Increasing the Effective Volume of Digital Watermark Used in Monitoring the Program Code Integrity of FPGA-Based Systems

Kostiantyn Zashcholkin
Department of Computer Intelligent Systems and Networks
Odessa National Polytechnic University
Odessa, Ukraine
const-z@te.net.ua

Oleksandr Drozd
Department of Computer Intelligent Systems and Networks
Odessa National Polytechnic University
Odessa, Ukraine
drozd@ukr.net

Ruslan Shaporin
Department of Computer Intelligent Systems and Networks
Odessa National Polytechnic University
Odessa, Ukraine
rshaporin@gmail.com

Olena Ivanova
Department of Computer Systems
Odessa National Polytechnic University
Odessa, Ukraine
en.ivanova.ua@gmail.com

Yulian Sulima
Computer Systems Department
Odessa Technical College of the Odessa National Academy of Food Technologies
Odessa, Ukraine
mr_lemur@ukr.net

I. Introduction

At the moment a considerable share of hardware of the computer and control digital systems is based on programmable devices. One of the main reasons of preference of the very programmable devices is that there is the possibility to modify their operation during all the life cycle. Due to this possibility a number of typical tasks, which can appear at the different stages of life cycle of the computer or control system, is solved simply enough (as compared to nonprogrammable devices). We can refer to such kinds of tasks the following ones: a) functioning faults elimination detected in the course of the device operation; b) expansion and changing the set of functions, which are provided by the device; c) functioning optimization of the device.

However the possibility to modify the program code of programmable devices generates the problem of provision of this program code integrity. The potential accessibility to the program code rewriting function is the basis for vulnerability, which allows to illegitimately bring modification to the program code. The presence of legitimate program code modification in the process of the device operation permits to mask a malicious modification presenting it as a part of a legitimate one. The program code integrity violation of devices, entering the composition of systems of both safety-critical and mass usages, creates the excessive risk with unacceptable consequences. So the safety of systems, in which the programmable devices are included, cannot be ensured without the solution of problem of the program code integrity provision.

In the given paper a problem of the program code integrity provision of one of the widely used classes of programmable devices – FPGA chips (Field Programmable Gate Array) is considered. The FPGA chips are a set of programmable basic calculating units, the links between which are ensured by the programmable system of commutation. The natural parallelism of the computing tasks solution with the help of FPGA chips creates their (chips) advantage in performance characteristics as compared to microprocessors.

In spite of the presence of embedded mechanisms of the program code protection from rewriting in many FPGA-based systems there are the bypass ways of such kind of protection allowing to enter the illegitimate modification in the program code. By virtue of this the most popular approaches to the provision of the program code integrity of FPGA-based components is a combination of processes of access restriction to the program code and integrity monitoring. The integrity monitoring is traditionally based on the usage of extra monitoring data units allowing to make the conclusions about the code integrity.

II. Literature review and goal of the paper

The most popular approaches to the program code integrity monitoring used in practices have become the ones, which use a hash sum. For the program code information object a hash sum is calculated with the help of the set hash function. This hash sum is considered further to be a standard one. A standard hash sum is in some way matched with the program code information object or joins it. Further if checking the program code integrity is to be executed the recalculation of information object hash sum is implemented. The comparison of the standard and newly calculated hash sum permits to confirm the integrity or detect its violation.

One of the substantial constituents of the integrity monitoring efficiency (in the point of counteraction to the attempts to bypass monitoring) is a way and location of the standard hash sum storage. In using the traditional approaches to the integrity monitoring the following ways (or their variations) of storing the standard hash sum are applied.

1) A standard hash sum is stored separately from the program code information object in some centralized database. The main disadvantage of this way of storing is the complexity of the database protection from information leakage. The mass leakage of information from database with hash sum (which is a quite frequent event as the practice shows) compromises all

the systems of integrity monitoring, which this database provides. Even under the conditions of extra encryption of the standard hash sum the access to its encrypted values creates a potential chance of spoofing and bypassing the integrity monitoring.

2) The standard hash sum is stored together with the program code information object in the FPGA configuration memory. The disadvantage of this way is conditioned, firstly, with the evidence for an outside surveillance that the integrity monitoring of the given information object is carried out, and secondly, that the standard hash sum is accessible and this makes the attempts to spoof it easier.

3) The standard hash sum is included in the program code information object and stored as its constituent. The hash sum detection inside the information object is not of great difficulty because the hash sum is not distributed about the information object but is centrally stored in its structure. By virtue of this the given way has the disadvantages similar to the previous one.

Thus the described ways of storing a standard hash sum potentially create the vulnerability, which can become a cause to attempt to spoof the hash sum with the aim to hide the integrity violation.

A perspective approach to integrity monitoring is the standard hash sum embedding into the program code information object in the form of a digital watermark. Such kind of approach masks from an outside surveillance the very fact of the integrity monitoring implementation. The digital watermark imbedding does not change the size of the program code information object. Moreover as a result of the digital watermark embedding the operation of programmable device, which functioning is set by the program code, is not modified. These features of the digital watermark are the results of usage of the special equivalent conversion with respect to program code elements. For embedding the digital watermark into FPGA an equivalent conversion of program code of the series-connected LUT (Look Up Table) basic calculating units is used. Wherein the system of links between LUT units as well as the operation, energy characteristics and performance of the device are not changed.

The digital watermark extraction from the FPGA program code is possible if steganographic key is available. The key determines the rules of the digital watermark bits placement in the LUT unit set.

The peculiarity of integral monitoring, which is carried out with the help of the digital watermark, is the necessity to recover an initial state of the program code information object. At the moment of monitoring execution the digital watermark (containing hash sum) is to be extracted from information object, and the information object itself is to be recovered in the state, in which it existed prior to embedding the digital watermark (initial state). Such recovery is necessary because the standard hash sum is calculated for the initial state of information object. Embedding the digital watermark changes this state.

To ensure the initial state recovery of the program code information object a compression-based approach is used. The bit values $M = \langle m_1, m_2, \dots, m_n \rangle$ (values of the specify bits of the LUT units program code) of information object, which are along the embedding path of digital watermark, are combined in a bit sequence. This bit sequence is subjected to the lossless compression procedure. The compressed bit sequence M_{com} together with service data S (which contains the fields length of the digital watermark) and the standard hash sum creates the digital watermark DWM. This digital watermark is embedded into the place of the bit sequence M by the equivalent conversion. Thus the standard hash sum size in the digital watermark cannot exceed value (1).

$$L_{Hash} = L_M - L_{Mcom} - L_S \qquad (1)$$

The size of service data S field is fixed. On this basis the size L_{Hash} is dependent on: the number of LUT units (the sequence M length), in which the digital watermark embedding is executed; the applied compression method; the content of the bit sequence M. Wherein the value L_{Hash} can be learnt after indicating the location of watermark embedding and bit sequence M compression.

In case if the amount of bits necessary for storing the standard hash sum exceeds the value L_{Hash} a situation arises when the information object cannot be finally prepared for integrity monitoring. In this case a hash function, which gives a hash sum with less number of bits, should be chosen. If such kind of hash-function change is inaccessible in accordance with the monitoring conditions one should give up monitoring with the help of the digital watermark. On this basis we can constant the following issues: a) field size limitation of the monitoring digital watermark, which is dedicated for storing the standard hash sum; b) instability of this size and possibility to learn it only at the final stages of the information object preparation for integrity monitoring.

The goal of the given paper is to increase the effective volume (which is intended for storing the monitoring hash sum) of the digital watermark as compared to the integrity monitoring methods using compression for the recovery of information object state.

III. The integrity monitoring method providing the increased effective volume of the digital watermark

We offer a method of the FPGA program code integrity monitoring, which allows like all compression-based methods:

- to save the initial state of the FPGA program code information object (at the stage of preparing the information object for monitoring);
- to execute the initial state recovery of the FPGA program code information object and the digital watermark extraction simultaneously (at the stage of integrity monitoring implementation).

However wherein the proposed method allows providing the larger effective volume of the digital watermark (the volume intended for the hash sum monitoring storage) than the one (volume) provided by compression-based methods.

That is why the initial state recovery by means of the preliminary preparation of the FPGA program code information object is offered. The proposed method uses some Wong's method ideas as a base. According to the method offered by Wong a fragile digital watermark is embedded into a bitmap image. The property of the digital watermark fragility in the method by Wong makes possible the image integrity monitoring. This method also requires to bring some bits in the values of image pixels to the predetermined state.

The proposed method in the given paper differs from the one by Wong in the following aspects.

The proposed method is oriented to the digital watermark embedding into the FPGA program code and permits only the equivalent conversion of basic units values of the information object. But the method by Wong is oriented to the digital watermark embedding into a multimedia information object and allows the distortion of the basic unit values of this object.

The method by Wong requires to mandatorily bring all basic unit values of the information object to predetermined state. The method offered in the given paper requires to bring only the basic unit values, which are along the embedding path of the digital watermark, to the predetermined state.

The method by Wong fixes the least significant bits as target embedding bits (this is conditioned by the peculiarity of multimedia information objects the method by Wong is oriented to). The proposed method gives the possibility to use equally any of the bits of the basic units (LUT units) program code of information object.

The method by Wong indicates only a single rule how to bring the target bits to the predetermined state – their setting in value 0. The proposed method allows to use any determinate rules (described in the corresponding steganographic key component) to bring the target bits to the predetermined state.

To formulate the principles of the proposed method the following notations and definitions are introduced.

Let $L = \{LUT_1, LUT_2, \dots , LUT_p\}$ is a set of LUT units of FPGA-based device, in the program code of which the monitoring digital watermark is embedded.

On the basis of the rules indicated by steganographic key an ordered set of LUT units, which are along the embedding path of the digital watermark $EmbPath = <l_1, l_2, \dots , l_n>$, is formed from this set. In the course of embedding the digital watermark bits are directly embedded into the program codes of the *EmbPath* LUT units.

Each of the units $l_i \in EmbPath$, where i=1…n contains k-bit program code P_i, respectively. In each of the program codes P_i one of the bits d_i of monitoring digital watermark can be embedded with the help of equivalent conversion.

To each of the units $l_i \in EmbPath$ (which is along the embedding path) corresponds one bit $m_i \in P_i$. This bit of program code P_i can be used for embedding one bit of the digital watermark. The correspondence between $l_i \in EmbPath$ and $m_i \in P_i$ is set by rules described in steganographic key. Below the bits m_i will be called the *target bits of embedding*.

The basic theoretical principles of the proposed method are as follows.

The first principle of the method: the initial state recovery of FPGA program code information object is provided on account of the preliminary preparation of this information object. The preparation is carried out prior to embedding the digital watermark into information object. The preparation lies in bringing the target bits of embedding to some predetermined state. The state, which these bit values are brought to, is indicated by rules including in the steganographic key structure.

The second principle of the proposed method: bringing the target bits to the predetermined state (set by steganographic key) is performed with the help of the equivalent conversion similar to those, which are used for the digital watermark embedding.

The third principle of the proposed method: the digital watermark within the framework of the proposed method contains only the monitoring hash sum. There is no information for initial state recovery of information object in it. The lack of necessity to save this information is conditioned by the fact that the initial state of information object is recovered according to the rules described in steganographic key.

The fourth principle of the proposed method: steganographic key (which is used in embedding and extracting the digital watermark) contains the rules for bringing the target bits to the predetermined state. These rules regulate both values themselves (fixed or changed according to some law) and their location in the space of FPGA program codes of LUT units.

To provide this principle a component, which describes the rules of bringing the target bits to the predetermined state, is offered to include in steganographic key:

$$PD\text{-}rule = <value, location>,$$

where $value \in \{$*fixed-value, value-pattern, random-value-rule*$\}$; $location \in \{$*fixed-location, location-pattern, iteration-location-rule, random-location-rule*$\}$.

Component *PD-rule* consists of two elements: element *value* indicates a rule of the target bits value formation in the course of their bringing to the predetermined state; element *location* sets the target bits location in the space of LUT units program codes.

Element *value* determines three possible ways of the target bits value formation: *fixed-value* is fixed value 0 or 1 for all the target bits; *value-pattern* is the values, the changes of which are described by some regular pattern; *random-value-rule* is the values, the changes of which are set by a rule based on pseudo-random number sequence.

Element *location* determines four possible ways of specifying the target bits location: *fixed-location* is the location in bits, which have one and the same number in all LUT units program codes; the rest three ways set the location in bits, number of which changes from unit to unit in accordance with some rule; *location-pattern* is a regular pattern of the bit number change; *iteration-location-rule* is an iteration rule of the bit number change; *random-location-rule* is a rule of the bit number change based on the pseudo-random sequence.

The proposed method is a sequence of stages which are performed in preparing the FPGA program code information object for integrity monitoring, as well as the ones, which are executed in the course of the monitoring itself.

The preparations of FPGA code information object for integrity monitoring.

Stage 1. According to the rules included in the steganographic key components the units, which are along the embedding path, are chosen from the set of *LUT* units. These units create the ordered sequence $EmbPath = <l_1, l_2, \dots , l_n>$.

Stage 2. In accordance with the steganographic key component *location* $\in$ *PD-rule* the ordered sequence of target bits $M = <m_1, m_2, \dots , m_n>$ is formed wherein each target m_i is a bit of the LUT unit l_i program code.

Stage 3. In accordance with the steganographic key component *value* $\in$ *PD-rule* the binary values sequence $A = <a_1, a_2, \dots , a_n>$ is formed. These values are considered to be the initial ones for target bits of the digital watermark embedding.

Stage 4. With the help of the equivalent conversions the target bits $m_i \in M$ replacement with the initial values $a_i \in A$ is performed. After this FPGA program code information object is considered to be brought to the initial predetermined state.

Stage 5. For the program code information object a monitoring hash sum is calculated. This hash sum is calculated with the help of a hash function, set by steganographic key.

Stage 6. The obtained hash sum is embedded into the target bits of the FPGA program code information object in the form of digital watermark. The embedding is performed according to the traditional methods of the digital watermark embedding into FPGA program code.

The executions of information object integrity monitoring.

Stage 1. In accordance with the rules determined by the steganographic key components the units, which are along the embedding path, are chosen from the LUT units set.

Stage 2. According to the steganographic key component *location* $\in$ *PD-rule* an ordered sequence of target bits is formed. At the stage of information object preparation the digital watermark is embedded into these bits.

Stage 3. The digital watermark, which contains the monitoring hash sum, is extracted from these bits.

Stage 4. An action analogous to the one, which is performed at *Stage 3* in preparing the information object for monitoring, is carried out: in accordance with the steganographic key component *value* $\in$ *PD-rule* the binary values sequence $A = <a_1, a_2, \dots , a_n>$ is formed.

Stage 5. The initial state recovery of the program code information object is performed. To do this the target bits $m_i \in M$ replacement with the initial values $a_i \in A$ is implemented with the help of the equivalent conversion.

Stage 6. For the program code information object obtained at *Stage 5* the monitoring hash sum is calculated. This hash sum is calculated with the help of a hash function, set by steganographic key.

Stage 7. The comparison of hash sum extracted from the information object at *Stage 3* and the one calculated at *Stage 6* is performed. If these hash sums coincide the information object integrity is considered to be confirmed. Otherwise the integrity violation is fixed.

IV. THE PROPOSED METHOD AND EXPERIMENT DISCUSSION

The proposed method efficiency in the point of effective volume increase of the digital watermark is as follows. The traditional methods of integrity monitoring (which are based on the digital watermark usage) apply the compression to save the initial information object state. These methods permit to use only a small part (1) of the digital watermark volume for storing the monitoring hash sum. This reason does not allow for in some cases the traditional methods to provide the saving of hash sum with a size necessary for monitoring. As to the method proposed in the presented paper it gives the possibility to use all the available volume of the digital watermark for storing the hash sum.

To compare the offered method to the traditional ones an experiment was made. The experiment was made for five FPGA projects of different volume. The synthesis of these projects was implemented in CAD environment Intel (Altera) Quartus for target FPGA chips Intel Cyclone IV.

For all the five projects the embedding path formation was performed with the help of one and the same steganographic key. Then the authors indicated what size of the hash sum is which can be provided by a traditional compression-based method of integrity monitoring. The sequence of target bits was formed with further performing the compression of this sequence.

Then according to equation (1) the maximal possible size of hash sum was calculated. We also indicated which of the most popular hash functions can provide a hash sum that could fit to this possible size.

One can see that the traditional method does not give the possibility to save a suitable size hash sum (obtained with the help of some high-usage hash function) for projects 1 and 2. This is connected with relatively small total amount of LUT units in these projects. As a result we have small amount of LUT units placed along the embedding path (the total size of the digit watermark) and, consequently, too small length of the hash sum field.

Projects 3 and 4 have more amount of LUT units than the ones 1 and 2. However in applying the traditional method the hash sum field size for these projects permits to use only a hash sum obtained with the help of hash function MD5 (the size is 128 bits).

For project 5 the traditional method gives the possibility to use a hash sum obtained with the help of both hash function MD5 (the size is 128 bits) and hash function SHA1 (the size is 160 bits).

The method offered in the given paper provide the effective volume (for the hash sum storage), which equal to the amount of LUT units placed along the embedding path. Thereby the proposed method allows saving a hash sum in the digital watermark for all projects, which participate in the experiment (Table 1). For those projects, for which the traditional method provides the minimum possible size of the hash sum field, the proposed method permits to use a hash sum of the larger size.

V. Conclusions and directions of the further research

A method of FPGA program code integrity monitoring based on the digital watermark usage is offered in the paper. The method is different from the similar ones existing in the literature with the fact that it does not apply the compression to save and recover the initial information object state at the stage of monitoring. To provide the recovery of initial information object state within the framework of the proposed method the preliminary bringing of information object to a predetermined state, set by steganographic key, is carried out. For performing the integrity monitoring after extracting the digital watermark the repeated bringing of information object to the specified predetermined state is carried out.

The experimental research of the proposed method has shown its efficiency (as compared to the traditional methods) in the point of provision of the effective digital watermark volume sufficient for saving the hash sums obtained with the help of the most widely used hash functions.

We assume that perhaps the usage of the proposed method reduces (as compared to the traditional compression-based methods) the program code information object resistance to stegoanalysis. We assume that perhaps the usage of the proposed method reduces (as compared to the traditional compression-based methods) the program code information object resistance to stegoanalysis. Here we mean only the process of detection of the digital watermark presence in FPGA program code. But the question if the information object resistance to stegoanalysis becomes less (and if it decreases then to what extension) requires extra research. If as a result of this research we come to the conclusions that the resistance really reduces then the technique of the following compromise variant choice is to be created: what is more important – to use a hash sum with more amount of bits (with larger cryptographic secure) or to decrease the probability of detection of the digital watermark in the program code.