computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

IMPLEMENTATION OF SYMMETRIC CRYPTOGRAPHY IN EMBEDDED MEASUREMENT SYSTEMS

Wiesław Winiecki [1], Piotr Bilski [1),2)]

[1)] Institute of Radioelectronics, Warsaw University of Technology, Warsaw, Poland, W.Winiecki@ire.pw.edu.pl
[2)] Department of Applied Informatics, Warsaw University of Life Sciences, Warsaw, Poland, piotr_bilski@sggw.pl

Abstract: The paper presents the implementation of the symmetric cryptography in the distributed measurement system. Motivation for increasing the security in such systems is presented. Next, fundamentals of cryptography from the measuring systems' point of view are introduced. The role and structure of embedded systems in modern distributed environments is explained. As the example of presented problems, implementation of the AES algorithm on the Compact RIO module is presented and tested for both efficiency and accuracy. The paper is supplemented with conclusions and future prospects of the approach. *Copyright © Research Institute for Intelligent Computer Systems, 2015. All rights reserved.*

Keywords: measurement, cryptography, embedded systems, distributed measurement systems.

1. INTRODUCTION

Importance of contemporary computer-based measurement systems steadily increases with their new applications. Currently, industry and society-oriented installations (healthcare, power plants or water distribution) widely use computer devices to monitor and control processes of critical importance for large groups of citizens. The Distributed Measuring System (DMS) may become the target for terrorists to maximize casualties among civilians by, for example, damaging the nuclear plant or poisoning the water supply. Therefore creating reliable and safe systems becomes more important [1]. This includes protecting data in conditions of the possible unauthorized access to the infrastructure [2].

Software methods of increasing safety for the transmitted data are authentication and authorization of users, imposed by the operating system and cryptographic protocols. The latter are widely investigated. New algorithms of encrypting and decrypting data are developed. For the needs of DMS, integrated programming environments, such as LabVIEW or Lab Windows/CVI are used. The challenge of ensuring security in DMS is its versatility. It contains desktop computers or workstations (processing nodes) and small units (measurement or execution nodes), such as sensory networks or embedded systems. This requires implementing cryptographic algorithms separately for each node. Attempts to adjust the encryption system to the multi-core server have been made [3]. It is important to verify the ability to secure the data in small, measurement nodes.

The paper presents the implementation of such an algorithm in LabVIEW to the industrial module responsible for measurements and control actions on monitored objects [4]. It must be adjusted to the node with small computational power and limited power supply. In section II, the structure of the secure DMS is presented, stressing the role of embedded systems in measurements processing. Cryptographic methods for ensuring security are presented here. Section III describes the implementation of the Advanced Encryption System (AES) to the embedded platform. Section IV presents the laboratory test stand, in which verification and testing of the algorithms were performed. In Section V experimental results are described. Section VI contains conclusions and future prospects.

2. STRUCTURE OF DMS

To ensure security in the DMS, the first step may be to isolate it from the outside world. This can be achieved by separating the transmission medium used in the system, either physically, or by the firewall. If the system consists of multiple geographically distant nodes, this may be not enough. The intruder can infiltrate the DMS and connect to the medium (the wireless communication makes it even easier). Then, he would be able to

intercept messages between the nodes to find out the syntax of communication protocols or discover values of measured parameters. Another attack would consist in pretending to be the supervisory node, sending control messages not intended by the DMS designer or legitimate operator.

Standards (such as the SSL protocol) used in the computer networks allow for automated security mechanisms. Unfortunately, in the DMS there are specialized devices, which do not have the embedded encryption and decryption methods. It is true for sensory networks and small nodes (microcontrollers), where the limited power supply (such as battery) imposes the minimum processor usage. In such systems cryptographic schemes were omitted, as they increase the load of computing units. Specialized protocols used in communication between such nodes are also often deprived of required security mechanisms. It is then justified to develop and implement efficient secure algorithms for such devices [5].

Contemporary DMS are inexpensive as most of their modules are computer-based devices running the flexible software installed. Security challenges of the DMS depend on its structure. As in the traditional computer network, the intruder may attempt to intercept data transmitted between nodes, or take control over them. When specialized devices are used, there is the need to implement cryptographic algorithms in hardware (such as FPGA) or in the specific unit (intelligent sensor). The FPGA structures are currently the fastest solutions available [3]. Because of small dimensions, they can be used in the embedded system. The traditional method of configuring the FPGA system is to design it using VHDL. As this is the time-consuming process, compilers are developed to generate the logical gates configuration from the code of the higher level language. The example of such an approach is the FPGA module for LabVIEW, aimed at configuring gates in the CompactRIO platform. It is flexible industrial computer equipped with the embedded processor and the FPGA array. The advantage of the compilation technique is the short design time. The disadvantage is the size and speed of the resulting configuration, depending on the LabVIEW FPGA compiler.

The main problem of the measurement server [3] is processing large amount of information, coming from various sources simultaneously. It decrypts the measurement data from distant nodes and encrypts commands for them. The problem of measurement nodes is the limited processing capability required to encrypt the data and decrypt commands online. While the server is equipped with multi-core processors, the embedded system contains the small single-core unit. It is close to the monitored object and must work in the Real-Time mode to analyze and interact with the environment on-line. The module is required to work under the Real-Time Operating System (RTOS) to ensure timely execution of particular measurement tasks. In the presented research the National Instruments CompactRIO 9073 was selected (see section IV), which fulfills these requirements.

The main drawback of measurement nodes is their limited processor speed and flexibility. Introducing additional (cryptographic) task for such small computers is difficult. While large computers are equipped with powerful processors (such as Intel Core i7), which have the AES algorithm implemented as a set of new instructions, executing modules rely on their slow processing units or use the external modules responsible for data encryption and decryption. A good example is the ARM9 microcontroller, with the hardware AES implementation, which can be used in the sensory networks, also using the wireless computer network. The CompactRIO platform is flexible enough to run the additional algorithm using the onboard computer, as presented in the paper.

3. IMPLEMENTATION OF THE AES IN THE COMPACTRIO PLATFORM

The AES is the most popular and secure symmetric system used in the professional industrial applications. Therefore it was selected to provide the security in DMS from the execution modules perspective. There are multiple alternatives for this solution, such as Blowfish, DES [6], IDEA or Twofish. Most of these standards are currently of limited use. Their disadvantages include:

- too short encryption/decryption keys, which, because of the increasing computational power, are easy to break. These are 56 bits for DES or 128 bits for IDEA, while currently required symmetric key lengths are 192b, 256b, or more.
- too small amount of data to be processed during one algorithm execution. This requires a greater amount of iterations to encrypt or decrypt the same vector. This is the problem of DES (and its successor 3DES), Blowfish and IDEA, where 64-bit plaintexts are processed.
- existence of so-called “weak keys” (the bit combinations especially easy to break) [7], and existing types of attacks, which decrease the usefulness of IDEA, Blowfish and DES.

The strongest option is Twofish algorithm [8], comparable to AES. As it was not selected for standardisation, AES was selected for experiments presented in the paper.

The following section describes the general AES working scheme and details of its implementation in the LabVIEW code run on the CompactRIO module.

AES Description

The AES algorithm is currently the dominant symmetric block cipher, approved by the National Institute of Standards and Technology in 2001 [9]. The cipher replaced its less secure predecessor, Data Encryption Standard (DES). The encryption scheme of AES takes 128-bit data blocks and 128-, 192- or 256-bit key at the input. It produces 128-bit cryptogram on the output. The decryption scheme takes 128-bit cryptogram and the key presented above as the input and produces 128-bit plaintext on the output. Its hardware and software requirements are relatively small, so AES can be used in all nodes of the considered DMS. In the measurement nodes encryption of data blocks and decryption of commands from the server are done simultaneously. Therefore all devices in the DMS must be able to execute cryptographic schemes fast enough. The implementation of the algorithm is based on the substitution-permutation network. The detailed description of the system is beyond the scope of this paper. Only the general structure of the algorithm will be presented; the full presentation of the system is in [9].

Both encryption and decryption procedures contain analogous operations, but in different order. The encryption operates on the *state* matrix (which is firstly filled with the input vector), performing *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey* operations repeatedly in a loop (the number of its iterations depends on the length of the key *K* and can be 10, 12 or 14). The transformation of the *state* matrix is done using a constant array s-box, shifting rows of the matrix, performing matrix multiplication and performing XOR operation. The last iteration is different than previous ones (as it does not contain the *MixColumns* operation). After the last iteration the *state* matrix is copied to the output. The decryption uses operations *InvShiftRows*, *InvSubBytes*, *AddRoundKey* and *InvMixColumns*, also executed in a loop [9].

Before the data (which for encryption are measurements taken by the embedded system and for decryption it is the cipher coming from the server or control module) are processed, the expanded key *W* must be generated from *K* [6]. The procedure consists in performing operations on four input bytes using s-box and permutation.

3.1. IMPLEMENTATION OF AES IN LABVIEW FOR THE COMPACTRIO

Currently, cryptographic operations are not the part of the LabVIEW environment, therefore there is the need to create them using the G language, or exploit the existing libraries. One of the latter is the Crypto-G library [10]. It does not cover multiple algorithms (for instance, asymmetric systems) and cannot be used under the RTOS. Therefore there was the need to design the proper version of the algorithm for such a system. The structure of the CompactRIO includes the controller (an industrial computer) the FPGA subsystem and I/O modules (Fig. 1). The latter are controlled by the FPGA, which ensures the high speed of operation. Existence of the computer and the FPGA makes possible AES implementation in two ways. The first one includes the code run on the processor of the computer system, which is the slower version of the typical general-purpose processing unit. Its programming technique should be the same as for the PC run under RTOS [11], considering limited abilities of the embedded system. The second approach involves design of the logical gates configuration to execute the encryption and decryption by the hardware. Such a solution promises better efficiency, as the fastest AES implementations to date are created using FPGA or other hardware-based solutions [12,13]. In CompactRIO this may cause a problem, as FPGA, controls I/O modules in the first place. Therefore adding another function to the gates may result in the speed decrease. Also, for too small hardware arrays, the simultaneous implementation of the cryptographic algorithm and I/O operations may be impossible (not enough gates to implement both measurement and cryptographic tasks).

Implementation of the embedded computer-targeted version of the algorithm required preparation of the LabVIEW code correctly executable under RTOS (as the Crypto-G functions cannot be used there). The structure of the code required modifications, mainly consisting in calculating the key string lengths, which are used to select the number of algorithm rounds. Functions calculating the length of the string gave wrong results and had to be corrected in order to execute the key expansion procedure (modified LabVIEW "Expand" function in Fig. 2a and original in Fig. 2b). Modifications increased complexity of the code, but made it usable under RTOS.

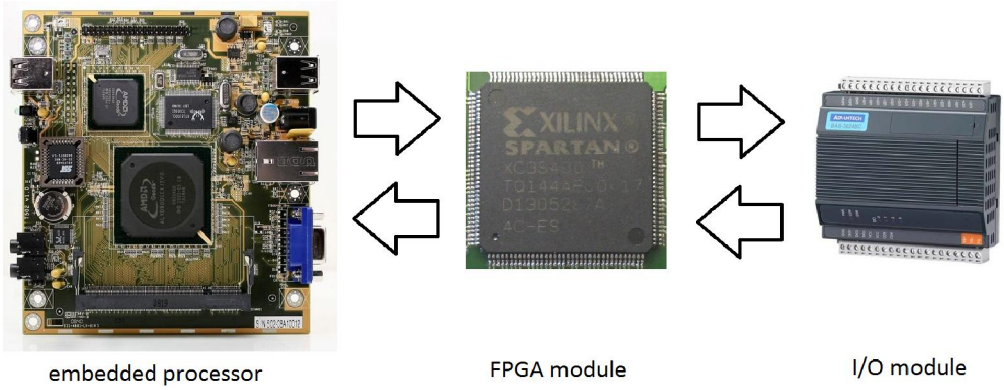

Fig. 1 – Structure of the CompactRIO platform.

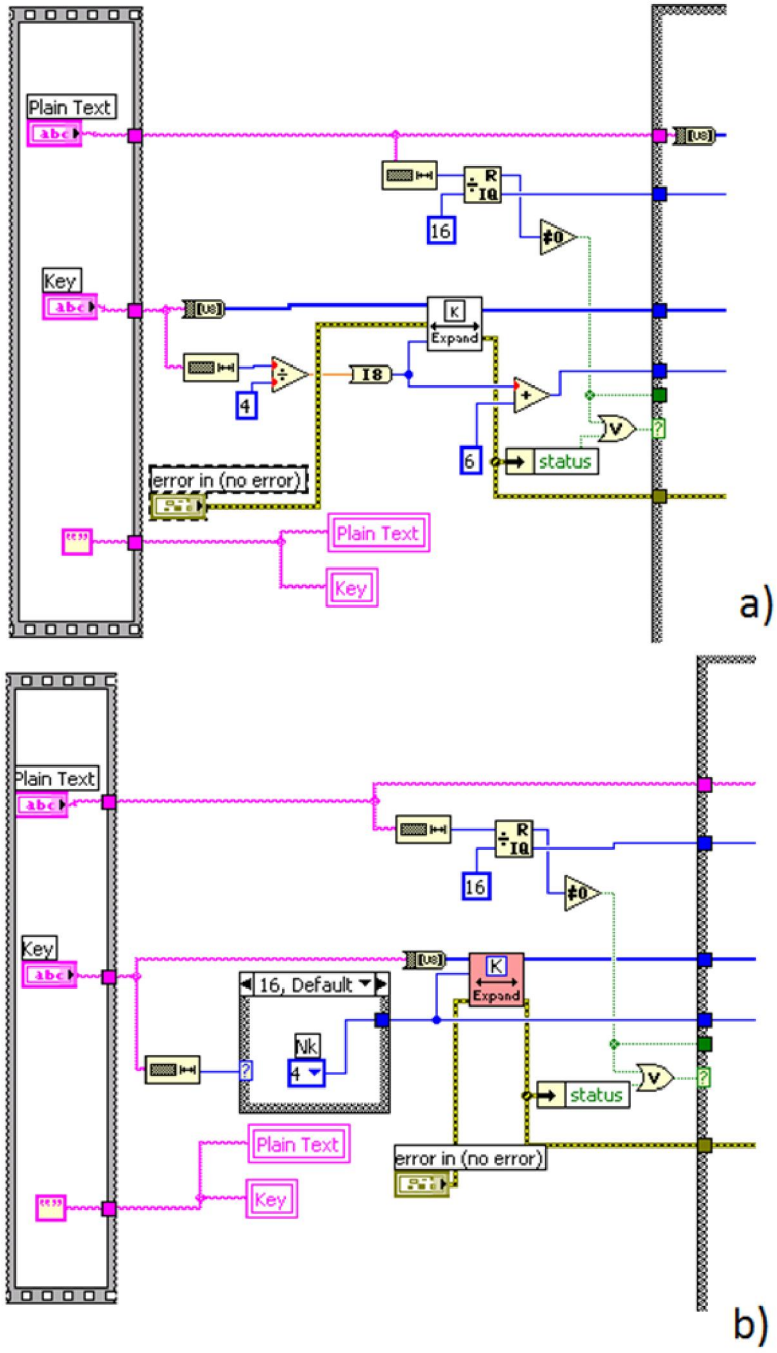

Fig. 2 – Comparison of the RTOS-executable (a) and original (b) code.

Another version of the algorithm prepared to be executed in the computer of the embedded system was the multithreaded AES, where each s-box processing operation was divided into four parallel suboperations, finally assembled to obtain the result. The theoretical assessment of the concurrent algorithm suggests that dividing the data stream into parallel substreams should increase speed of operation. Disassembling data structures into simple elements also requires time, which might be long enough to make the whole approach inefficient. This solution was successful in the multicore server system [11,12], where each processor's core was able to process its data block concurrently. Although there were two versions of this code, i.e. implicitly and explicitly assigning threads to the particular core, only the first one was useful in the embedded system. The second version was unable to correctly encrypt and decrypt the data (as the deterministic loops did not work with the hardware). The efficiency of this solution (Fig. 3) on the simple computer was verified.

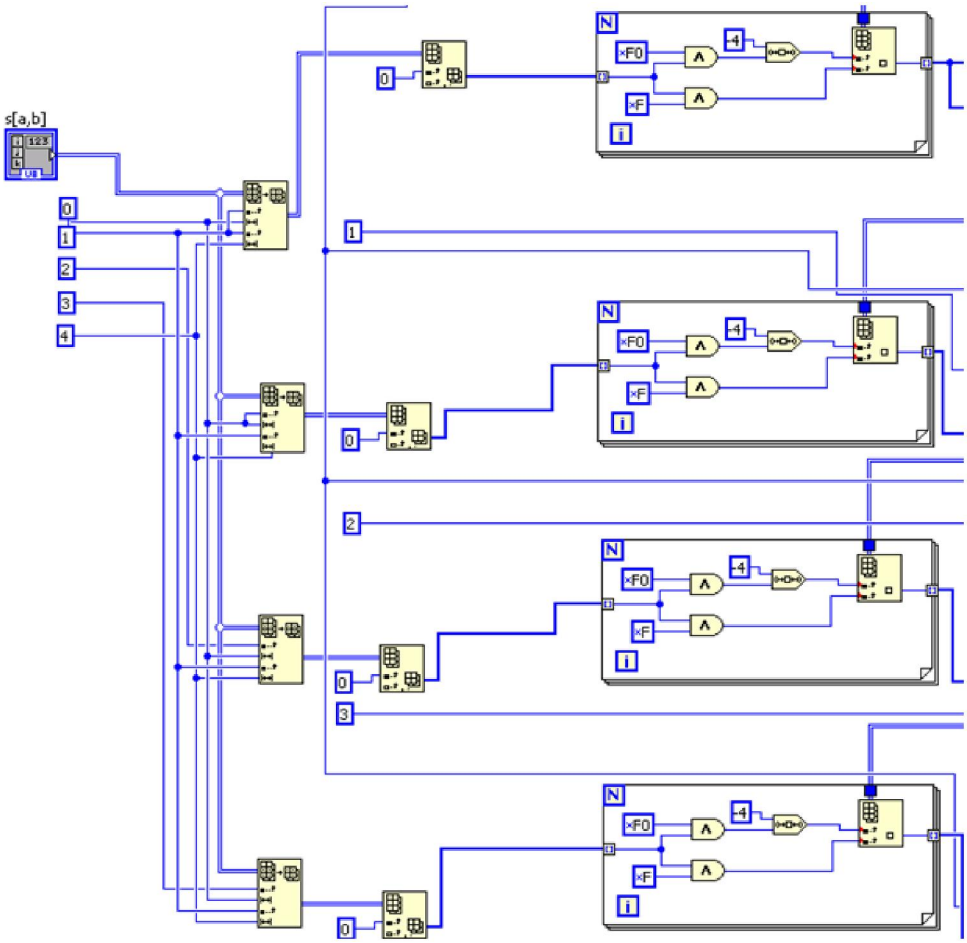

Fig. 3 – Exemplary code of the parallel AES operation.

The FPGA implementation of the AES for the CompactRIO platform was more difficult. The programming library for this system is limited and multiple functions available in general version of LabVIEW are not executable on FPGA. Therefore most functions had to be redesigned to meet the system's requirements. The main limitations and modifications of the implemented code are:

- Inability to use the string variables and constants. The LabVIEW implementation of AES works on the character chains, which are internally converted into the byte array. Therefore the conversion of the string or number into the byte array must be done before the data are sent to the FPGA system. This imposes design of the two-part code. The first one is run on the CompactRIO computer, where data are prepared for processing and encryption/decryption results are converted to the required form. The second is the FPGA code processing the plaintext or the cipher. Communication between both systems is assured using FIFO variables or by specialized functions – see Fig. 4. The conversion is not required when only numerical data (here unsigned integers) are transmitted
- Usage of only one-dimensional data. As AES operation is based on the s-boxes processing, which are two-dimensional arrays, the modification of the code changed arrays into vectors. This complicated the structure of the

operation, including addressing of the particular array elements and modifying matrix structures (for instance, transpositions also had to be eliminated). Instead, only vector operations were implemented.

- Inability to use the dynamic structures. The number of iterations in the for loop has to be explicitly declared in the code and cannot depend on the currently processed data size. All vectors incoming from the computer system must be of fixed length. Because AES works with three key lengths, three different versions of the encryption and decryption functions had to be prepared, differing in the size of data structures (key and expanded key lengths) and the number of AES rounds (10, 12, or 14, respectively).
- Usage of the floating point numbers is not allowed. Therefore any functions producing float or double values must be exchanged into the fixed point counterparts. This is especially important for the division operations.
- Multiplication of array elements using one operation only is not allowed. Therefore such operations must be done using "for" loops.

The example of the code for the FPGA system is in Fig. 5.

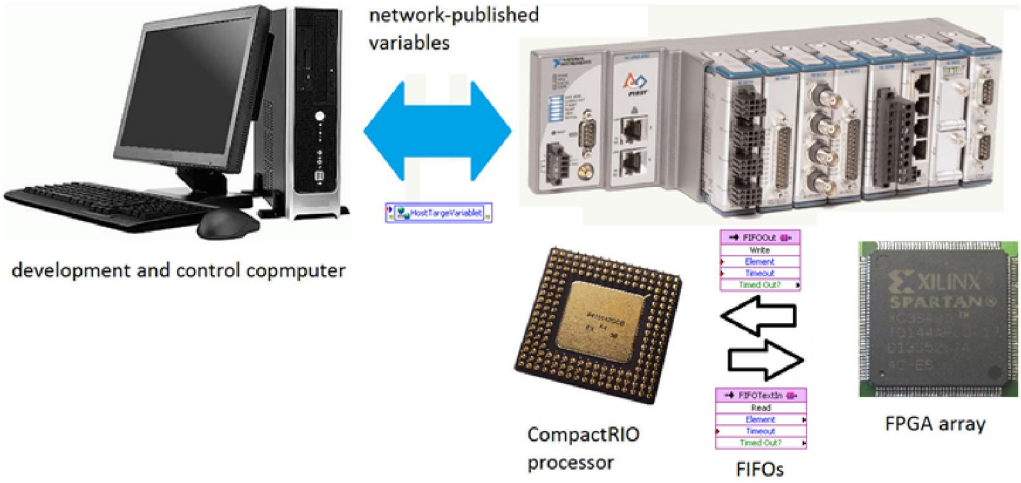

Fig. 4 – Architecture of the system implementing AES in the FPGA array.

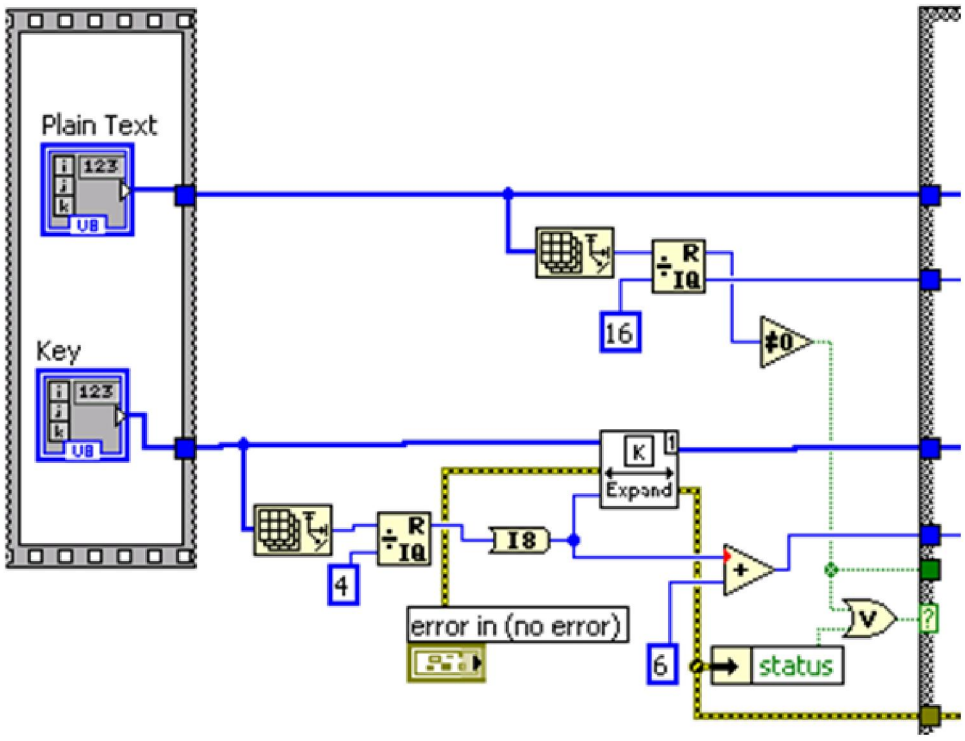

Fig. 5 – The fragment of the AES code for the FPGA implementation.

The additional problem is the selection of the coding mode. As AES is the block cipher, there is the need to ensure that the repeated sequences of the encrypted data will not produce repeated cryptograms, which might compromise the system. In practical applications, extensions to the basic Electronic Code Book (ECB) code have to be verified, including Cipher Block Chaining (CBC), Cipher Feedback (CFB) and other modes [14]. Implementing them increases security of the system, although is an additional task for the processor. Description of experiments using coding modes is beyond the scope of this paper.

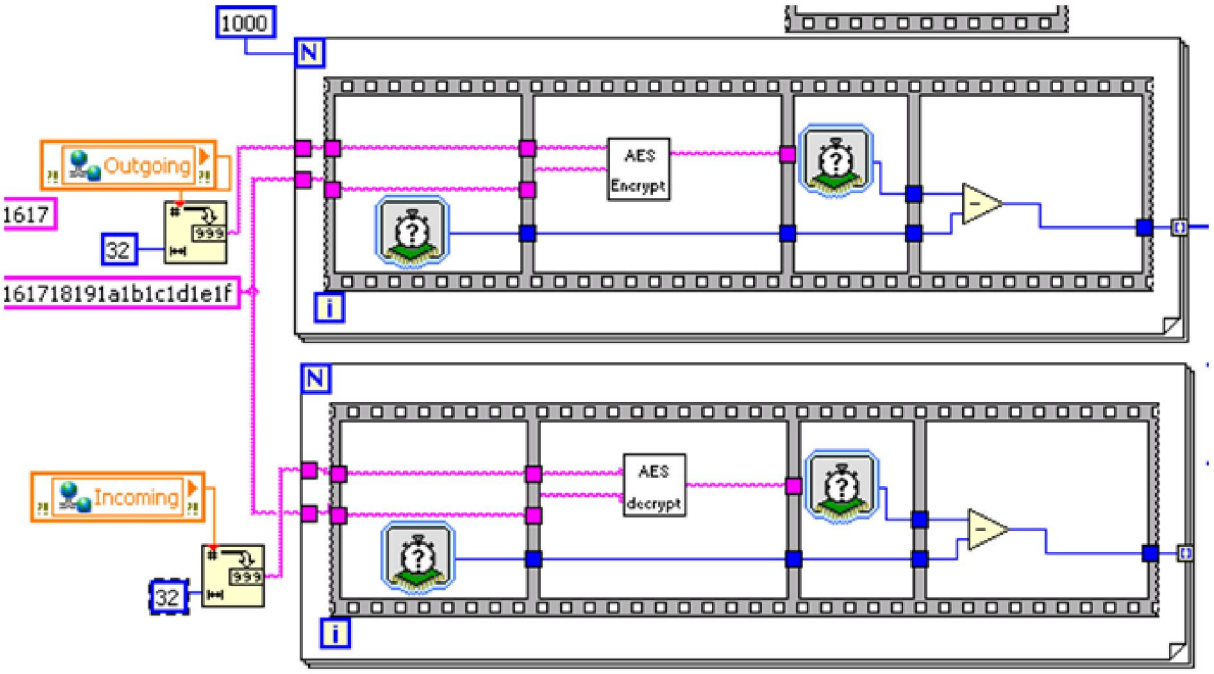

Fig. 6 – Code for the AES efficiency measurement under embedded processor.

4. LABORATORY TEST STAND

The following equipment was used to create the AES-capable embedded system:

- National Instruments CompactRIO 9073 modular instrument. It is equipped with PowerPC 266 MHz processor and FPGA module. The latter can be used to control the modules installed in the chassis, for example initiate the samples generation or acquisition. The module placed in the chassis was NI 9205 DAQ, able to acquire 250kS/s. The whole system works under RTOS: Wind River VxWorks [15].
- Personal computer with LabVIEW installed, acting as the design platform of the software. It was equipped with Intel T2300 processor and 2.5 GB of RAM. This machine also worked as the server obtaining samples from the measurement node. The computer works under Windows XP.
- The Agilent 32210A function generator, able to generate sinusoidal waveform up to 20 MHz, which is enough to test the abilities of the CompactRIO module.
- Both computers were connected using the 100Mb/s Ethernet network, ensured by the simple switch. The planned experiments included:
  - encryption and decryption of the single 16-byte block using single- and multi-core versions of the algorithm. This is the simulation of the real situation when the subsequent samples are put into the block and then encrypted. The decryption of the incoming instructions is also simulated. As both operations have analogous structure (see Section IV), it is assumed that both have similar computational complexity.
  - encryption of the samples acquired by the DAQ module of the Compact RIO instrument. The FPGA-driven acquisition obtains new samples with the constant speed, determined by the abilities of the hardware. The embedded system encrypts them on-the-fly and sends to the server. This experiment verifies the practical speed of the Compact RIO module, as duration of sending samples through the network depends not only on the encryption speed, but also DAQ efficiency.

The software created for the embedded system cooperates with the analogous software of the measurement server (described in [11]). As the decryption and decryption are asynchronous (i.e. are executed only when the new samples are acquired or when the new command is sent from the server), they must be placed in separate threads of the code.

The fragment of the prepared G-language diagram with two independent data streams is in Fig. 6.

5. EXPERIMENTAL RESULTS

Experiments described in section IV were performed using all three lengths of keys. The time of each operation was taken using the "TickCount" function with the µs precision (which is currently the greatest resolution of the RTOS) – see Fig. 6 and 7. It allows to measure duration of the single encryption and decryption, so no additional loops are required, as in [16]. The code for the FPGA version measurement included functions invoking the VHDL code (Fig. 7).

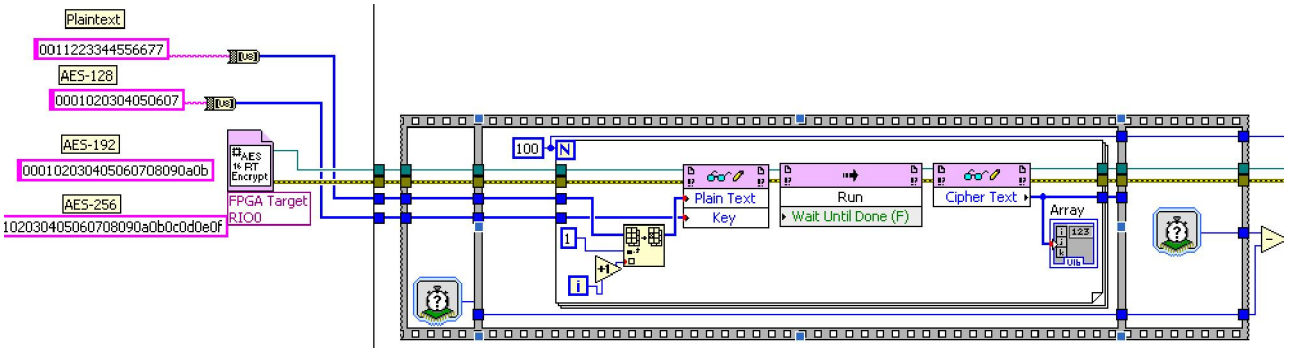

Fig. 7 – Code for the AES efficiency measurement under FPGA.

5.1. EXPERIMENTS DESCRIPTION

The first experiment was measuring the simulated samples, i.e. the single 16-byte block, consisting of the following sequence: "00112233445566778899aabbccddeeff" (where each element is a hexadecimal digit). As the duration of the single encryption and decryption depends on the state of the RTOS, both operations were repeated 1000 times to see, what is their average duration and standard deviation. Results of the single sweep of the single-core encryption are in Fig. 8, while the average durations for different keys are in Tab. 1. Results confirm that both operations require similar amount of time to complete. Note that the maximum rate of sending encrypted samples is about four samples per second (for the 32-bit representation of the real number), which strongly limits the DAQ module efficiency. Results of the multi-core operations show that the multi-threaded code is not a good solution here, as it slows down the whole operation. Although the concurrent s-box transformation execution should speed up the whole process, the data transfer connected to division of the initial array into subarrays requires too much time. The multithreaded operation is slower and as long as only one processor with a single core is installed in the device, imposing parallel computations is not recommended.

Additional observation from Fig. 8 is that most iterations of the cryptographic operation have similar duration, which is never longer (sometimes, if RTOS has free resources - shorter) than the particular value (as requires Hard Real-Time mode). The standard deviation is also small – in the case of Fig. 8, it is only 30.89 ms. This phenomenon is present only in systems under RTOS, GPOS, such as MS Windows produce much greater spread of durations [16].

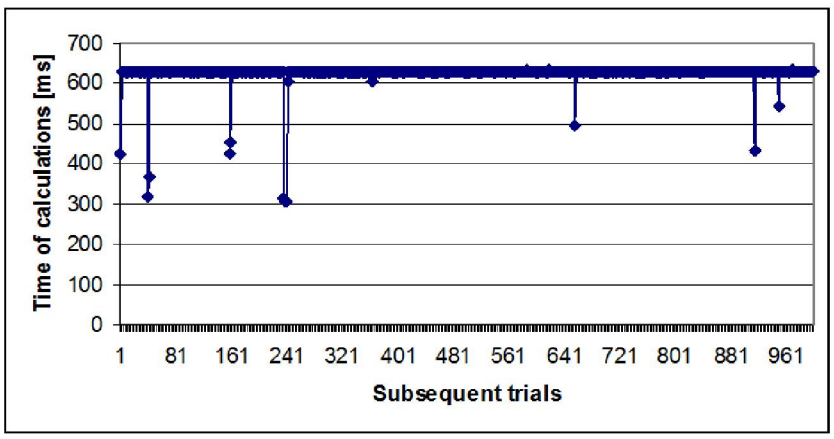

Fig. 8 – Execution of the single-core encryption operation for the 256-bit key.

Table 1. Results of the Single String Encryption and Decryption Efficiency Using the Computer-Implemented AES

Key length	Encryption time [ms]		Decryption time [ms]	
	Single-thread	Multi-thread	Single-thread	Multi-thread
128 bit	226.16	295.13	230.01	289.04
192 bit	518.06	590.21	504.35	588.73
256 bit	625.97	653.47	612.85	647.38

Experiments including the FPGA version of the algorithm were performed in the similar fashion. The FPGA code was invoked 100 times to encrypt or decrypt the predefined vectors. Such an experiment was repeated 100 times to verify the repeatability of results. To ensure that the functions work properly, during each experiment, the input vector was changed so the algorithm had to work with various content. The decryption or encryption results were immediately visible on the front panel to check if the operation is executed correctly. Then average durations were calculated. Results for the FPGA module are in Table 2. Duration of the encryption and decryption is also similar here and depends on the key length (which determines the number of rounds to be executed).

Table 2. Results of the Single String Encryption and Decryption Efficiency Using the FPGA-Implemented AES

Key length	Encryption time [ms]	Decryption time [ms]
128 bit	8.68	8.67
192 bit	11.65	11.74
256 bit	15,98	16,04

The second experiment consisted in acquiring the whole waveform from the analog input module. After the predefined number of samples is gathered, they are encrypted and sent through the network to another node (for instance, measurement server). The analog signals measured the I/O modules in the CompactRIO platform are represented as unsigned integers and depending on their length, their various number can fit into one input block for the AES procedure. Therefore processing the whole waveform is much longer. Results for the 1024 32-bit samples waveform encryption and decryption using the microprocessor are in Table 3. In this case the data are only numerical, so the conversion between various types is not necessary here. Therefore the versatile implementation of the algorithm works only on numbers and conversion functions should be implemented in the outside code. Again, multi-threaded version of the scheme is slower than its simpler counterpart.

Table 3. Results of the 1024-Samples Waveform Encryption and Decryption Efficiency Using the Computer-Implemented AES

Key length	Encryption time [s]		Decryption time [ms]	
	Single-thread	Multi-thread	Single-thread	Multi-thread
128 bit	51.07	57,46.13	50.03	57.97
192 bit	134.02	152.96	134.84	151.82
256 bit	160.41	167.59	159.22	168.31

5.2. DISCUSSION

The FPGA implementation of the AES encryption and decryption is faster than the processor version and can be used to immediately process the acquired samples from the I/O modules. Comparing results with the multi-core server, it shows the superiority of the FPGA module, which is even 5 times faster than the Core2Quad Q6600 processor (2,4 GHz). However, the encrypted data must be sent to the embedded computer (see Fig. 4), which will be able to transfer them further. It is difficult to assess the time of transmission between the FPGA and the computer, although it is much shorter than the AES execution duration. The main problem with this implementation is that the gate array is responsible for the communication with hardware modules and it might be not enough for some applications, where multiple operations must be executed by FPGA. This depends on the particular embedded system and application. Also the disadvantage of the FPGA version of any algorithm is that the programming functions available for this hardware platform are strictly limited, imposing much greater amount of work from the programmer to obtain the same result. The final problem is the efficiency of the LabVIEW VHDL compiler. It is currently extremely slow, requiring 12 to 48 hours to complete the logical gates configuration (depending on the complexity of the code and the speed of the computer executing the task). The future implementations of this tool are expected to be much faster.

The efficiency of the presented secure system is influenced by the application of the algorithm to the particular data. As it works with the data blocks of the constant size, it is reasonable to use the so-called burst mode, which will encrypt as many samples as possible. If the whole waveforms are gathered, the samples should be inserted into 128-bit blocks and then processed. When the scalar values are obtained, the system should wait until the whole block is full before performing encryption. The number of samples encrypted at the same time depends on their size. For instance, four 32-bit samples can be processed with a single algorithm execution. The

decryption should be executed immediately after the data are obtained.

The work regime of the CompactRIO platform and its application suggest the optimal structure of the code implemented both in the microprocessor and FPGA subsystems. Because the encrypted instructions from the supervisory modes arrive through the network to the computer which interprets and executes them, they should be decrypted as soon as possible. Therefore the decryption scheme should be executed by the processor. The measurement data are obtained by the DAQ hardware, therefore they can be encrypted by FPGA. Such a solution balances the computational load within the CompactRIO module between all its elements.

6. CONCLUSIONS

The AES algorithm [17] presented in the paper can be used to create a secure distributed measurement system. However, depending on the type of the node considered, various problems arise. The measurement server must process multiple data from different sources. The embedded system must be fast enough to send the acquired samples on-line. As limitations of such a computer are numerous, the designer of the system must evaluate abilities of the equipment and select the models suiting the particular computational needs. Also, the multithreaded version of the implemented algorithm doesn't work well on the single core processor. As the block ciphers are successfully implemented using FPGA technology, the latter could be also used to create the algorithm implementation for the Compact RIO module.

Although the presented research was focused on the CompactRIO platform, the conclusions and recommendations are valid also for other, similar equipment. It is apparent that hardware implementations of the algorithm are much faster than its versions for the microprocessor. The latter is more flexible and can be used when the former is not available (because of, for instance, supporting data acquisition). However, combining both solutions can bring a novel quality in the DMS and should be investigated in the future.

7. REFERENCES

[1] W. Winiecki, T. Adamski, P. Bobiński, R. Łukaszewski, Security of distributed measurement and control systems, *Przegląd Elektrotechniczny (Electrical Review)*, (LXXXIV) 5 (2008), pp. 220-227. (in Polish).

[2] A. Guruprasad, P. Pandey, B. Prashant, Security features in Ethernet switches for access networks, *Proceedings of the Conference on Convergent Technologies for Asia-Pacific Region TENCON 2003*, October 15-17, 2003, Vol. 3, pp. 1211–1214.

[3] D. Cheng-Hua, J. Jun, W. Xing-Ming, X. Wen-Yuan, Fast S-box substitution instructions and their hardware implementation for accelerating symmetric cryptographic processing, in *Proceeding of the International Conference on E-Business and Information System Security, EBISS'09*, 23-24 May 2009.

[4] P. Bilski, W. Winiecki, T. Adamski, Implementation of symmetric cryptography in embedded systems for secure measurement systems, in *Proceedings of the I2MTC 2011*, Hangzhou, China, 9-12, May 2011, pp. 1288-1293.

[5] A. Biedermann, G. H. Molter, Design methodologies for secure embedded systems, *Lecture Notes in Electrical Engineering*, Springer, (78) (2011).

[6] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, FPGA implementations of the DES and Triple-DES masked against power analysis attacks, in *Proceedings of the Field Programmable Logic and Applications Conference*, Madrid, Spain, Aug. 28-30, 2006.

[7] J. Daemen, R. Govaerts, J. Vandewalle, Weak keys for IDEA, in *Proceeding of the Advances in Cryptology, CRYPTO 93*, 1993, pp. 224–231.

[8] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*, John Wiley & Sons, New York, 1999.

[9] Announcing the Advanced Encryption Standard (AES), Available: www.nist.gov.

[10] Vartor Crypto-G library, available at http://www.vartortech.com/cryptog.html.

[11] P. Bilski, W. Winiecki, Multi-core implementation of the symmetric cryptography algorithms in the measurement system, *Measurement*, (43) (2010), pp. 1049-1060.

[12] O. Gervasi, D. Russo, F. Vella, The AES implantation based on OpenCL for multi/many core architecture, in *Proceedings of the International Conference on Computational Science and its Applications*, 2010, pp.129-134.

[13] T. Good, M. Benaissa, AES on FPGA from the fastest to the smallest, *CHES 2005, Lecture Notes in Computer Science*, (3659) (2005), pp. 427-440.

[14] R. Doomun, J. Doma, S. Tengur, AES-CBC software execution optimization, in *Proceedings International Symposium on Information Technology, ITSim'2008*, 26-28 August 2008, Kuala Lumpur, pp. 1-8.

[15] Build reliable and optimized embedded systems with the world's most widely adopted RTOS, available at: http://www.windriver.com/products/vxworks/
[16] W. Winiecki, P. Bilski, Analysis of the time efficiency assessment in the virtual measurement systems, in *Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications IDAACS'09*, September 21-23, 2009, Rende (Consenza), Italy, pp. 179-184.
[17] J. Daemen, V. Rijmen, *The Design of Rijndael*, Springer Verlag, 2002.

***Prof. Wiesław Winiecki**, M. Sc. ('75), Ph. D. ('86), D. Sc. ('03); Prof. Title ('11), Professor of Measurement Science at the Institute of Radioelectronics, Warsaw University of Technology, has fourty-year research experience in the field of measurement and control systems, including the development and implementation of various kinds of measurement devices and systems. The record of his achievements in this respect comprises more than 200 research publications, 2 monographs, 1 book and 1 academic textbook, as well as over 100 reports on scientific research and implementation. He is a member of the Committee on Metrology and Scientific Instrumentation of the Polish Academy of Sciences and the Vice-President of the Polish Society for Measurement, Automatic Control and Robotics POLSPAR.*

In the years 1994-2001 and 2004-2005, he held the position of Deputy Director for Research at the Institute of Radioelectronics at Warsaw University of Technology (WUT), then, in 2005-2008 he was Vice-Dean for Research in the Faculty of Electronics and Information Technologies, WUT. In 2008, he has been re-appointed as Deputy Director for Research at the Institute of Radioelectronics.

Piotr Bilski, PhD was born in 1977 in Olsztyn, Poland. He graduated from Warsaw University of Technology, Institute of Radioelectronics, obtaining MSc degree in 2001 (with honors), PhD degree in 2006 (with honors) and DSc degree in 2014. Currently he is an Assistant Professor in the Institute of Radioelectronics, Warsaw University of Technology and Department of Applied Informatics, Warsaw University of Life Sciences. His main scientific interests include diagnostics of analog systems, design and analysis of virtual instrumentation, application of artificial intelligence and machine learning methods to the environmental sciences. He is the member of IEEE, IMEKO TC10 and POLSPAR and reviewer for such journals like Measurement, IEEE Transactions on Instrumentation and Measurement, Expert Systems with Applications.