using formula $k_i = CT_i / CP_i$, $i=1,10$, where is number of processes in transformed code, is number of processes in input code. Table 2 contains results of Experiment 2.

Table 2 - Results of security code against reverse engineering assessing (Experiment 2).

Categories	1	2	3	4	5	6	7	8	9	10	Middle rate
Number of processes in input code	14	4	39	22	16	17	31	8	5	28	18.4
Number of processes in StiK	55	19	383	58	38	123	157	81	49	106	106.9
Number of processes in Smart Assembly	47	26	187	39	67	84	85	46	32	79	69.2
Coefficient for STiK	3.93	4.75	9.82	2.64	2.38	724	5.06	1013	9.80	3.79	5.95
Coefficient for Smart Assembly	3.36	6.50	4.79	1.77	4.19	4.94	2.74	5.75	6.40	2.82	4.33

Source: *developed by the authors*

Conclusions. Industry 4.0 is a very progressive project, which aims to help enterprises to automate production in order to get the maximum productivity with minimal involvement of human resources. That's why, nowdays, with constantly developing information technologies, software protection is an urgent issue in the field of information security for the MNO enterprises. In this paper, new obfuscation method for software protection for enterprises was proposed and it was based on a new sequence of obfuscation transformations. It allows to provide software protection from the reverse engineering. Also, the software tool has been developed according to StiK method and experimental studies have been carried out.

As a result, authors found that the average difference index between the transformation code and the source code was 36.21% and the average speed characteristics for obfuscation process were 140.6 KB per second. In accordance to experimental results StiK obfuscator is 10% faster as well as 1.37 times more protected than analogues. This helps enterprises to ensure data protection from the different types of attacks. Therefore it will help to save investments and to increase the profit.

However, in the future, the implementation of more obfuscation transformations is planning and also comparative analysis with existing obfuscation programs will be carried out. On this basis, new techniques for increasing the information security of the enterprises will be developed.

References

Anderson, E. E., & Choobineh, J. (2008). Enterprise information security strategies. *Computers & Security*, 27(1–2), 22-29. https://doi.org/10.1016/j.cose.2008.03.002

Cafasso, D., Calabrese, C., Casella, G., Bottani, E., & Murino, T. (2020). Framework for Selecting Manufacturing Simulation Software in Industry 4.0 Environment. *Sustainability, 12*, 5909. https://doi.org/10.3390/su12155909

Danik, Yu., Hryschuk, R., & Gnatyuk, S. (2016). Synergistic effects of information and cybernetic interaction in civil aviation. *Aviation*, 20(3), 137-144. https://doi.org/10.3846/16487788.2016.1237787

De Smit, Z., Elhabashy, A. E., Wells, L. J., & Camelio, J. A. (2016). Cyber-physical security challenges in manufacturing systems. *Procedia Manufacturing*, 5, 1060-1074. https://doi.org/10.1016/j.promfg.2016.08.075

Dechow, N., Granlund, M., & Mouritsen, J. (2006). Management control of the complex organization: relationships between management accounting and information technology. *Handbooks of Management Accounting Research*, 2, 625-640. https://doi.org/10.1016/S1751-3243(06)02007-4

Dzwigol, H., Dzwigol-Barosz, M., Miskiewicz, R., & Kwilinski, A. (2020). Manager Competency Assessment Model in the Conditions of Industry 4.0. *Entrepreneurship and Sustainability Issues,* 7(4), 2630-2644. https://doi.org/10.9770/jesi.2020.7.4(5)

Dźwigoł, H., Shcherbak, S., Semikina, M., Vinichenko, O., & Vasiuta, V. (2019). Formation of Strategic Change Management System at an Enterprise. *Academy of Strategic Management Journal*, 18(SI1), 1-8.

Foket, Ch., De Bosschere, K., & De Sutter, B. (2019). Effective and efficient java-type obfuscation. *Journal of Software: Practice and Experience*, 50(2), 136-160. https://doi.org/10.1002/spe.2773

Granlund, M., & Mouritsen, J. (2003). Introduction: problematizing the relationship between management control and information technology. *European Accounting Review*, 12(1), 77-83. https://doi.org/10.1080/0963818031000087925

Hu, Z., Gnatyuk, V., Sydorenko, V., Odarchenko, R., & Gnatyuk, S. (2016). Cyber Stealth Attacks in Critical Information Infrastructures. *IEEE Systems Journal*, 12(2), 1778-1792. https://doi.org/10.1109/JSYST.2015.2487684

Henrie, M. (2015). Cyber Security Risk Management in the SCADA Critical Infrastructure Environment. *Engineering Management Journal*, 25(2), 38-45. https://doi.org/10.1080/10429247.2013.11431973

Jeet, K., & Dhir, R. (2016). Software Module Clustering Using Hybrid Socio-Evolutionary Algorithms. *International Journal of Information Engineering and Electronic Business*, 8(4), 43-53. https://doi.org/10.5815/ijieeb.2016.04.06

Kaur, J., & Tomar, P. (2018). Clustering based Architecture for Software Component Selection. *International Journal of Modern Education and Computer Science*, 10(8), 33-40. https://doi.org/10.5815/ijmecs.2018.08.04

Kuang, K., Tang, Z., Gong, X., Fang, D., Chena, X., & Wang, Z. (2018). Enhance virtual-machine-based code obfuscation security through dynamic bytecode scheduling. *Computers & Security*, 74, 202-220. https://doi.org/10.1016/j.cose.2018.01.008

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, 1-10. https://doi.org/10.1016/j.jii.2017.04.005

Mayadunne, S., & Park, S. (2016). An economic model to evaluate information security investment of risk-taking small and medium enterprises. *International Journal of Production Economics*, 182, 519-530. https://doi.org/10.1109/SP.2015.47

Merhi, M. I., & Ahluwalia, P. (2019). Examining the impact of deterrence factors and norms on resistance to Information Systems Security. *Computers in Human Behavior*, 92, 37-46. https://doi.org/10.1016/j.chb.2018.10.031

Miśkiewicz, R. (2019). Challenges Facing Management Practice in the Light of Industry 4.0: The Example of Poland. *Virtual Economics*, 2(2), 37-47. https://doi.org/10.34021/ve.2019.02.02(2).

Miśkiewicz, R, & Wolniak, R. (2020). Practical Application of the Industry 4.0 Concept in a Steel Company. *Sustainability*, 12(14), 5776. https://doi.org/10.3390/su12145776

Rangel, A. (2019). Why enterprises need to adopt 'need-to-know' security. *Computer Fraud & Security*, 2019(12), 9-12. https://doi.org/10.1016/S1361-3723(19)30127-7

Sari, A. (2015). Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications. *Journal of Information Security*, 6(2), 142-154. https://doi.org/10.4236/jis.2015.62015

Shariati, M., Bahmani, F., & Shams, F. (2011). Enterprise information security, a review of architectures and frameworks from interoperability perspective. *Procedia Computer Science,* 3, 537-543. https://doi.org/10.1016/j.procs.2010.12.089

Stepanenko, I., Kinzeryavyy, V., Nagi, A., Lozinskyi, I. (2016). Modern obfuscation methods for secure coding. *Ukrainian Scientific Journal of Information Security*, 22(1), 32-37. https://doi.org/10.18372/2225-5036.22.10451

Uchenna, P., Ani, D., He, H. M., & Tiwari, A. (2017). Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective. *Journal of Cyber Security Technology*, 1(1), 32-74. https://doi.org/10.1080/23742917.2016.1252211

Wang, P., Wu, D., Chen, Z., & Wei, T. (2018). Protecting million-user IOS apps with obfuscation: motivations, pitfalls, and experience. In ICSE-SEIP '18: *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. Association for Computing Machinery. New York, NY, United States. https://doi.org/10.1145/3183519.3183524

Yadegari, B., Johannesmeyer, B., Whitely, B., & Debray, S. (2015). A generic approach to automatic deobfuscation of executable code. *IEEE Symposium on Security and Privacy*, San Jose, CA, 674-691. https://doi.org/10.1109/SP.2015.47

Zeng, W., & Koutny, M. (2019). Modelling and analysis of corporate efficiency and productivity loss associated with enterprise information security technologies. *Journal of Information Security and Applications*, 49, 102385. https://doi.org/10.1016/j.jisa.2019.102385

Received: 20.06.2020
Accepted: 28.06.2020
Published: 31.07.2020