Specifics of modern security requirements for software of electronic machine control systems

Serhii F. KASHTANOV[1], Yury O. POLUKAROV*,[1], Oleksiy I. POLUKAROV[1], Liudmyla O. MITIUK[1], Nataliia F. KACHYNSKA[1]

*Corresponding author
[1]Department of Labor Protection, Industrial and Civil Safety,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37 Peremohy Ave., 03056, Kyiv, Ukraine,
kshtnv@gmail.com, polukarov@ukr.net*, polucarov_alex@ukr.net,
luda2010703@gmail.com, natalik12345@gmail.com

DOI: 10.13111/2066-8201.2021.13.S.9

Received: 11 March 2021/ Accepted: 22 June 2021/ Published: August 2021

Abstract: *The required level of safety of machines and mechanisms is achieved through the use of appropriate safety management systems for industrial equipment, including programmable electronic ones. Such systems usually include a variety of security devices for managing industrial equipment settings. Since electronic control systems are currently considered the most promising control systems in this area, the study of the security parameters of their application support determines the relevance of this study. This study analyses the main requirements of IEC 61508 and IEC 62061 standards for compliance with modern safety requirements of embedded and applied software for electronic control systems of machines and mechanisms. This study proposes an algorithm for step-by-step implementation of software for electronic machine control systems in accordance with basic security standards for both built-in and application software. Testing has been determined as the main method of verification of application software. Based on the results of the analysis, it was found that the specification of security requirements, both built-in and application software, should highlight the necessary characteristics of each subsystem, providing information that allows choosing the equipment that meets existing security requirements. Relevant recommendations are given on the specifics of practical application of these standards.*

Key Words: *standard, verification, parameterization, system configuration, software tools*

1. INTRODUCTION

During the production process, protective measures and equipment must be applied in a certain manner that meets the requirements of EN ISO 12100-1. This is necessary to eliminate hazards and reduce the level of possible risks of injury, as well as the occurrence of occupational diseases [1]. Dangerous areas can be protected by using protective fences, light barriers, emergency stopping devices, etc. Any technological operation to use any device must be constantly monitored, and if necessary (accident, failure, lack of power, etc.), the device must be brought to a safe state.

Key regulatory documents governing safety requirements at various stages of development, design, and operation of machines and mechanisms, including their safety management systems, include: Directive 2006/42/EC [2] and standards existing in this area, such as EN 954-1 (DSTU EN 954-1: 2003) [3], EN ISO 13849-1 (DSTU EN ISO 13849-1-2016) [4], IEC 62061 [5] and IEC 61508 [6]. Of importance are also the IEC 62061 and IEC 61508 standards, which were developed specifically for safety-related electrical, electronic, and programmable electronic control systems for machines and mechanisms. It should be emphasised that programmable electronic control systems are currently considered to be the most promising control systems for this industry [7], [8], [9].

In accordance with the requirements, if the software is to be used in parts related to the security of programmable electronic control systems (SPECS) that implement the security management functions (SMF) of machines and mechanisms, the specification of software security requirements should be developed and documented. Security requirements specifications for built-in and application software should be developed for each individual subsystem based on the SPECS specification and architecture [10], [11]. The specification of software security requirements for each subsystem should be obtained from:

1) security requirements that are specified for SMF;

2) requirements that arise from the SPECS architecture;

3) any requirements for functional security planning. The information provided should be freely available to software developers.

The specification of software security requirements should be detailed to ensure that the existing design and implementation stages of the SPECS are completed in order to achieve the necessary security completeness. The requirements specification should also allow verification. Software developers should familiarise themselves with the information contained in the specification in order to ensure that the requirements are properly defined, namely, in accordance with the IEC 62061 standard, the following should be taken into account:

– security-related management functions;
– system configuration or architecture;
– performance and response time;
– equipment and operator interfaces;
– all corresponding operating modes of the equipment;
– diagnostic tests of external devices (for example, sensors and actuators).

The requirements that have been set for software security should be expressed and structured in such a way that they:

– are clear, suitable for verification, testing, support and execution, and proportional to the level of security completeness;
– are suitable so that their source could be determined in the specification of the safety requirements of the SPECS;
– do not contain information or descriptions that can be interpreted ambiguously.

The software security requirements specification should reflect the required characteristics of each subsystem, as well as provide information that would allow selecting the hardware that meets existing security requirements [12], [13]. The following requirements must also be defined for SMF programming:

– logic (functionality) of all functional blocks must be executed by each subsystem;
– input and output interfaces are assigned for each function block;
– format and ranges of input and output data values and their relationship to function blocks;

– availability of relevant data describing any limitations of each block, such as maximum response time, limit values for validation;
– a number of functions related to achieving or maintaining a safe state of the machine; detection, notification and error handling; periodic testing in offline and non-autonomous SMF modes;
– prevention of unauthorised changes to the SPECS;
– interfaces of non-security functions (the interfaces include programming tools for autonomous and non-autonomous modes);
– performance and response time.

The functional security plan should define a strategy for developing, integrating, verifying, and verifying software compliance. In this context, the purpose of this study is to identify key requirements for compliance with modern security requirements of SPECS software, both built-in and applied [14].

At the same time, the modification of transport aircraft in the future is of great importance in terms of reducing operating costs, aimed at maintaining the aircraft in working condition for as long as possible [5].

Taking into account the significant number of transport aircraft that are currently operated on domestic and international airlines, this problem is very relevant throughout the state and requires detailed consideration.

A qualitative assessment of the economic efficiency of aircraft structure modifications is necessary for a comprehensive solution of the issues of managing the fleet of aircraft in the country and optimizing the state budget funds allocated annually for the development of the aviation industry in the country.

2. METHODS

When implementing SPECS, compliance of its software with modern security requirements can be ensured only if the requirements of such interrelated basic security standards as IEC 61508-1, IEC 61508-2 and IEC 61508-3 are met. Figure 1 presents a simplified algorithm for step-by-step implementation of SPECS, taking into account the requirements of these standards.

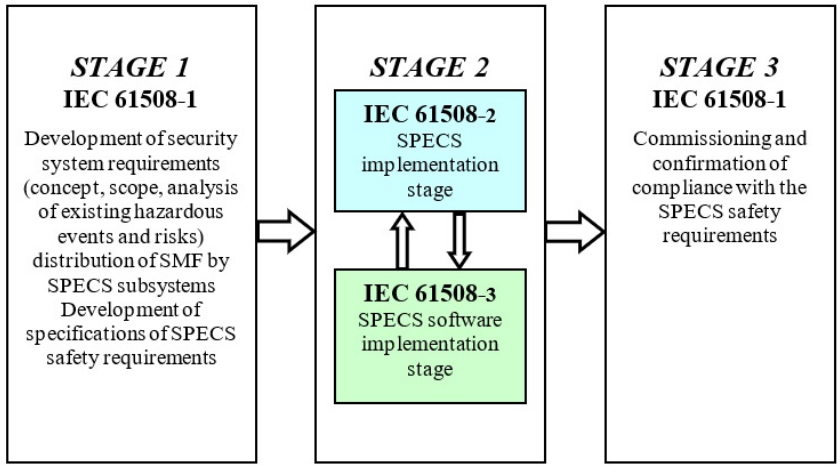

Fig. 1 – Step-by-step SPECS implementation algorithm

At the stages of development and implementation of SPECS software, both embedded and applied, it is necessary to make provision for the application of two main basic security standards in this area, IEC 61508-2 and IEC 61508-3. Their interrelation and areas of application are presented in Figure 2. The software embedded and included in the SPECS subsystems must meet the requirements of IEC 61508-2, IEC 61508-3, as well as the required security completeness level (SCL) under the IEC 62061 standard [5], [6], [15].

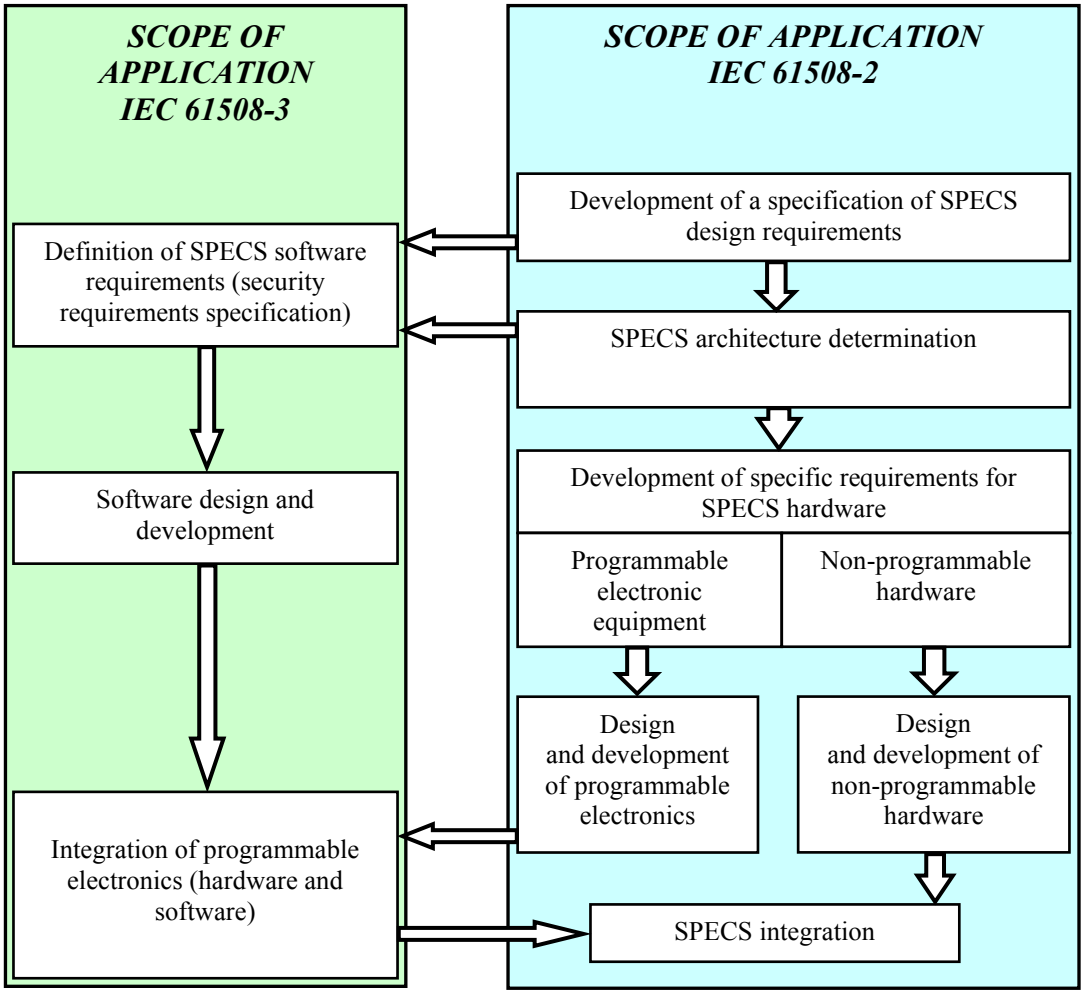

Fig. 2 – Interrelation and scope of IEC 61508-2 and IEC 61508-3 standards

In parameterization-based software, security-related parameters should be considered as aspects related to the security of the SPECS design, which are described in the software security requirements specification. Parameterization must be performed using a special tool of the SPECS supplier or the corresponding subsystem. This tool must have its individual identification (title, version, etc.) [16].

The completeness of all data used for parameterization should be maintained by means of appropriate management measures:

– range of acceptable input data;
– corrupted data before transmission;

– consequences of errors in the process of transmitting parameters;
– consequences of incomplete transmission of parameters;
– consequences of failures and crashes of technical means and software in the tools used for parameterisation.

The parameterisation tool must meet all relevant subsystem requirements in accordance with IEC 62061 standard to ensure correct parameterisation; a special procedure was applied to set safety parameters. The following confirmation must be included in this procedure:
– input parameters for the SPECS by retransmitting the changed parameters to the parameterisation tool, or using other means that confirm the completeness of the parameters;
– the result (for example, automatic verification with a parameterisation tool).

To avoid systematic failures of diversity in functions, one should use software modules designed for encoding or decoding during reception or transmission, as well as software modules designed to visualise security settings for users.

The parameterization-based software documentation should include all the data used (for example, initially defined parameter sets), as well as information necessary to identify parameters related to the SPECS, including information about the person(s) who performed the parameterization, along with other available information (for example, the date of parameterization). Along with this, the following verification actions are applied for parameterization-based software:
– verification of the correct installation for each security-related parameter (minimum, maximum, and representative values);
– verification that security-related parameters are checked for validity (for example, by detecting invalid values, etc.);
– verification that unauthorized changes to security-related parameters are not possible;
– verification that the data (signals) intended for parameterization are created and processed in such a way that possible failures cannot lead to loss of SMF.

This is especially important when parameterizing a device that is not specifically designed for this purpose (for example, a personal computer or similar device).

3. RESULTS AND DISCUSSIONS

The results of actions that were carried out in the process of its development and implementation must be checked at the corresponding stages. According to the required SCL for SMF, the selected implementation method and application language should have characteristics corresponding to this application that simplify the following:

1) abstractions, division into modules and other characteristics that control the complexity level, where possible, the software should be based on logical functions that are well-established and can connect functions intended for user libraries and have clearly defined rules for linking logical functions;

2) expression of the following:

– functions performed, ideally as a logical description or as algorithmic functions;
– data exchange between module components;
– requirements related to the sequence and time of execution;
– synchronisation restrictions;
– data structures of their properties, including types and validity of value ranges;

3) understanding of both the functionality of the application and the limitations of SPECS technology by developers and other persons associated with the project;

4) verification and confirmation of compliance, including structural testing of the application software, functional testing of the integrated application programme and testing of interfaces for interaction with the SPECS, and its specific configuration of technical means;

5) safe modification.

Testing should be the main verification method for application software [17]. Testing planning should cover a number of aspects:

– policy for verification of integration of software, hardware and technical means;
– test examples and test results;
– types of tests to be performed;
– test equipment, including tools, software support, and configuration descriptions;
– test criteria by which the completion of the tests will be assessed;
– territorial location (e.g., plant or consumer site);
– dependence on external functionality;
– number of tests required;
– completeness of related functions or requirements.

If the application software is to implement management functions, both related and non-security, then it should generally be considered as security-related, unless the project demonstrates sufficient independence between these functions [18]. The project should also include checking the security completeness and the validity of verification at the application level (for example, checks in communication channels, checks of limit values at sensor inputs, limit values of data parameters).

Application software development involves self-checking the control flow and data flow, if the specified functions are not included in the firmware. In case of a nonconformity, appropriate measures must be taken to achieve or maintain a safe state. If the project involves partial use of previously developed software, its ability to meet the security requirements of the software must be justified. This ability should be based on satisfactory performance data in similar applications for which similar functionality has been demonstrated. One should also evaluate the limitations that are associated with the software environment (for example, dependence on the operating system and compiler). Any changes or modifications to the application software shall be subject to an analysis that identifies all relevant software modules and identifies the steps required for re-verification to confirm that the software still meets security requirements. Software configuration management has to perform the following:

1) ensure that all necessary operations are performed and demonstrate that the necessary completeness of software security has been achieved;

2) maintain neat and uniquely identified all documents that relate to configuration objects and that are necessary to maintain the completeness of SPECS security. In addition, configuration objects must include at least the following:

– analysis of security and its requirements;
– specification of software and project documentation;
– software modules in source codes;
– test plans and results;
– already existing software modules and packages that should be included in the SPECS;
– all tools and development environments that are used to create, test, or perform any actions related to the software application;

3) apply change control procedures for:

– prevention of unauthorised changes;
– requests to change documents;

– analysis of the impact of the changes proposed, and in order to accept or reject the request(s);

– preparation of a document describing in detail and allowing all adopted changes;

– documentation of software configuration at appropriate points in software development;

4) document the following information to allow further audit: release status, justification and approval of all modifications, as well as detailed information about the modification;

5) officially register the application software release (master copies of the software and all related documentation must be kept to ensure maintenance and modification throughout the entire service life of the released software).

The basic requirements for software architecture are the software architecture that defines the main elements and subsystems of SPECS and Application Software, their interrelation and ways to achieve the required characteristics [19]. Examples of application software modules include application functions that are copied across the entire machine, I/O, overwriting and disabling components, checking data and value ranges, etc. The software architecture also depends on the underlying subsystem architecture provided by the vendor [20]. Software architecture design should be based on the SPECS security specification within the limitations of the SPECS system architecture and subsystem.

The software architecture project should do the following:

1) provide a complete description of the internal structure and functioning of the SPECS and its components;

2) include specifications of all identified software components, as well as a description of the relationships and interactions between these components (software and hardware);

3) include the internal design and architecture of all identified components;

4) identify software modules that are included in the PBESU, but are not used in any safety-related operating mode.

It is also important that the documentation of the SPECS architecture is up-to-date and complete. To meet the specification, it is necessary to describe and justify the set of methods and measures that are necessary for developing application software. These methods and measures should be aimed at ensuring predictable behaviour of the SPECS and should comply with the restrictions set out in the SPECS documentation. It is necessary to describe and justify the measures that are planned to be used to preserve the completeness of all data. They can be input and output data, switching data, interface operational data, maintenance data, and database content.

There are also certain requirements for support tools, user guides, and programming languages. A suitable set of tools should be selected, including configuration management, modelling, and testing [21]. It is necessary to factor in the ability of tools (not necessarily those that were used in the initial preparation of the system) to perform the necessary tasks throughout the entire life of the SPECS. This ability needs to be explained and documented.

The choice of development tool depends on the nature of the software development activity, firmware, and software architecture. One may also need to check and verify compliance with tools such as code analyzers, as well as modelling tools. Application specifications may include any performance limitations.

The selected application language must:

– have a translator/compiler that needs to be evaluated for suitability;

– be fully and unambiguously defined, or bounded for a subset of unambiguously defined elements;

– meet the application characteristics;

– have properties that simplify the detection of programming errors;

– maintain characteristics that correspond to the design method.

The architecture project description should document the shortcomings of the software language, as well as explain the suitability of the language for a specific purpose, including additional measures that are necessary to eliminate the identified shortcomings of the language.

Procedures in an application language should define acceptable configuration procedures, disable dangerous software features (such as undefined language parameters, unstructured templates, etc.), define controls that can be used to detect configuration errors, and define application documentation procedures. Documentation related to the application language must contain at least the following information: legal entity (for example, company, author, etc.); description; monitoring the functional requirements of the application; monitoring standard library functions; input and output data; history of configuration changes. Before starting a detailed application software design, certain information should be available:

– specification of software security requirements;
– software architecture project description;
– software security compliance plan.

The project description of the software architecture should include: definitions of application logic and functions that implement fault tolerance, a list of input and output data, software modules and support tools that have a common use, and application software configuration procedures to provide application functionality for certain input and output data.

When developing application software, one should use a structured approach. It is used to ensure modular functional applications; data that controls input and output; ability to check functionality (including fault tolerance) and internal structure; and secure modification parameters. Further improvements to the design of each major component or subsystem in the application description of the application software architecture should take into account: functions that are reused in the project; display of input and output information of application software modules; implementation of application functions using general software functions and display of input and output [22].

One needs to define the design of each software module and check the structure that should be performed for each application software module. One should specify the appropriate SPECS software integration tools to ensure that the application meets the required application software security requirements. Integration of security-related SPECS software includes tests defined at the design and development stage to ensure compatibility of the software with the hardware and embedded software platform, provided that functional and security requirements are met. The following should be subject to consideration:

– division of software into controlled integrated subsets;
– control examples;
– types of checks performed;
– testing conditions, tools used, configuration and programmes;
– conditions under which verification is considered completed;
– procedures to be performed if the check gave a negative result.

When implementing output tests of application programmes, the application software must:

– be readable, understandable, and verifiable;
– meet all design principles;
– meet all the requirements that were defined upon security planning.

The application software must be tested to ensure compliance with the specified project, coding rules, and existing security planning requirements. Application software considerations

include methods such as software validation or full control, code analysis, or formal validation. These methods should be used together with testing or modelling to ensure that the application software meets the specifications associated with it.

When testing software modules, one needs to meet several requirements. The process of verifying the compliance of application software with all the requirements contained in the test specification refers to verification processes [23]. The combination of initial testing and structural testing of software modules ensures that the software module meets the requirements of its specification, i.e., the module is tested. Each I/O configuration must be checked during analysis, testing, or simulation to make sure that the input and output data are compared with the correct application logic. In the process of analysis, modelling, and testing, it is necessary to test each software module to determine whether it performs the intended functions, and whether it performs functions that were unintended.

Tests must correspond to a particular module and must ensure verification of each branch of all application software modules; verification for boundary data; correct execution of the sequence of actions, taking into account the relevant synchronization requirements. The results of testing software modules must be documented. If the software has been pre-evaluated or there is information regarding a considerable positive experience of its operation, the number of tests can be reduced. When testing application software integration, one must consider the following requirements. The process of verifying the correct software integration refers to verification processes. The application software must be checked to ensure that all application software modules and components/subsystems interact correctly with each other and with the built-in software in order to perform the intended functions, and to exclude the performance of functions that could threaten any security function [24].

The results of testing application software integration should be documented and clearly defined: test results; whether the goals were met and whether the verification criteria were met. If testing fails, the reasons for the failure and corrective actions should be specified in the documentation. When integrating the application software, any changes or modifications to the software must be analysed for their security impact.

4. CONCLUSIONS

Based on the results obtained in the course of this study, it is possible to draw conclusions about the economic feasibility of modifying the design of modern aircraft transport at any stage of its operation. Measures to modify modern air transport can qualitatively change the situation in transport aviation, since the use of the latest technological solutions in the design of modern aircraft significantly increases its service life and reduces operating costs. There is a general economic effect from the modernisation of transport aircraft, expressed in a combination of a number of factors, both financial and technological.

Notably, the economic feasibility of modifications to the design of modern air transport is impossible without performing special calculations that take into account all possible components of this process. As the calculated data show, high indicators of economic efficiency of the considered process can be achieved only if the high-quality standards of work on the modernization of transport aircraft are maintained and timely deliveries of components to the place of their implementation are ensured. Otherwise, it is not necessary to consider the economic feasibility of all the operations performed as a whole. In addition, the economic feasibility of design modifications involves mandatory accounting for the costs of retraining maintenance personnel, who are forced to face the need to study the latest technological solutions used in the modernization of all types of air transport, as well as the practical

application of new knowledge. The assimilation of new technologies is a long process and does not always proceed smoothly — at its initial stages, temporary problems associated with insufficient qualifications of service personnel may occur, which require timely and high-quality resolution.

In general, the results obtained allow for the conclusion that the implementation of modifications to the design of transport aircraft should be considered economically feasible. With the proper organization of this process, in most cases, there is a tangible economic effect from its implementation. An increase in the operating life of aircraft without carrying out repairs and in conditions of a significant reduction in operating costs in combination provides savings in the funds allocated annually by the state for the arrangement of aviation. This, in turn, has a positive effect on the development trends of the aviation industry of any country, regardless of its current state.

REFERENCES

[1] * * * EN ISO 12100-1/2 “Safety of machinery General principles for design and risk evaluation. Basic concepts”, 2011, Available at: https://www.iso.org/obp/ui/#iso:std:iso:12100:ed-1:v1:en.

[2] * * * Directive 2006/42/EC of the European Parliament and of the Council, 2006, Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32006L0042.

[3] * * * DSTU EN 954-1: 2003 “Safety of machines. Security elements of control systems. Part 1. General principles of design”, 2003, Available at: https://budstandart.ua/normativ-document.html?id_doc=54828&minregion=852.

[4] * * * DSTU EN ISO 13849-1: 2016 “Safety of machines. Security system details. Part 1. General principles of design”, 2016. Available at: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=65214.

[5] * * * IEC 62061 “Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems”, 2005, Available at: https://webstore.iec.ch/publication/6426.

[6] * * * IEC 61508 (all parts) “Functional safety electrical/electronic/programmable electronic safety-related systems”, 2010, Available at: https://webstore.iec.ch/publication/5515.

[7] I. K. Sagynganova and V. B. Markin, The organizations of the tasks implementation in the distributed automatic control systems of heat supply stations, *News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences,* vol. **1**, no. 433, pp. 63-67, 2019.

[8] M. Dovzhik, A. Kalnaguz and Yu. Sirenko, Curvilinear movement of a four-wheel machine using a satellite navigation system, *Scientific Horizons,* vol. **7**, no. 92, pp. 126-135, 2020.

[9] V. I. Roman and L. O. Banduryna, Iionization and autoionization of rubidium atoms by electron impact, *Scientific Herald of Uzhhorod University. Series “Physics”,* vol. **43**, pp. 74-81, 2018.

[10] V. P. Babak and S. I. Kovtun, Calibration thermoelectric heat flux sensor in the diagnostic system of thermal state of electric machines, *Technical Electrodynamics,* vol. **2019**, no. 1, pp. 89-92, 2019.

[11] K. Imanbayev, B. Sinchev, S. Sibanbayeva, A. Mukhanova, A. Nurgulzhanova, N. Zaurbekov, N. Zaurbekova, N. V. Korolyova and L. Baibolova, Analysis and mathematical modeling of big data processing, *Peer-to-Peer Networking and Applications,* vol. **1**, pp. 1-10, 2020.

[12] V. Babak, V. Eremenko and A. Zaporozhets, Research of diagnostic parameters of composite materials using Johnson distribution, *International Journal of Computing,* vol. **18**, no. 4, pp. 483-494, 2019.

[13] R. Liang, H. Zhi and M. M. Kamruzzaman, Methods of moving target detection and behavior recognition in intelligent vision monitoring, *Acta Microscopica,* vol. **28**, no. 4, pp. 750-759, 2019.

[14] I. K. Sagynganova, A. I. Kvasov and A. A. Kalinin, Comprehensive methods to obtain and process information flows in centralized heat supply systems, *IOP Conference Series: Materials Science and Engineering,* vol. **972**, no. 1, 012074, 2020.

[15] S. F. Kashtanov, Compliance of the built-in software of electronic control systems with modern safety requirements, in *Proceedings of the Twenty-first All-Ukrainian scientific-methodical conference “Problems of labor protection, industrial and civil safety”*, pp. 138-142, 2019.

[16] S. Huda, J. Abawajy, M. Abdollahian, R. Islam and J. Yearwood. A fast malware feature selection approach using a hybrid of multilinear and stepwise binary logistic regression, *Concurrency and Computation: Practice and Experience*, vol. **29**(23), 1-18, 2017.

[17] T. Meade S. Zhang and Y. Jin, IP protection through gate-level netlist security enhancement, *Integration*, no. 58, pp. 563-570, 2017.

[18] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama and S. Achiche, Design, modelling, simulation and integration of cyber physical systems: Methods and applications, *Computers in Industry*, no. **82**, pp. 273-289, 2016.
[19] S. F. Kashtanov, Yu. O. Polukarov and L. O. Mityuk, Modern safety requirements for the design of electrical and electronic control systems, *Bulletin of Kremenchuk National University named after Mykhailo Ostrogradsky*, no. **119**, pp. 161-166, 2019.
[20] Z. DeSmit, A. Elhabashya, L. Wells and J. Camelio, An approach to cyber-physical vulnerability assessment for intelligent manufacturing systems, *Journal of Manufacturing Systems*, no. **43**, pp. 339-351, 2017.
[21] Á. M. Guerrero-Higueras, N. DeCastro-García and V. Matellán, Detection of Cyber-attacks to indoor real time localization systems for autonomous robots, *Robotics and Autonomous Systems*, no. **99**, pp. 75-83, 2018.
[22] S. Huda, J. Yearwood, M. M. Hassan and A. Almogren, Securing the operations in SCADA-IoT platform based industrial control system using ensemble of deep belief networks, *Applied Soft Computing*, no. **71**, pp. 66-77, 2018.
[23] Y. LeCun, Y. Bengio and G. Hinton, Deep learning, *Nature*, no. **521**, pp. 436-444, 2015.
[24] K. Sundaramoorthy, V. Thomas, T. O'Donnell and S. Ashok, Virtual synchronous machine-controlled grid-connected power electronic converter as a ROCOF control device for power system applications, *Electrical Engineering*, no. **101**, pp. 983-993, 2019.