The Control Technology of Integrity and Legitimacy of LUT-Oriented Information Object Usage by Self-Recovering Digital Watermark

Kostiantyn Zashcholkin and Olena Ivanova

Odessa National Polytechnic University, 1, Shevchenko Avenue, Odessa, Ukraine
(const-z@te.net.ua, en.ivanova.ua@gmail.com)

Abstract. The paper proposes the technology of control of integrity and legitimacy of information object usage with the Look Up Table-oriented (LUT-oriented) architecture. The technology is based on embedding the self-recovery digital watermark into information objects of such kind. The technology is the composition of approaches to forming the self-recovery digital watermarks in the passive multimedia containers, and approaches to embedding the extra information into LUT-containers. The process of embedding the extra information occurs with the help of classification of container elements and their purposeful modification within the set of the formed classes. The procedure of immediate embedding the data at the level of LUT-container elementary parts includes the value inversion of current processed LUT unit and propagation of the inversion around all the inputs of LUT units connected to the current unit output. The description of practical realization of the proposed technology is represented.

Keywords: digital watermarks, control of information object integrity, control of information object usage, information security, cybersecurity, IT systems safety, steganography, LUT-oriented architecture, FPGA.

Key terms: Information Technology, Data, Object, Approach, Method.

1 Introduction: Topicality of the Problem and Aim of the Paper

The technologies of digital watermark (DWM) usage are one of the most effective approaches to the comprehensive information security provision [1]. DWMs are mainly used for:

1. the control of information object integrity;
2. the control of information object usage: information object copyright provision, digital content authentication, the tracking of digital content move (including the tasks of source retrieval of information leak).

DWM technologies are based on steganographical technique, with the help of which the fact of DWM presence in an information object (DWM container) is

hidden. At the same time DWM can be read in the container if someone has a stego-key possessing the set of access rules to DWM elements [2].

The *control of information object integrity* by DWM is based on embedding some control data unit allowing to analyze the object integrity in an information object. Hash sum calculated with the help of the definite hash-function is most frequently used as such kind of data unit [3]. However such kind of control unit embedding results in information object change and consequently violates its integrity by itself. Under these conditions the so-called *self-recovering DWMs* possessing the ability to recover the initial object value (a value, which it had before DWM embedding) in the process of DWM reading from the object are used [4]. In order to control the integrity the DWM extraction from information object, hash sum calculation and this hash sum with DWM contents comparison is performed. The analysis of integrity control guarantees the impossibility of the information object substitution or corruption. Finally this provides the *safety of functioning the information system* processing the given information object.

The control of information object usage legitimacy by DWM is based on possibility of a copyright owner to embed DWM having the copyright owner identifying data into the object. So it is only a copyright owner who possesses a stego-key can check the DWM presence and its contents in the information object [5]. DWM technologies are actively used in the structure of Digital Rights Management systems (DRM – systems), but the field of their usage is commonly limited with the multimedia content protection: graphical, audio and video files [6], [7]. In performing the tasks of control of information object usage legitimacy by DWM the recovery of initial object value after DWM extraction is necessary.

In the present paper the author proposes the information technology of self-recovery DWM usage in active hardware containers based on LUT-oriented architecture (further LUT-containers). We can refer, for example, Field Programmable Gate Array microchips (FPGA microchips) [8], which at present are often used as an element base for computer and control system design, to such kinds of containers. The main element of such containers is LUT units, which are the data structure used to replace the calculation with prepared data search operation [9]. LUT units in FPGA are normally represented in the form of RAM. In this case LUT unit inputs are the address inputs of RAM. If the number of inputs equals n a LUT unit stores 2^n bits information and is capable of calculating the value of 1 n-argument Boolean function.

Self-recovering DWMs have been developed and used in the field of control of integrity and legitimacy of usage for passive multimedia containers, e.g. graphical, audio and video files. However it is necessary to control not only the information objects of such kind by DWMs. But the task of self-recovering DWM usage in active containers performing some calculating or controlling function is not realized at the moment. In the articles [10], [11] the techniques of DWM embedding in active LUT-containers are proposed. But the techniques described do not possess the possibility to recover the container original. So *the aim of the given paper* is to describe the possibility of container original recovery after DWM extraction by developing the technique of forming the DWMs in LUT-containers.

2 The Information Technology of Embedding the Self-Recovering DWM in LUT-Container

The information technology considered in the paper is a formal process of usage of the proposed and already existing techniques as well as the means of information processing, which provides DWM forming in LUT-container space.

The proposed information technology uses a set of *Fridrich-Goljan-Du* method (further *Fridrich* method) approaches [12], [13] in the part, which is intended for container recovery, and technique described in [10], [11] in the part intended for DWM embedding in LUT-containers. Besides the represented technology uses such peculiarities of LUT-containers as activeness, accurate data presentation, non-autonomousness of their elementary parts [10].

The methods proposed in [10], [11] are based on the change of codes of a pair of successively integrated LUT units, which does not alter this pair functioning. Assume there is a pair of LUT units and the output of the first one is connected directly or through a trigger to the address input of the second one. The inversion of all the bits of the first unit code and a definite rearrangement of the bits of the second unit code does not change the functioning of the given pair of units. It is the consequence of equivalence of Boolean functions realized by the pair of units before and after values inversion. The rearrangement of bits in the second unit code of the pair is produced according to the rules depending upon the binary weight of its address input, which the first unit output is connected to. The inversion of bits of the first unit code of the pair gives the possibility to achieve the required value in a definite bit of a unite code. This peculiarity is exactly used by the methods considered in [10], [11] for DWM embedding in a LUT-container. However these methods are not able to recover the container original.

Fridrich method [12], [13] is based on the group processing – disjoint subsets of elementary parts of a container with the help of two functions: Flipping-function F and function of discrimination f.

Function of discrimination is used in Fridrich method on order to classify the groups of elementary container parts by including them to the classes of regular, singular and unused groups. To determine the function value in each of the container groups we are to calculate the sum of pairwise subtractions with overlapping for elementary container parts included in a group.

Flipping-function is used by Fridrich method to modify the groups according to some rule possessing the characteristic of involution. In the proposed information technology the LUT-container architecture peculiarities are taken into account and this fact differs it from Fridrich method in the following features:

- within the frames of the proposed technology in the course of DWM embedding the modification of elementary parts exactly (LUT units) is performed but not the groups of elementary parts of a container;
- within the frames of the proposed technology the calculation of Flipping-function $F(x)$ unlike Fridrich method contains the LUT unit value inversion according to the procedure mentioned in [10], [11]. The procedure is the invention of values of the current processed LUT unit and propagation the inversion around the inputs of all

the LUT units connecting to the current unit output. It is obvious that the Fridrich method requirement concerning the presence of involution characteristics is performed for the given type of Flipping-function, i.e. $F(F(\mathrm{LUT}_i)) = \mathrm{LUT}_i$ where LUT_i is any valid value of some LUT unit;

- unlike Fridrich method within the frames of the proposed technology a principle of container element classification without calculating any function of discrimination is determined. The element classification is organized as a determination of zero and one value proportion in LUT unit codes. It is performed according to the following principle: if LUT unit contains zero values more than one values it is classified as *a regular unit (R*-unit); if a unit contains more one values than zero values it is classified as *a singular unit (S*-unit); if the number of zero values of LUT unit equals the number of its one values the unit is classified as *an unused unit (U*-unit).

Certainly, the rules of moving the container from one class to another defined by Fridrich method are performed within the frame of the proposed approach as well:

$$\begin{aligned} F(\mathrm{LUT}_R) &= \mathrm{LUT}_S; \\ F(\mathrm{LUT}_S) &= \mathrm{LUT}_R; \\ F(\mathrm{LUT}_U) &= \mathrm{LUT}_U, \end{aligned} \tag{1}$$

where LUT_R, LUT_S, LUT_U – LUT units classified as *R*-unit, *S*-unit and *U*-unit, respectively; $F()$ – Flipping-function.

The succession of actions in the proposed information technology in DWM embedding in LUT-container is as follows.

Stage 1. The movement about LUT-container units in the order determined by embedding path is realized. During this action each of the LUT units refers to the classes *R*-unit, *S*-unit or *U*-unit on the basis of the above mentioned classification rules. According to the results of classification the binary *RS*-vector is formed in the following way: if the next LUT unit is classified as *R*-unit then zero value is fixed in *RS*-vector; if the next LUT unit is classified as *S*-unit then one value is fixed in *RS*-vector; *U*-unit values are not fixed in *RS*-vector.

Stage 2. The obtained *RS*-vector compression is performed with the help of some lossless compression algorithm. As a result of this the compressed vector RS_{com} is formed. It is obvious that RS_{com} vector length is less than *RS*-vector length. Difference of length of these vectors is denoted as ΔL.

Stage 3. DWM, which is a binary succession the length of which does not exceed ΔL is added to vector RS_{com} by concatenating. Vector *RS** obtained after concatenation is added with arbitrary binary values until it reaches the *RS* vector length. So:

$$RS^* = RS_{com} \,.\, DWM \,.\, Add, \tag{2}$$

where "." – operation of concatenation; *DWM* – watermark embedded into container; *Add* – optional addition of vector *RS** to *RS* vector length.

Stage 4. The movement about LUT units of container in the order determined by embedding path is performed. In the course of this movement *U*-units are ignored, and within *R*-units and *S*-units a mutual transition of one into another with the help of Flipping-function is made according to the table.

Table 1. The rules of LUT-unit move from one class to another

Class of the current LUT-unit LUT_i	Value of a bit, corresponding to the current LUT-unit LUT_i in vector *RS*	The current bit of vector *RS**	Action
R	0	0	—
R	0	1	Transformation LUT_i to *S*-unit
S	1	0	Transformation LUT_i to *R*-unit
S	1	1	—

The actions at the given stage leads to vector *RS** embedding in a LUT-container. Vector *RS** contains DWM and a compressed vector *RS* (vector RS_{com}) version in accordance with equation (2). Thus except for a DWM itself the information necessary for the original container value recovery is embedded in a container. The rules represented in table 1 describe the actions providing the vector *RS** embedding in a LUT-container. If the corresponding current bits of vectors *RS* and *RS** coincide then no modification for the current LUT unit is produced. In the case of mismatch of these bits the current LUT unit transition in the class corresponding to the current bit value of vector *RS** is performed.

Taking into account that the Flipping-function impact on LUT units occurs according to the unit inversion rules represented in [10], [11] LUT-container functioning does not change after DWM embedding.

3 Example of DWM Embedding According to the Proposed Technology

Let us consider the example of described stages of DWM embedding in container technology. In table 2 the hexadecimal values of codes of twenty LUT units ($i = 1…20$), which are in some embedding path of a container are shown in the lines "Unit code".

Table 2. Initial values of codes of LUT units lying in embedding path

i	1	2	3	4	5	6	7	8	9	10
Unit code	51F0	FF96	56EC	E8FE	2002	0008	8040	1000	0FF0	CCCD
n^1	7	12	9	11	2	1	2	1	8	9
Class of unit	*R*	*S*	*S*	*S*	*R*	*R*	*R*	*R*	*U*	*S*
RS-vector	0	1	1	1	0	0	0	0	—	1
i	11	12	13	14	15	16	17	18	19	20
Unit code	0220	88F8	F128	3F00	0001	9B94	AABA	A340	B0E1	EE00
n^1	2	7	7	6	1	8	9	5	7	6
Class of unit	*R*	*R*	*R*	*R*	*R*	*U*	*S*	*R*	*R*	*R*
RS- vector	0	0	0	0	0	—	1	0	0	0

For each of these units a number of one values (n^1) in the unit code bits are indicated. Depending on the proportion of amounts of one and zero values in the code the unit is classified as *R, S* or *U* unit according to the above mentioned rules. In accordance with the results of the classification a *RS*-vector is formed, in which zero values correspond to *R*-units and one values – to *S*-units. *U*-units do not participate in forming the *RS*-vector.

As a result the *RS*-vector takes the value *RS* = 011100001000001000 consisting of 18 bits (the information about two *U*-units was not included in *RS*-vector). Then the *RS*-vector is to be subjected to lossless compression. In the given example the simplest way of compression on the basis of Huffman method is used for illustration (in practice the more effective compression methods are to be used). For this purpose the triads of *RS*-vector bits are taken as elementary characters and their frequency of entering the vector is found. The triad "000" enters the vector twice, the triad "100" – once, the triad "011" – twice. As a result of Huffman method we obtain the following system of uneven prefix codes for the triads of *RS*-vector: "000" ⇒ "11", "100" ⇒ "100", "011" ⇒ "101", "001" ⇒ "0".

As a result of usage of the obtained codes we can have 12 bits vector RS_{com} = 101100011011 instead of the initial triads in 18 bits *RS*-vector. *RS* and RS_{com} vector lengths difference is $\Delta L = 6$. This value expresses the maximum amount of DWM bits, which can be embedded in LUT units in the given example.

In table 3 the values of bits of the initial *RS*-vector and its compressed version RS_{com} are shown. Let us consider the example of addition of the 6-bits DWM – *DWM* = 101010 to the vector RS_{com}. As a result of concatenation of the vector RS_{com} and *DWM* embedded in DWM container we obtain vector *RS** having the similar length as to vector *RS*.

Table 3. Binary vector values

RS	0	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0
RS_{com}	1	0	1	1	0	0	0	1	1	0	1	1						
DWM													1	0	1	0	1	0
*RS**	1	0	1	1	0	0	0	1	1	0	1	1	**1**	**0**	**1**	**0**	**1**	**0**

Then according to the rules specified in table 1 the LUT-container modification is performed. The aim of this modification is to adapt the LUT-unit classes lying in the embedding path to the value of vector *RS** containing DWM. As we see from table 1 the LUT-unit movement from one class to another is only performed when the values of the corresponding bits of vectors *RS* and *RS** do not coincide. The bits characterizing with the absence of coincidence of such kind are distinguished in table 3. For the LUT-unit codes corresponding to the distinguished bits a Flipping-function is applied and this leads to the movement of these units from one class to another.

In table 4 the values of LUT-unit codes after Flipping-function application to the distinguished units (units, for which the class is to be changed). As a result of these actions the *RS*-vector of units, which are in the embedding path takes a value of the vector *RS** and DWM is embedded into the LUT-container.

Table 4. Values of LUT-unit codes after embedding the DWM in the container

i	1	2	3	4	5	6	7	8	9	10
Unit code	AE0F	0069	56EC	E8FE	2002	0008	8040	EFFF	0FF0	CCCD
n^1	9	4	9	11	2	1	2	15	8	9
Class of unit	*S*	*R*	*S*	*S*	*R*	*R*	*R*	*S*	*U*	*S*
RS-vector	1	0	1	1	0	0	0	1	—	1
i	11	12	13	14	15	16	17	18	19	20
Unit code	0220	7707	0ED7	C0FF	0001	9B94	AABA	A340	4F1E	EE00
n^1	2	9	9	10	1	8	9	5	9	6
Class of unit	*R*	*S*	*S*	*S*	*R*	*U*	*S*	*R*	*S*	*R*
RS-vector	0	1	1	1	0	—	1	0	1	0

4 The Proposed Procedure of DWM Extraction from LUT-Container

Within the frame of the proposed technology the stego-key dedicated for data extraction consists of the following four components:

$$key = (order, classification, RSrule, \Delta L), \tag{3}$$

where *order* – the information defining the order of move around LUT units in the container (embedding path) for DWM embedding or extracting; *classification* – concrete definition of the principle of LUT-unit classification (unit is considered to be *R*-unit if the amount of parts in the unit code is more or less); *RSrule* – the rule of interpretation of *RS*, RS_{com}, *RS** vector bit values: for *R*-units a zero value and for *S*-units one value are fixed in the *RS*-vector and vice versa; ΔL – the difference between the vector *RS* length and its compressed version RS_{com} length.

The succession of actions of the proposed information technology in DWM extracting and recovering the initial value of LUT-container is as follows.

Stage 1. The movement about LUT-container units in the order specified by embedding path occurs. And along with this a binary vector *RS'* is formed on the basis of LUT-unit classification like in the case of embedding the information.

Stage 2. The obtained vector *RS'* structure corresponds to the one of vector *RS** (2) formed in DWM embedding in container. The last ΔL bits of vector *RS'* are DWM with the possible addition to the necessary vector length. At this stage the reading of this bits and DWM obtaining is performed.

Stage 3. The other vector bits are subjected to the procedure of decompression opposite to compression, which is performed at the stage of embedding the information. As a result a binary vector RS_{decom} is formed.

Stage 4. On the basis of information contained in vector RS_{decom} the recovery of original container value is performed. The movement about LUT-units of the container in the order specified by the embedding path and successive review of vector RS_{decom} values are performed for this. In the course of this movement the *U*-

units are ignored and within *R*-units and *S*-units a mutual transition is made by Flipping-function according to table 1.

For the example considered above DWM extraction can be described in the following way. As a result of analysis of LUT-unit codes lying in the embedding path the unit classification is performed and a binary vector *RS'* = 101100011011101010 is formed. From the vector end the $\Delta L = 6$ bits containing DWM "101010" are taking. The rest bits of the vector "101100011011" are subjected to decompression with the usage of the table making the match between the uneven prefix codes of compressed vector and the triads of decompressed vector bits. This leads to creation of binary vector RS_{decom} = 011100001000001000, which coincides with the container vector *RS* before DWM embedding in it. The given vector has the information for the original value recovery of LUT-container. According to the rules of table 1 the LUT-unit classes become the original ones and the container acquires the value it had before DWM embedding.

5 Experimental Research of the Proposed Technology

Aims of the experimental research:

1. to show experimentally that LUT-containers, in which DWMs were embedded with the help of the proposed technology acquire their original value after DWM extraction;
2. to indicate the degree of change of the main characteristics of containers after DWM embedding in them. The main characteristics are considered to be the following ones: a) maximal clock signals frequency expressing the limits of processing speed within the frames of one family of target chips; b) energy consumption of the input-output system of a chip; c) energy consumption of the chip core; d) thermal dissipation of a chip.

Environment for making the experiments. The hardware-software means based on the chips FPGA Altera Cyclone II and CAD Altera Quartus II usage have been developed for the experimental research. The group of scripts performing the interaction with CAD Altera Quartus II for reading and recording the LUT-unit contents was organized in TCL language. The read data processing subsystem itself is realized in language C# within the frame of the platform .Net in accordance with the proposed technology. The estimation of design characteristics mentioned above were carried out by means of CAD Altera Quartus II *Timing Analyzer* (estimation of the limit processing speed) and *Power Play* (estimation of energy consumption and thermal dissipation). The states of container before DWM embedding and after DWM extraction were compared by bit-by-bit analysis of configuration files of the corresponding containers.

The Materials for experiments are 40 FPGA – projects of different complexities and purposes. The hardware complexity of devices within the frames of the given projects varies from 1,2% to 65% resource volume (logic cells, RAM units) of target chip FPGA.

The process of making the experiments. The experiments for each of the researched project consist of the following stages:

1. estimation of the main characteristics (processing speed, energy consumption, thermal dissipation) for the device represented in the form of initial container;
2. embedding the random binary succession of DWM in container;
3. estimation of the main characteristics for a container having the embedded DWM;
4. DWM extraction from the filled container;
5. comparison of configuration files of the final and initial container states.

The results of experimental research. As a result of experiments a complete coincidence of configuration files of the initial containers and the ones obtained after DWM extraction for all of the 40 projects has been established. So due to the features of self-recovery after DWM extraction from containers they take the original form and their characteristics return to the initial values.

In the part of determining the degree of changes of the main container characteristics (after DWM embedding) the following results were obtained. We found that insignificant changes of container characteristics occurs as a result of DWM embedding. Along with this the dependence of this change upon hardware complexity and its structural peculiarities is extremely low. The difference between the changes of characteristics for devices occupying 1,2% and the ones occupying 65% resource volume of the target chip FPGA is hundredths of a percent. On average (for all of the researched 40 projects) the processing speed characteristic change is 0,18%, changes of energy consumption and thermal dissipation characteristics – 0,22%. As we see the change values are within the limits of error value of measurement means.

The technology offered in the paper does not change the mutual connections of LUT units but does change the values of the specific units codes. It is accordingly of interest to learn whether the changes of units codes are able to influence on the FPGA project features under such conditions. Except for the previously considered experiment another experimental research of the influence of the values of LUT units codes on the basic features of FPGA projects has been made with the participation of one of the authors of the given paper. The description of this research and its results are shown in [14]. In the above considered experiment a random binary succession was embedded in various FPGA projects. This led to the fragmentary changes of LUT units codes as in accordance with table 1 the codes of units change only in the two of four cases of possible combinations of values of bits of vectors RS and RS*. In [14] the extreme cases of codes change in all the involved LUT units of FPGA-project were considered. In order to create such kinds of changes the projects of similar structures, but with the LUT units codes containing minimum and maximum amount of one value, respectively, were formed and compared. The projects were chosen to deploy the resources of the corresponding FPGA chip in a maximum way. In table 5 the main results of comparison of such projects are represented. The projects with the minimum and maximum amount of one values are marked in table 5 as P_{min} and P_{max}, respectively. In the given table the following characteristics are represented: maximum clock frequency F; dynamic and static power supply as well as the total

energy consumption of chip core obtained for the indicated clock frequencies; δ – a relative energy consumption change in mass changes of the number of logical values in the codes of LUT units of a project. All the families of chips used in the experiment have the core power supply equal 1,2 V.

Table 5. The results of energy consumption and processing speed estimation

Project	Total core energy consumption (mA)	Dynamic power supply (mA)	Static power supply (mA)
Cyclone II EP2C35F672C6 F = 420,18 MHz			
P_{min}	**699, 17**	629,98	69,19
P_{max}	**701,40**	632,19	69,22
δ	**0,31 %**	0,34 %	0,04 %
Cyclone III EP3C40F780C6 F = 516,26 MHz			
P_{min}	**495,6**	491,49	4,11
P_{max}	**497,37**	493,39	3,98
δ	**0,35 %**	0,38 %	3,16 %
Cyclone III LS EP3CLS70F780C7 F = 516,26 MHz			
P_{min}	**494,6**	482,91	11,69
P_{max}	**496,23**	484,73	11,51
δ	**0,32 %**	0,37%	1,5 %
Cyclone IV E EP4CE30F29C6 F = 600,6 MHz			
P_{min}	**477,42**	470.42	7,00
P_{max}	**479,64**	472,22	7,41
δ	**0,46 %**	0,38 %	5,5 %

The experiment results shown in table 5 demonstrate that in maximum mass change of the number of one values correlation in LUT units:

1. the energy consumption of input output system for chips within a FPGA family is retained at the same level;
2. the chip core energy consumption changes insignificantly within one family mainly due to the dynamic component (maximum value of this change equals 0,46% for chips EP4CE30F29C6 of family Cyclone IV E);
3. the maximum clock frequency expressing the extreme processing speed of devices does not practically change within one FPGA family.

The results show that such important characteristics as productivity and energy consumption are not practically dependent upon the type of Software code used in FPGA-projects with the fixed hardware realization. So we can come to the conclusions that according to the technology proposed in the given paper the DWM embedding in a LUT-container does not substantially impact on the FPGA project characteristics mentioned above.

6 Conclusions

The information technology proposed in the paper allows to perform DWM embedding in containers with LUT-oriented architecture. The technology gives the possibility to recover the initial state of container after DWM extraction from it. The technology is based on the combination of DWM embedding method into LUT-containers proposed in the given paper and the popular Fridrich method oriented at the passive multimedia containers. Unlike Fridrich method the proposed technology:

- uses less complicated Flipping-function taking into account the container peculiarities;
- does not include the discrimination function calculation in order to perform the element classification of container;
- performs information embedding at the level of the container elementary parts (LUT-units) but not at the level of groups of elementary parts.

The degree of the proposed technology effectiveness is of qualitative nature and is expressed in possibility to recover the initial LUT-container after DWM extraction, which was absent before. The proposed technology can be used in developing the hardware and software, which realize digital watermark embedding in computer and control devices created on the LUT-oriented element base (e.g. FPGA or programmable logic integrated circuits with the similar architecture). Such kind of embedding allows to control the device configuration integrity makes the substitution or corruption of configuration impossible, which is of vital importance for critical domains. In addition it allows to control the legitimacy of usage of project information and devices themselves at the different stages of CAD and life cycle: synthesized FPGA project, configuration file FPGA, operating device.

References

1. Shih, F.: Multimedia Security: Watermarking, Steganography, and Forensics. CRC Press, Boca Raton, FL (2013)
2. Cox, I., Miller, M., Bloom, J., Fridrich, J.: Digital Watermarking and Steganography. Morgan Kaufmann Publishers, Amsterdam (2008)
3. Vasu, S., George, S., Deepthi, P.: An Integrity Verification System for Images Using Hashing and Watermarking. Proceedings of the International Conference on Communication Systems and Network Technologies. 85–89 (2012)
4. Coatrieux, G., Hui Huang, Huazhong Shu, Limin Luo, Roux, C.: A Watermarking-Based Medical Image Integrity Control System and an Image Moment Signature for Tampering Characterization. IEEE J. Biomed. Health Inform. 17, 1057–1067 (2013)
5. Sencar, H., Memon, N.: Watermarking and ownership problem. Proceedings of the 5th ACM workshop on Digital rights management – DRM'05. (2005).
6. Arnold, M., Schmucker, M., Wolthusen, S.: Techniques and Applications of Digital Watermarking and Content Protection. Artech House, Boston (2003)
7. Anderson, R.: Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition. Wiley, New York (2008)
8. Huffmire, T.: Handbook of FPGA Design Security. Springer, Dordrecht (2010)

9. Paul, S., Bhunia, S.: Reconfigurable Computing Using Content Addressable Memory for Improved Performance and Resource Usage. In: Proc. Design Automation Conference ACM/IEEE (DAC-2008), 786–791, ACM, Anaheim (2008)
10. Zashcholkin, K., Ivanova, E.: Method of Steganographical Hiding of Information in LUT-Oriented Hardware Containers. Electrical and Computer Systems. 12(88), 83–90 (2013) (in Russian)
11. Zashcholkin, K., Ivanova, E.: Steganography Data Hiding in LUT-Oriented Hardware Containers Method Development. Electrical and Computer Systems. 13(89), 231–239 (2014) (in Russian)
12. Fridrich, J., Goljan, M., Du, R.: Lossless Data Embedding – New Paradigm in Digital Watermarking. EURASIP Journal on Adv. Signal Process. 2002, 185–196 (2002)
13. Goljan, M., Fridrich, J., Du, R.: Distortion-Free Data Embedding for Images. In: Proc. of the 4th International Workshop on Information Hiding (IHW-01), 27–41, USA, Pittsburg (2001)
14. Zashcholkin, K., Kuznetsov, N., Drozd, A.: Research in Key Features of FPGA-projects in Case of Changing the Codes in LUT-blocks. Scientific Herald of Yuriy Fedkovych Chernivtsi National University: Computer Systems and Components. Vol. 5, Issue 2, 82–86 (2014) (in Russian)