ISSN 1392 - 1207. MECHANIKA. 2012 Volume 18(2): 196-202

Security method of embedded software for mechatronic systems

A. Venčkauskas*, N. Jusas, L. Kižauskienė***, E. Kazanavičius****, V. Kazanavičius*******
**Kaunas University of Technology, Studentų 50, 51368 Kaunas, Lithuania, E-mail: algimantas.venckauskas@ktu.lt*
***Kaunas University of Technology, Studentų 50, 51368 Kaunas, Lithuania, E-mail: nerijus.jusas@stud.ktu.lt*
****Kaunas University of Technology, Studentų 50, 51368 Kaunas, Lithuania, E-mail: losta@ifko.ktu.lt*
*****Kaunas University of Technology, Studentų 50, 51368 Kaunas, Lithuania, E-mail: ekaza@ifko.ktu.lt*
******TEO LT, AB, Lvovo g. 25, 09320 Vilnius, Lithuania, E-mail: vkaza@ifko.ktu.lt*

crossref http://dx.doi.org/10.5755/j01.mech.18.2.1572

1. Introduction

Mechatronic systems are widespread in various areas of life – home, office, manufacturing, and transport. They are widely used in robots, digitally controlled machines, "smart machine tool" and so on. The typical view of mechatronics is as a combination of mechanical and electrical systems controlled by an embedded control system [1] (Fig. 1).

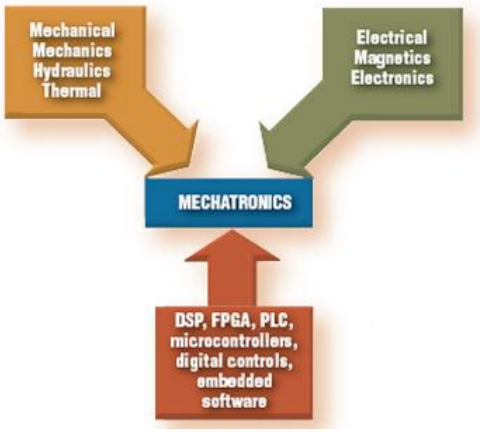

Fig. 1 Mechatronics is a synergy of mechanical and electrical systems controlled by an embedded system

Machining is a process that removes a lager of material from a workpiece in the form of chips to obtain the desired product shape, size, accuracy, and surface quality. Conventional machining operations, which include turning, milling, grinding, and drilling are among the most common activities in the manufacturing industry (US industries spend US $100 billion annually to machine metals). Experimental structure of smart machine tool is presented in Fig. 2.

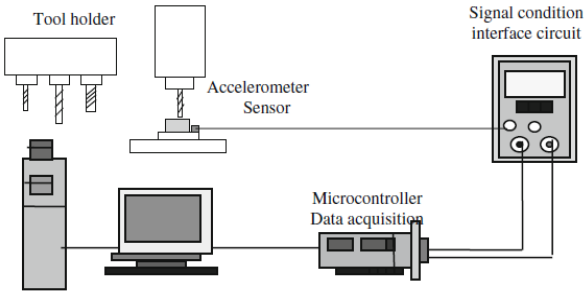

Fig. 2 Experimental structure of smart machine tool

The complex interaction between machines, tools, workpieces, fluids, measurement systems, material handling systems, humans and the environment in cutting operations requires the application of sensors or embedded systems to ensure efficient production identify the needs for maintenance, protect workers and the environment [2]. Standard approaches of process monitoring are the measurement or identification of the interaction between the process and machine structure.

In a "smart machine tool" the objective is to maintain an optimized cutting performance by using sensors and control systems with knowledge accumulation capability for use in future production. Vibrational behavior of the tool is of utmost importance since it significantly affects the workpiece [3]. For example, measurement of vibrations on the tool fixture is one of the indirect methods to evaluate the effects of the cutting force.

Vibration sensor signals are very sensitive to the change of workpiece dynamics, which reflects the change of cutting force due to the tool wear. During machining operation the sensors collect tool vibration signals in real-time, which are transmitted to the machine control system via feedback loop, which adjusts cutting parameters, if required, in order to reduce excessive unwanted vibrations in machine-tool-workpiece system thereby ensuring high machining quality and higher productivity. These cutting parameters may include feed rate, depth of cut, spindle speed, etc. As the sensors need to be installed near the cutting area inside the machining chamber, the wiring is an obstacle to the application of vibration measuring device in machining centers, in particular in milling machines, where the cutter and workpiece are always moving. Therefore, wireless data transmission is an attractive solution for vibration monitoring in machining operation.

Sensor systems must be able to be interfaced with open system architecture controllers for machines and systems must be designed to accommodate needs of so called "reconfigurable" systems. Activity in both of these areas is still predominately in the research stage with few industrial applications. Accordingly, one of the main challenges in future machining process monitoring systems is the development of algorithms and paradigms that are truly autonomous from machine tool operators with signal feature extraction and decision making performed without intervention of the operator, who should provide only very simple (the lesser, the better) input and information.

Integral parts of mechatronic systems, which often determine the system's functionality and vitality, are the embedded control systems – digital hardware and software subsystem. As an integral part, mechatronic systems

and embedded systems face significant challenges in information security; these systems usually have very limited resources and function in an unsafe environment. Embedded systems usually perform critical functions – control important real time objects, process important information, therefore its work can be sabotaged.

Security requirements of an embedded system's depend on specific areas of application [4]. The following requirements are related to the general requirements for information security: integrity, availability and confidentiality. However, the specificity of mechatronic systems, their mobility and work in real time, typically have certain limitations such as processing gap, energy gap, flexibility, tamper resistance, assurance gap and cost, largely due to limited resources, performance and security requirements.

An important component of embedded systems, which often determines the system's performance and vitality is software. Software security has two aspects: secure program and program protection. We will explore the protection aspect of the program security. The main program protection vulnerabilities are [5]: violation of intellectual property – illegal copying and distribution, improper use of licenses, and reverse engineering – disclosure of software code, theft of algorithms and falsification of software codes.

According to a study by the Business Software Alliance (BSA) [6], software creators lost 51.4 billion dollars and pirated software accounted for 43% of all software, observing approximately 2% annual growth trend of piracy.

No matter from what threats software is protected, for example copying or stealing algorithms, attackers attempt to crack the protection by several methods including reverse engineering, including disassembly and decompilation, debuggers, disassemblers, decompilers, emulators, simulators and spoofing attacks [7].

There are many software protection methods, which are divided into software-based and hardware-based.

Software-based protection mechanisms are installed into software or algorithms that are protected and can be added to software code - code and date obfuscation [8], anti-debugging method [9], code encryption technology, self-modifying code and self-extracting code [10].

Hardware-based methods can significantly increase the level of security, because it is external device in which the level of security is controlled by the software provider and not by the end-user [11, 12]. By using additional hardware (commonly Dongle or USB keys), part of the program code or data (encryption keys) required to run the program, can be stored. However, this protection mechanism is relatively expensive and is generally only used for those programs that are of great commercial value.

Intermediate software/hardware methods are also used – tethering a program to a computer or devices signatures (CPU, RAM, ROM, BIOS, OS and etc. serial numbers, model ID and so on) [13-15]. Firewalls are used for the protection of internet programs [16]. These methods are usually used for anti-piracy in personal computers.

In assessing the limitations of embedded systems [17], one of the most acceptable software protection methods is encryption of a code. However, one needs to take into account the key's management issues; external storage medium, network – transfer must be secure, using SSL protocol and the encryption key entered manually.

Software development is one of the most challenging tasks during the design of a mechatronic system. Mechatronic system software is related to and dependent on the other system components; mechanics, electronics, controllers, etc. Therefore, ranges of techniques are used for the development of mechatronic system software.

Model driven architecture is an approach to increase the quality of complex software systems based on creating high level system models that represent systems at different abstract levels and automatically generating system architectures from the models. In the papers [18, 19] is proposed a model-driven (model-based) approach to design the software part of a mechatronic system, which consists of two major parts; systematic modeling and correctness-preserving synthesis. In the paper [20] is presented an agent-based embedded control system design methodology for mechatronic systems. The paper [21] puts forward a component-based development method for increasingly complex embedded systems. Most methods used the UML (Unified Modeling Language) for the description of mechatronic systems.

Protection of programs is not directly related to mechatronic system functionality. In order for the developer to concentrate on the functionality, he should be free from issues related to program protection. Protection of programs must be automatically included in the system during the realization. For this it is necessary to describe the program protection requirements at a high level of mechatronic system design (UML).

Model-based approach is also widely used to create secure software. In the paper [22] are described processed data security and an access control requirement in the UML and OCL (Object Constraint Language), each vulnerability defined by its own stereotype. In the paper [23] is proposed an approach to the security model as a separate concern by augmenting UML with separate and new diagrams for role-based, discretionary and mandatory access controls; collectively, these diagrams provide visual access-control aspects. In the paper [24] is proposed security primitives (Authentication, data Integrity, data Confidentiality …) for UML; [25] defines User rights as UML and OCL context. The Secure UML meta-model [26] introduces the concepts of User, Role, and Permission to annotate UML diagrams with information pertaining to access control. In the paper [27] are described security criteria, such as confidentiality and integrity. He also defines in UMLSec a UML profile extension using stereotypes, tagged values and constraints.

As we can see, the UML is extended in various ways and is mainly used for creating secure software.

Our goal is to extend the model-driven embedded system development methodology measures to describe the requirements for the program protection to create a mechatronic system embedded software protection method. This method should implement a sufficient level of protection and not require additional hardware and security infrastructure.

In the following sections we describe the proposed security method of embedded software for mechatronic systems and investigate its characteristics and the possibilities of using for protection of embedded software.

2. Embedded software protection method

Protection method for mechatronic systems embedded software core is:

- protection requirements of the program modules are described in the UML diagram by using OCL constraints;
- installation procedure of mechatronic systems embedded software automatically integrates program protection;
- program data and code modules are stored separately;
- critical program modules are encrypted by symmetric algorithms independently of each other;
- encryption keys are not stored; they are generated from the system component's signature on demand before encryption or decryption;
- code modules are decrypted just before the execution (runtime decryption). After execution they are destroyed.

To describe the program module requirements for the protection, we extended the UML diagrams by special OCL constraints. These requirements, we describe in the UML class and components level, use these types of OCL constraints:

<< protectionRequirements >>
context *programModule : ProgramModule:*
self.ProtectionLayer = {1…3}
self.TimeRestrictions = real
self.SignaturesNumber = {1…7}
self.KDFfunction = {MD, SHA, SHA-2}
self.encryption = {DES, AES, Blowfish}

In the constraints there may be specified a necessary level of protection, time limitations, encryption key, the number of signatures and the generation function and the encryption algorithm. If the protections settings are not specified, then the default level of the program protection is applied.

A representation of program protection requirements in UML diagram format is shown in Fig. 3.

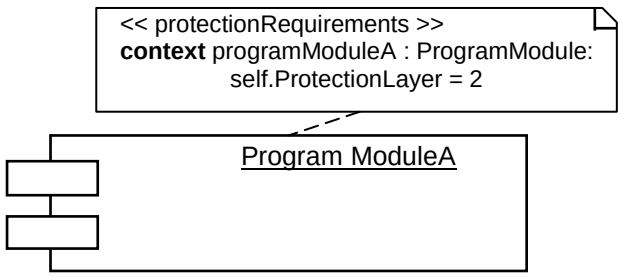

Fig. 3 Representation of program protection requirements in UML diagram

By installing embedded software of a mechatronic system, according to a description of the UML, a special install program automatically adds the security measures, created by protection templates.

Secret keys are generated in our proposed method [28]. Secret key generation process is shown in Fig. 4.

Protection key of software module is generated according to the protecting software headers and mechatronic system hardware and software components (controller, CPU, RAM, ROM, BIOS, OS, and etc.) signatures, using the fastest and simplest logical commands (XOR, OR).

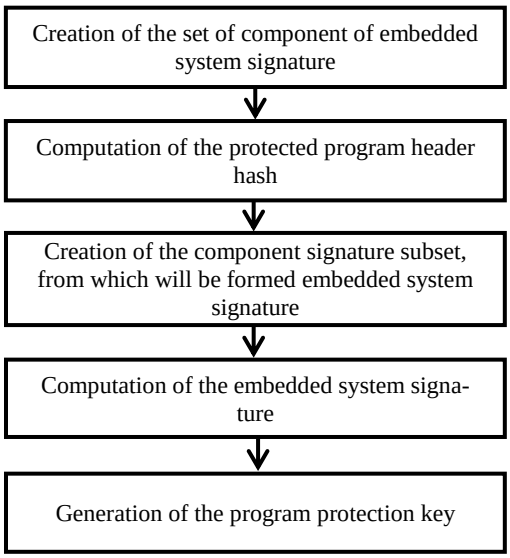

Fig. 4 Secret key generation process

The encryption key must be a fixed length and must have sufficient value of entropy. The strings of an embedded system signature are variable in length. Key Derivation Functions [29] and hash functions MD5, SHA, SHA-2 [30] are used to format fixed-length and high entropy secret keys from the variable-length strings.

The structure of the protected program is presented in Fig. 5. To increase effectiveness of the program, only critical code modules are encrypted and other modules – the program header, the data segments and noncritical modules are not encrypted.

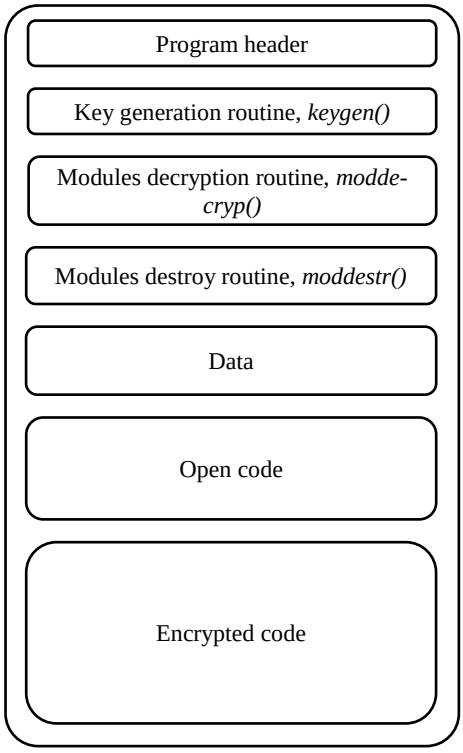

Fig. 5 The structure of the protected program

Encrypted code modules are decrypted in execution time automatically. Therefore, each module includes calls to key generation and decryption routines (Fig. 6).

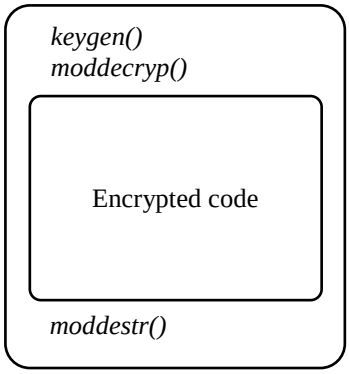

Fig. 6 The structure of the protected module

The program is protected (the required modules are encrypted) during installation in mechatronic systems by using a special software installer, whose functioning is shown schematically in Fig. 7.

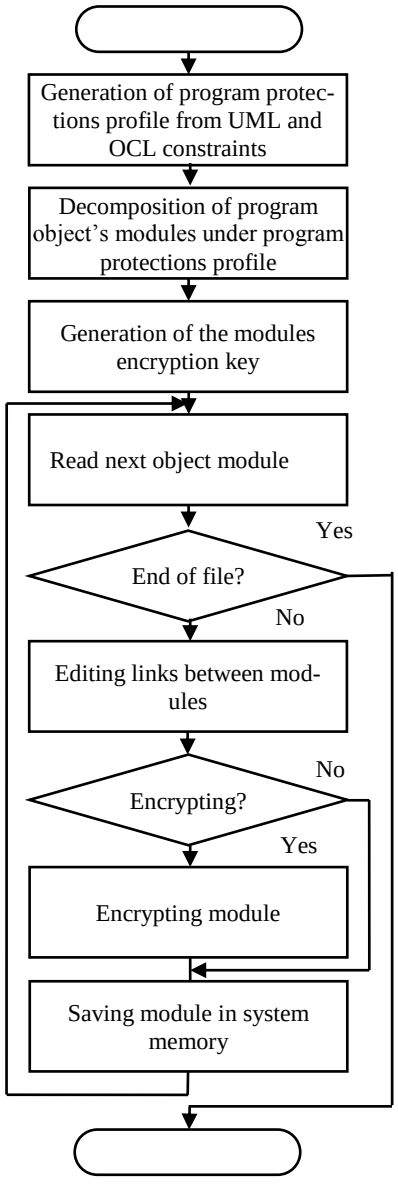

Fig. 7 The software installer operation scheme

The main steps of the installation process:

- generation of program protections profile from UML and OCL constraints;
- decomposition of program object's modules under program protections profile;
- generation of the modules encryption key. Editing links between modules, encrypting and saving modules in system memory.

The next section will investigate the created method of the program protection characteristics.

3. Evaluation of embedded software protection method

For evaluation of the proposed method, we created a prototype of mechatronic system software installer that realizes the described options. We investigated the secret encryption key entropy and its dependence on the signature creation and the hash function, and the formation time. We also estimated the impact of various encryption algorithms to operation speed of protection mechanisms; this is vital to mechatronic systems operating in real time.

The experiments were performed on the PDA (Personal Digital Assistant) of the model ASUS P750 (Pocket PC platform, Intel PXA270 520 MHz CPU, 256 MB RAM, Windows Mobile © 6 Professional CE OS 5.2). We simulated the software of a mechatronic system by programming discrete mathematical methods. The experiment's initial data – header of the program to be protected, mechatronic system hardware and software components signatures elements (Vendor ID, Type ID, Model ID and Serial Number), their lengths and numbers generated with programmable random strings and numbers generators. 20 sets of signatures (from 2 to 7 elements) were generated.

Secret encryption keys are generated from the embedded system signature using Key Derivation Function. These functions use hash functions, such as MD5, SHA, SHA-2 etc. Furthermore, we investigated the influence of the hash function algorithm for the value of entropy. Since the embedded system signature, which was formatted using sign 4 function, based on OR and XOR operations [28], has the best entropy, we investigated the key generated by this function. Fig. 8 displays the entropy of keys, which was formatted from 7 component signatures, using sign 4 function and MD5, SHA and SHA-2 hash functions.

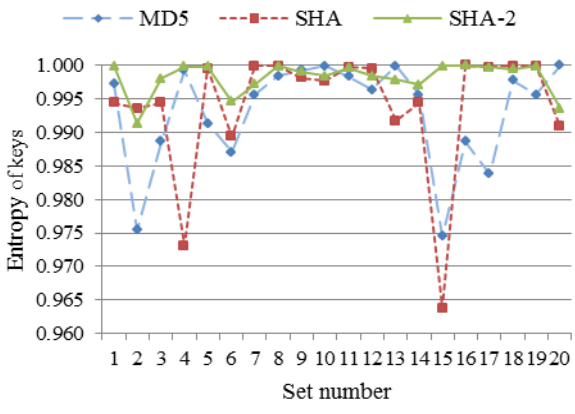

Fig. 8 Keys entropy depend on the hash functions

Entropy estimates – average, standard deviation

and prediction interval depending on the hash function are shown in Table 1.

Table 1

Secret keys entropy depend on the function

Function	Average	Standard deviation	Prediction interval	
			min	max
MD5	0.994	0.008	0.985	1.000
SHA	0.995	0.007	0.988	1.000
SHA-2	0.998	0.003	0.994	1.000

All hash functions generate high-entropy cryptographic keys, however the least standard deviation (0.003) and the lower limit of prediction interval (0.994) contain keys generated using function SHA-2.

The computing time (ms) of the keys, which was formatted from 7 component signatures, using sign 4 and MD5, SHA and SHA-2 hash functions is shown in Fig. 9.

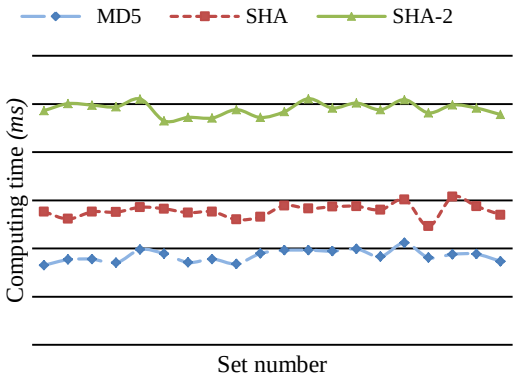

Fig. 9 Keys generation time (ms) dependence on the hash functions

Key computing time estimates – average, standard deviation and prediction interval, depending on the hash function are shown in Table 2.

Table 2

Keys computing time (ms) dependence on the function

Function	Average	Standard deviation	Prediction interval	
			min	max
MD5	23.515	0.802	22.713	24.317
SHA	28.209	0.791	27.418	29.000
SHA-2	38.805	0.867	37.938	39.672

As can be seen from Table 2, the best time characteristics were obtained by using the MD5 hash function, 65% faster than SHA-2. In the assessment of the generated key entropy (Table 1) and the generation time (Table 2), it is clear that for key generation it is better to use MD5, as the entropy is high enough, only 0.4% lower than the SHA-2, but with a much shorter generation time.

To investigate the impact of encryption algorithms to characteristics of program protection method, the simulated module solved the system of differential equations by using the Runge-Kutta method. The experiment was repeated 20 times and different algorithms were used to encrypt the module. Program execution times average and encryption module size (kB) are presented in Table 3.

As can be seen from Table 5, the best time characteristics were obtained by using the Blowfish, DES and IDEA algorithms. Blowfish are known to have better encryption (i.e. stronger against data attacks) than the other two. The Blowfish algorithm is the smallest size at 7.2 kB. It is therefore proposed to use the Blowfish algorithm to protect programs.

Table 3

Module execution time (ms) dependence on the encryption algorithm

	Unprotected	Encryption algorithm					
		DES	TR-DES	AES CBC	AES CFB	IDEA	Blow-fish
Average	26.5	37.9	80.0	48.1	49.3	38.6	37.6
Increase		11.4	53.5	21.6	22.8	12.1	11.1
Size kB		15	12.9	11.9	12.2	12.1	7.2

4. Conclusions

In this paper we have presented security method of embedded software for mechatronic systems. This method is based on encryption and decryption code of critical program modules during execution.

We proposed to describe protection requirements of the program modules in the UML diagram by using OCL constraints.

The proposed method effectively generates high entropy keys using the embedded system signature.

The Blowfish algorithm is the fastest and has better encryption: it is therefore proposed to use the Blowfish algorithm to protect programs.

References

1. **Lennon, L.; Mass, N.** 2008. Model-based design for mechatronics systems, Machine Design, Embedded Systems Industry Focus – Electronics World, 23-26.
2. **Bargelis, A.; Mankute, R.** 2010. Impact of manufacturing engineering efficiency to the industry advancement, Mechanika 4(84): 38-44.
3. **Ubartas, M.; Ostasevicius, V.; Samper, S.; Jurenas, V.; Dauksevicius, R.** 2011. Experimental investigation of vibrational drilling, Mechanika 17(4): 368-373. http://dx.doi.org/10.5755/j01.mech.17.4.563.
4. **Kocher, P.; Lee, R.; McGraw, G.; Raghunathan, A.** 2004. Security as a new dimension in embedded system design, In Proceedings of the 41st annual Design Automation Conference (DAC '04). ACM, New York, NY, USA, 753-760.
5. NIST. National Vulnerability Database Version 2.2, http://nvd.nist.gov/home.cfm.
6. BSA. 2010. Seventh Annual BSA and IDC Global Software Piracy Study, 18 p.
7. **Main, A.; van Oorschot, P.C.** 2003. Software protection and application security: Understanding the battleground, International Course on State of the Art and Evolution of Computer Security and Industrial Cryptography, Heverlee, Belgium, 19 p.
8. **Collberg, C.; Thomborson, C.; Low, D.** 1997. A taxonomy of obfuscating transformations, Technical Report 148, Department of Computer Sciences, the

University of Auckland, 36 p.
9. **Gagnon, M.N.; Taylor, S.; Ghosh, A.K.** 2007. Software protection through anti-debugging, IEEE Security and Privacy, 5(3): 82-84. http://dx.doi.org/10.1109/MSP.2007.71.
10. **Kanzaki, Y.; Monden, A.; Nakamura, M.; Matsumoto, K.** 2003. Exploiting self-modification mechanism for program protection, Proc. the 27th Annual International Computer Software and Applications Conference, Washington: IEEE Computer Society, 170-179.
11. **Jozwiak, I.J.; Liber, A.; Marczak, K.** 2007. A hardware-based software protection systems-analysis of security dongles with memory, Proc. the International Multi-Conference on Computing in the Global Information Technology (ICCGI'07), Washington: IEEE Computer Society, 28-38.
12. **MeiHong, L.; JiQiang, L.** 2010. USB key-based approach for software protection, International Conference on Industrial Mechatronics and Automation: 151-153.
13. **Mumtaz, S.; Iqbal, S.; Hameed, I.** 2005. Development of a methodology for piracy protection of software installations, 9th International Multitopic Conference, IEEE INMIC, 1-7.
14. **Liutkevičius, A.; Vrubliauskas, A.; Kazanavičius, E.** 2011. Assessment of dongle-based software copy protection combined with additional protection methods, Electronics and Electrical Engineering, 6(112): 111-116.
15. PC GUARD. Professional software protection and licensing system, http://www.sofpro.com.
16. **Kazanavičius, E.; Paškevičius, R.; Venčkauskas, A.; Kazanavičius, V.** 2012. Securing web application by embedded firewall, Electronics and Electrical Engineering, 3(119): 65-68.
17. **Babar, S.; Stango, A.; Prasad, N.; Sen, J.; Prasad, R.** 2011**.** Proposed embedded security framework for Internet of Things (IoT), Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference: 1-5.
18. **Barner, S.; Geisinger, M.; Buckl, C.; Knoll, A.** 2008. EasyLab: Model-Based development of software for mechatronic systems, Mechtronic and Embedded Systems and Applications, MESA 2008. IEEE/ASME International Conference, 540-545.
19. **Huang, Jinfeng; Voeten, J.; Groothuis, M.; Broenink, J.; Corporaal, H.** 2007. A model-driven design approach for mechatronic systems, Application of Concurrency to System Design, ACSD 2007, Seventh International Conference, 127-136.
20. **Kižauskienė, L.; Kazanavičius, E.; Gaidys, R.** 2011. Agent-based methodology for developing mechatronic systems software, Mechanika 17(5): 551-556. http://dx.doi.org/10.5755/j01.mech.17.5.735.
21. **Torngren, M.; DeJiu, Chen; Crnkovic, I.** 2005. Component-based vs. model-based development: a comparison in the context of vehicular embedded systems, Software Engineering and Advanced Applications, 31st EUROMICRO, 432- 440.
22. **Peralta, K.P.; Orozco, A.M.; Zorzom A.F.** 2008. Specifying security aspects in UML models, In ACM/IEEE 11th International Conference on Model Driven Engineering Languages and System, Toulouse, França, Proceedings of the Workshop on Modeling Security (MODSEC08 1: 1-10.
23. **Pavlich-Mariscal, J.; Michel, L.; Demurjian, S.** 2007. Enhancing UML to model custom security aspects, Proceedings of the 11th International Workshop on Aspect-Oriented Modeling (AOM@AOSD'07).
24. **Nakamura, Y.; Tatsubori, M.; Imamura, T.; Ono, K.** 2005. Model-driven security based on Web services security architecture, Services Computing, 2005 IEEE International Conference 1: 7-15.
25. **Alam, M.M.; Breu, R.; Breu, M.** 2004. Model driven security for Web services (MDS4WS), Multitopic Conference, Proceedings of INMIC 2004, 8th International 498-505.
26. **Basin, D.; Doser, J.; Lodderstedt, T.** 2006. Model driven security: from UML models to access control infrastructure, ACM Transactions on Software Engineering and Methodology (TOSEM) 15(1): 39-91. http://dx.doi.org/10.1145/1125808.1125810.
27. **Jürjens J.** 2002. Using UMLsec and goal trees for secure systems development, Proceedings of the 2002 ACM symposium on applied computing, 1026-1030.
28. **Venčkauskas, A.; Jusas, N.; Mikuckienė, I.; Butleris, R.** 2012. Secret encryption key generation using signature of embedded systems, Information Technology and Control. 41(xx) (Submitted).
29. International Organization for Standardization. 2004. ISO/IEC FCD 18033–2, IT Security techniques — Encryption Algorithms — Part 2: Asymmetric Ciphers.
30. **Henke, Ch.; Schmoll, C.; Zseby, T.** 2008. Empirical evaluation of hash functions for multipoint measurements, SIGCOMM Comput. Commun. Rev. 38, 3: 39-50. http://dx.doi.org/10.1145/1384609.1384614.

A. Venčkauskas, N. Jusas, L. Kižauskienė,
E. Kazanavičius, V. Kazanavičius

MECHATRONINIŲ SISTEMŲ ĮTERPTOSIOS PROGRAMINĖS ĮRANGOS SAUGOS METODAS

R e z i u m ė

Straipsnyje pateiktas mechatroninių sistemų įterptosios programinės įrangos saugos modelis, paremtas kritinių programos modulių šifravimu ir dešifravimu kodo vykdymo metu. Slaptieji šifravimo raktai nesaugomi, o generuojami pagal mechatroninės sistemos komponentų signatūras. Darbe eksperimentiškai ištirta šifravimo raktų entropija, įvairių simetrinių kriptografinių algoritmų taikymo galimybės ir apsaugos priemonių įtaka įterptosios programinės įrangos charakteristikoms – greitaveikai ir papildomos atminties sąnaudoms.

A. Venčkauskas, N. Jusas, L. Kižauskienė,
E. Kazanavičius, V. Kazanavičius

SECURITY METHOD OF EMBEDDED SOFTWARE FOR MECHATRONIC SYSTEMS

S u m m a r y

This paper proposes embedded software of mechatronic system protection method based on encryption and decryption code of critical program modules during runtime. Secret keys are not stored, but generated by the signature of mechatronic system components. This paper experimentally researches the application of symmetric cryptographic algorithms and the influence of security mechanisms on characteristics (value entropy of secret key, operating speed, and amount of memory) of embedded software.

Keywords: mechatronic system, embedded software, security, secret key generation.

Received May 05, 2011
Accepted April 12, 2012