AUTOMATYKA/ AUTOMATICS • 2012 • Vol. 16 • No. 2

http://dx.doi.org/10.7494/automat.2012.16.2.155

Dariusz Rzońca*, Andrzej Stec*, Bartosz Trybus*

Secure Web Access to Mini Distributed Control System

1. Introduction

Network data transfer methods for control systems have been advancing especially during the last decade. Classic solutions are gradually replaced by distributed architectures called Networked Control Systems [6, 9, 29]. However, the new techniques have introduced additional risks and threats for the safety of transferred information. Data integrity and confidentiality expected from industrial DCS and SCADA systems with network facilities [2, 7] are not fully guaranteed by common communication protocols [16]. As shown in [14] typical protection solutions like firewall or anti-virus software do not provide sufficiently high safety level. Therefore, the protection of critical applications such as power systems and distribution networks is now an important research area, leading to new security procedures, as e.g. intruder detection mechanisms [4].

Large control systems usually provide sufficient computational power to implement security algorithms adapted from generic, PC-oriented techniques. Small systems, however, do not have adequate hardware and software resources, so simpler solutions must be developed instead, but still providing a reasonable level of security.

This paper presents an implementation of secure Internet access to a lab-scale small distributed control system. The system consists of a set of PLC modules and a microserver with 8-bit microcontroller. Two other industrial small systems programmed with the same engineering tool are also mentioned. Remote client, via web browser, can monitor the process variables of the lab system, display trends from short-term archive, and program or configure PLC modules. Challenge-response protocol [26] authenticates the user and initializes the connection. Symmetric cryptography, less demanding than asymmetric, provides the encryption of browser-server messages. Communication latency is estimated by simulating a Time Petri Net model of the system.

* Department of Computer and Control Engineering, Rzeszów University of Technology, Rzeszów, Poland; e-mail: drzonca; astec; btrybus@kia.prz.edu.pl

2. Mini Distributed Control System

Small distributed control systems, often called mini-DCS and here abbreviated to short mDCS, normally consist of remote I/O and control modules and of master module. PLC modules with analog and binary I/Os are slaves in the lab mDCS system of Figure 1. The modules communicate with microserver (master) by means of Modbus protocol. Each PLC consists of an Atmel development board with ARM7 32/16-bit microcontroller, and some extra I/O and RS-485 hardware. The mDCS microserver involves an evaluation board with Atmel AVR 8-bit microcontroller and Ethernet-interface chip. PLC applications are prepared in IEC 61131-3 compliant CPDev (*Control Program Developer*) engineering environment [10, 20]. The program is executed by a CPDev runtime virtual machine implemented in a FreeRTOS operating system [3]. Program variables are collected by mDCS server and available to the user by a web page.

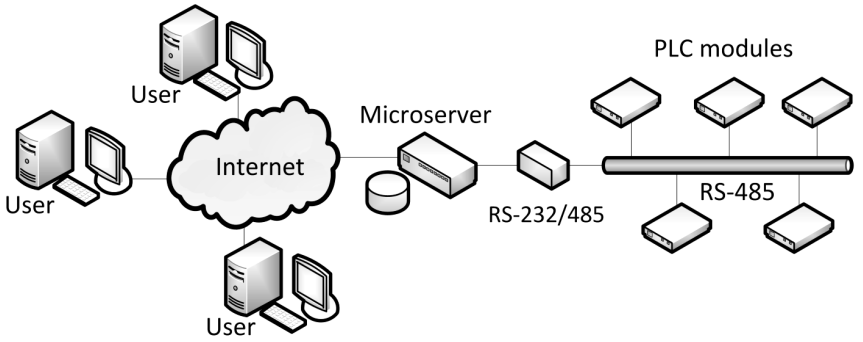

Fig. 1. Architecture of lab mini-DCS

The CPDev environment, developed especially for small DCS systems, integrates tools for programming, simulation, communication configuration, and on-line testing of control applications written in ST, IL and FBD languages [17, 18]. The main window with a program written in ST (center) and FBD (right side) is shown in Figure 2. The left side presents a project tree. The program turns MOTOR on if START is pressed, provided that STOP and ALARM are not set. MOTOR continues running after releasing START. PUMP is turned on and off by the timers TON, TOF, 5 seconds after the MOTOR.

CPDev compiles control programs into intermediate, universal code executed at the controller side by the runtime interpreter, i.e. virtual machine. The machine is written in C, so it may run on different hardware platforms, from 8-bit microcontrollers up to 32/64-bit general purpose processors. A mini-DCS with SMC controller manufactured by LUMEL, Zielona Góra, PL, is one of two industrial applications of CPDev [25]. Besides the SMC controller (8-bit AVR), the system includes remote I/Os, intelligent transmitters, actuators, displays, etc., and PC or HMI panel as operator interface. “Off-the shelf” applications involve heating substation controls (“hot spot”) and production line monitoring systems.

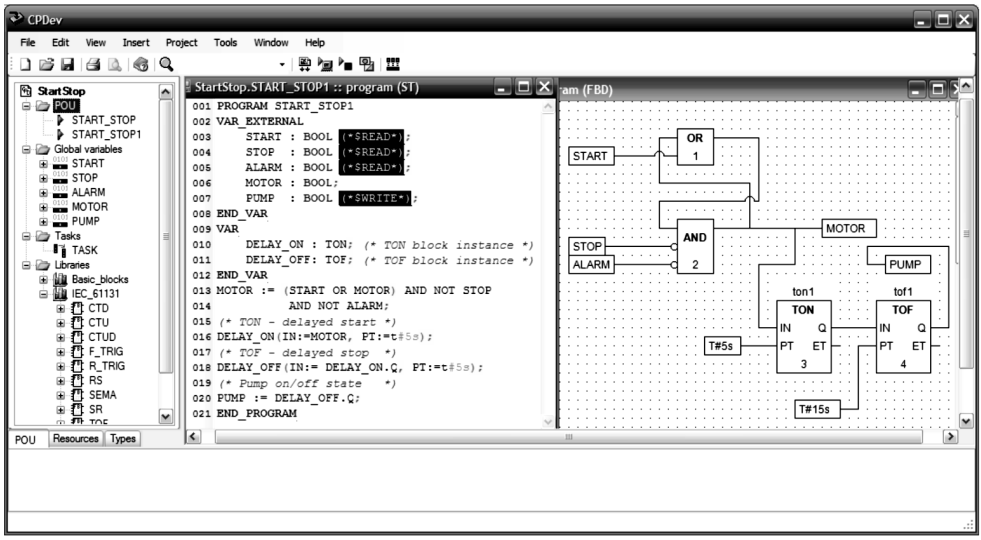

Fig. 2. Main window of CPDev programming environment

A Mini-Guard Ship Control and a Positioning System from Praxis A.T., Leiden, NL, is another industrial application of CPDev [13]. Mini-Guard controllers involve NXP ARM7 microcontrollers connected by Ethernet to marine-PC. A DAMEN-class towboat was the first ship with a Mini-Guard on board. The CPDev virtual machine can also be a core of soft-PLC, i.e. PC equipped with I/O boards, for lab, diagnostic and teaching applications [21]. The PC may run standard Windows, Windows Embedded, CE .NET or QNX6. A fast FPGA-based controller has been developed recently [8].

Secure web access described here may be adopted in the future to one of these systems.

3. Security Aspects

Small control systems with limited resources require non-standard solutions to keep data transmission secure. Particularly, a need for prompt responses prevents the use of complex cryptographic primitives [5], such as asymmetric key processing. Despite limitations, a reasonable level of security can be achieved with symmetric cryptography.

Industrial systems grant access to authorized persons only. Logins (user names) and passwords are entered by a dedicated program or administrator service via secure channel, e.g. USB. In the mDCS system of Figure 1 the passwords are stored in a microserver database in hashed form, together with random salt. No possibility exists to read it back. Common attacks involve the use of precomputed lookup tables or rainbow table for reversing the cryptographic hash functions. Employing the random salt and running the hash function multiple times (key stretching) make such attacks ineffective [28]. To minimize

the probability of successful brute force or a dictionary attack, the password must be long enough (at least eight characters) and may be combination of small and capital letters, digits, and special characters.

Authentication of web client who wants to get access to the mDCS server involves login and password. To avoid transmitting the password as plain text, the challenge-response protocol is applied [23]. The authentication scheme is shown in Figure 3. At first, the user enters the *login* (transmitted to the server in *hello message* as plain text) and password (not transmitted). The server replies with the *authentication request* which contains a *random number* and a *salt* corresponding to the login. The password is hashed by the client using the salt to make so-called secret key. Another secret key is read by the server from the database. Both sides independently hash their secret key with the random number. One part of the result calculated by the client is sent to the server as an *authentication answer* and then compared (*answer verification*). If the values are equal, the authentication is successful. The remaining part of the result becomes a *session key*, generated on both sides, and used later to encrypt the transmission. AES-128 [1] is applied as a cipher and SHA-256 [24] as a hash function. CTR (Counter) mode [30] used in mDCS for cipher operation turns the block cipher (AES) into a cipher stream. The current counter value is encrypted by AES using the session key to produce the stream. The AES output is XOR-ed with the plain text message to create the ciphered message. In this way, identical plain text messages are encrypted into different cipher text, unlike in the simple ECB (Electronic codebook) mode. Since some messages will be sent repeatedly (e.g. some group of the variables can be constantly monitored) it is essential to ensure that their cryptograms are different.

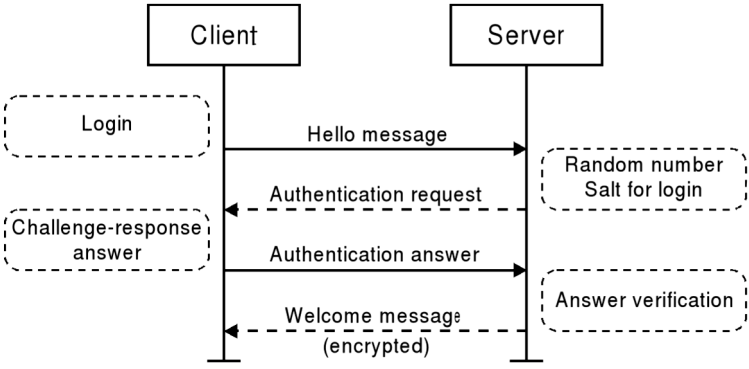

Fig. 3. Challenge-response authentication scheme

The CTR mode provides confidentiality, but integrity of the message is still not guaranteed, since encrypted message can be blindly modified in random way. Moreover, the attacker can guess the meaning of the message, obtain the cipher stream and prepare a spoofed message to perform malicious actions. Such guesses cannot be excluded, because

industrial applications often behave in predictable way, e.g. issue a "read configuration" command at the beginning of the connection. To avoid spoofing, integrity of the received messages must be confirmed. In the presented solution, the integrity of the transmission is guaranteed by HMAC (Hash-based Message Authentication Code) [27].

In the mDCS system, encryption and decryption on the client side is carried out by an intermediary service application which creates a secure channel between the server and web client (Fig. 4). The application is available for the web browser at localhost address (127.0.0.1), at a port selected by the user (e.g. 32000). The intermediary application also handles authentication. When the url address of the application is entered in the web browser (http://localhost:32000) the application returns the welcome page asking for a login. After the user enters login and password the authentication process proceeds as above (intermediary itself provides IP address of the server). If the login is successful, the intermediary encrypts all data transmitted to the server with the session key. It does not interpret messages sent by the browser or server, but if an invalid message is received, the connection is closed and another user authentication is required.

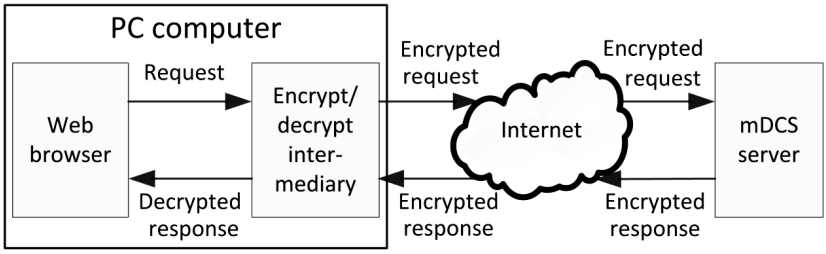

Fig. 4. Web access to mDCS server

4. Communication Model

Various classes of Petri nets as well as queuing networks [31] are commonly used recently as a tool for modelling miscellaneous aspects of communications networks. Moreover new approaches like multicriteria decision making methods are used in the process of design of such networks [15]. In this paper Hierarchical Timed Coloured Petri Nets (HTCPN, [12]) are applied. The model of communication between server and slaves presented in [22] is extended here to include communication with the web client and encryption/decryption processing. Thus, the impact of the encryption delay on performance parameters can be estimated. Essential parts of the model of the whole mDCS communications are shown in Figure 5. The upper part represents communication with a web client, center part communication of the server with PLC modules, and the bottom part the internal PLC transmissions (three are shown).

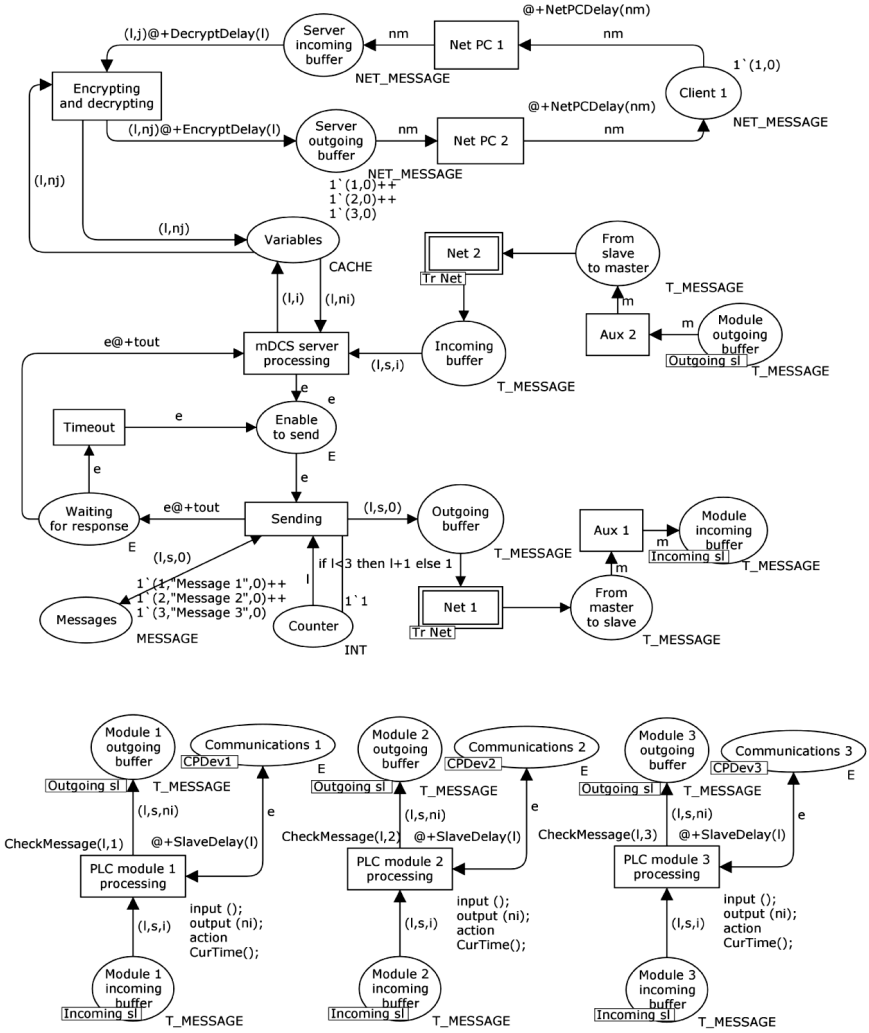

Fig. 5. HTCPN model of mDCS communications

The model will be now briefly explained, beginning with the server-PLC communication. Messages represented by the tokens in *Messages* place (center) are successively sent by the server through the *Net 1* substitution transition to the slave. The `CheckMessage` function in the guards of *PLC module processing* transitions (bottom) ensures that only the addressed module receives the message. The PLC processes the message during a communication time slot (when the program execution is finished) and responds after `SlaveDelay` time (*Communications* fusion place connects this part with the virtual machine model described in [19]). The server receives the reply (*mDCS server processing* transition, center)

and updates variable values (*Variables* place). These variables are periodically read by the web client. Communications with the client is encrypted, what introduces an additional delay. The delay is modelled by the `DecryptDelay` and `EncryptDelay` functions (top).

The model can be adjusted with several parameters (not shown in Fig. 5) according to preferences. Some of them may be constant, such as PLC cycle time t_{cycle}, while others are evaluated during simulation. For example, `SlaveDelay` function models the time t_{msg} needed by PLC to process a message. The delay depends on message size, which also affects bus transmission time t_{bus} (Modbus). Some parameters can be randomly chosen with a given probability. HTCPN model can be simulated with the CPN Tools [11].

Latency time t_{client_lat} of the program variable values presented by the mDCS server to web client is one of the estimated parameters. The server periodically acquires data from PLC modules, and stores them in local cache. Web client reads the cached data over the Internet, so presented values may be somewhat outdated. The latency depends on the number of PLC modules, the time t_{prog} the CPDev virtual machine needs to execute the program in PLC, and on network delays. The execution time t_{prog} is shorter than t_{cycle} and may vary between the cycles. To determine how t_{client_lat} depends on t_{prog}, i.e. on the complexity of the control program, some simulations of the model have been performed. The results for t_{cycle} = 200 ms in all three PLCs are shown in Figure 6 (S, grey). As seen, when t_{prog} approaches 200 ms the latency t_{client_lat} quickly goes up. Statistical boxplots consist of (from top): maximum, upper quartile, median, lower quartile, minimum.

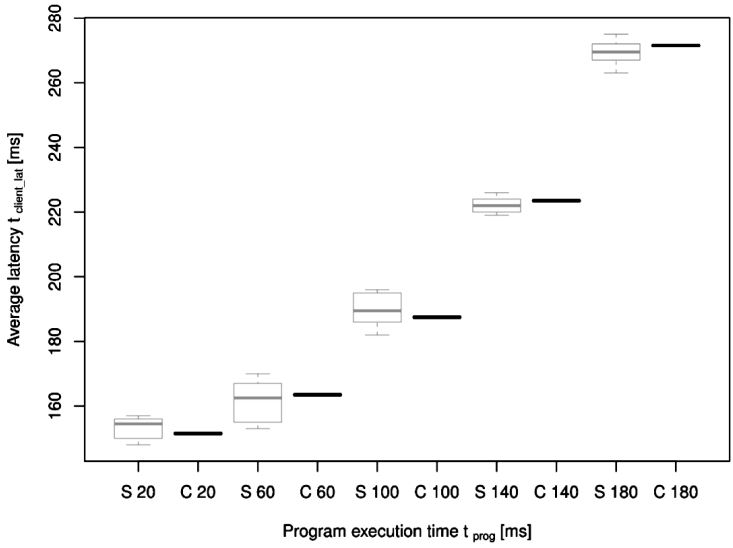

Fig. 6. Comparison of simulation results (S, gray) and calculated values (C, black) for the mDCS system with three PLC modules

Simulation results generated by CPN Tools can be compared with simple analytic calculations. In [22], the latency t_{lat} of server-PLC communication has been investigated. Assuming constant t_{prog}, t_{bus} and t_{msg} for each PLC, the latency t_{lat} can be written as

$$t_{lat} = \frac{nt_{prog}^2}{4t_{cycle}} + (n+1)t_{bus} + \frac{n}{2}t_{msg} \tag{1}$$

To evaluate t_{client_lat} one has to add delay t_{encr} introduced by encryption/decryption of the cached data, and transmission time t_{PC_net} over Internet. Therefore

$$t_{client\_lat} = \frac{nt_{prog}^2}{4t_{cycle}} + (n+1)t_{bus} + \frac{n}{2}t_{msg} + t_{encr} + t_{PC\_net} \tag{2}$$

For the following data, $t_{cycle} = 200$, $t_{bus} = 10$, $t_{msg} = 5$, $t_{encr} = 20$, $t_{PC\_net} = 80$ and varying t_{prog}, the latency calculated from this equation corresponds well to simulations (bars to the right from boxes in Fig. 6).

5. Conclusion

Small distributed control systems do not provide enough computing power and resources to accommodate typical security mechanisms for web access. It has been shown that a sufficient level of security can be achieved in such systems by using dedicated techniques. The challenge-response authentication has been applied to generate session key for symmetric cryptography. To make such a solution browser-compatible, an intermediary application is required at the web client side. Simulations and tests for lab mDCS prototype system have shown that latency time introduced by encryption/decryption algorithms is acceptable even for a 8-bit microserver.

References

[1] *Advanced Encryption Standard (AES).* Federal Information Processing Standards Publications 197, National Institute of Standards and Technology, Nov. 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] Baillieul J., Antsaklis P.J., *Control and Communication Challenges in Networked Real-Time Systems*. Proceedings of the IEEE, vol. 95, No. 1, Jan. 2007, 9–28.

[3] Barry R., *Using the FreeRTOS Real Time Kernel – A Practical Guide*. 2010.

[4] Carcano A., Coletta A., Guglielmi M., Masera M., Nai Fovino I., Trombetta A., *A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems*. IEEE Transactions on Industrial Informatics, vol. 7, No. 2, May 2011, 179–186.

[5] Cheminod M., Pironti A., Sisto R., *Formal Vulnerability Analysis of a Security System for Remote Fieldbus Access*. IEEE Transactions on Industrial Informatics, vol. 7, No. 1, Feb. 2011, 30–40.

[6] Chien-Liang Lai Pau-Lo Hsu, *Design the Remote Control System With the Time-Delay Estimator and the Adaptive Smith Predictor*. IEEE Transactions on Industrial Informatics, vol. 6, No. 1, Feb. 2010, 73–80.
[7] Feng-Li Lian, Moyne J., Tilbury D., *Network Design Consideration for Distributed Control Systems*. IEEE Transactions on Control Systems Technology, vol. 10, No. 2, Mar. 2002, 297–307.
[8] Hajduk Z., Sadolewski J., Trybus B., *FPGA-based Execution Platform for IEC 61131-3 Control Software*. Przegląd Elektrotechniczny (Electrical Review), R. 87, No. 8/2011, 187–191.
[9] Hespanha J.P., Naghshtabrizi P., Yonggang Xu, *A Survey of Recent Results in Networked Control Systems*. Proceedings of the IEEE, vol. 95, No. 1, Jan. 2007, 138–162.
[10] IEC 61131-3 Standard: Programmable Controllers. Part 3. Programming Languages, 2003.
[11] Jensen K., Kristensen L.M., Wells L., *Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems*. International Journal on Software Tools for Technology Transfer (STTT), vol. 9, No. 3–4, Jun. 2007, 213–254.
[12] Jensen K. Kristensen L.M., *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin, Heidelberg, 2009.
[13] *Mini-Guard Ship Control & Positioning System*. Praxis Automation Technology B.V., 2011. http://www.praxis-automation.com
[14] Nai Fovino I., Carcano A., Masera M., Trombetta A., *An experimental investigation of malware attacks on SCADA systems*. International Journal of Critical Infrastructure Protection, vol. 2, No. 4, 2009, 139–145.
[15] Olejnik R., *Computer network design based on AHP method*. In: Jałowiecki P., Łukasiewicz P., Orłowski A. (Eds), Information systems in management, vol. 11, Warsaw University of Life Sciences Press, Warsaw, 2011, 80–89.
[16] Ralston P.A.S., Graham J.H., Hieb J.L., *Cyber security riskassessment for SCADA and DCS networks*. ISA Transactions, vol. 46, No. 4, Oct. 2007, 583–594.
[17] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: *Programming controllers in Structured Text language of IEC 61131-3 standard*. Journal of Applied Computer Science, vol. 16, No. 1, 2008, 49–67.
[18] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Mini-DCS System Programming in IEC 61131-3 Structured Text*. Journal of Automation, Mobile Robotics & Intelligent Systems, vol. 2, No. 3, 2008, 48–54.
[19] Rzońca D. Trybus B., *Hierarchical Petri Net for the CPDev Virtual Machine with Communication*. In: Kwiecień A., Gaj P., Stera P. (Eds.) Computer Networks 2009, Communications in Computer and Information Science 39, Springer-Verlag, Berlin, Heidelberg, 2009, 264–271.
[20] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Open environment for programming small controllers according to IEC 61131-3 standard*. Scalable Computing: Practice and Experience, vol. 10, No. 3, Sep. 2009, 325–336.
[21] Rzońca D., Sadolewski J., Trybus B., *Coloured Petri-nets models of CPDev soft controller with I/O boards*. Przegląd Elektrotechniczny (Electrical Review), R. 86, No. 9/2010, 170–173.
[22] Rzońca D., Stec A., Trybus B., *Data Acquisition Server for Mini Distributed Control System*. In: Kwiecień A., Gaj P., Stera P. (Eds.), Computer Networks 2011, Communications in Computer and Information Science 160, Springer-Verlag Berlin Heidelberg, 2011, 398–406.
[23] Rzońca D. Stec A., *Small Prototype Acquisition System with Secure Remote Data Access*. Annales UMCS Informatica AI, vol. XI, No. 3, 2011, 87–100.
[24] *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publications 180-3, National Institute of Standards and Technology, Oct. 2008. http://csrc.nist.gov/publications/fips/fips180-3/fips180-3 final.pdf
[25] *SMC programmable controller*. Lumel S.A., 2010. http://www.lumel.com.pl/en
[26] Stinson D.R., *Cryptography: Theory and Practice*. Chapman and Hall/CRC, Nov. 2005.

[27] *The Keyed-Hash Message Authentication Code (HMAC)*. Federal Information Processing Standards Publications 198-1, National Institute of Standards and Technology, Jul. 2008. http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1 final.pdf
[28] Turan M.S., Barker E., Burr W., Chen L., *Recommendation for Password-Based Key Derivation Part 1: Storage Applications*. National Institute of Standards and Technology Special Publication 800–132, Dec. 2010. http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
[29] Yodyium Tipsuwan Mo-Yuen Chow, *Control methodologies in networked control systems*. Control Engineering Practice, vol. 11, 2003, 1099–1111.
[30] Yuen P.K., *Practical Cryptology and Web Security*. Pearson Education, 2005.
[31] Zatwarnicki K., *Identification of the Web Server*. In: Kwiecień A., Gaj P., Stera P. (Eds.), Computer Networks 2011, Communications in Computer and Information Science 160, Springer-Verlag, Berlin, Heidelberg, 2011, 45–54.