DEVELOPMENT OF PROTECTING A SOFTWARE PRODUCT MATHEMATICAL MODEL FROM UNLICENSED COPYING BASED ON THE GERT METHOD

The subject of the research is methods and algorithms for developing the GERT networks that provide a hidden transition along the branches of the control logic graph and the ability to encode digital watermarks in the context of existing threats to the licensing security of software systems. The aim of the work is to develop a system of a software product licensed security based on hidden transition algorithms in GERT networks that carry the properties of digital watermarks. The article solves the following goal: development of a model of a software licensed security system based on the developed algorithms using GERT networks. Methods of mathematical modeling, numerical experiment, complexity theory and cryptography are used. The following results were obtained: on the basis of the analysis of existing models for ensuring software licensed security, the main requirements for the algorithms synthesized within the framework of the developed model, as well as for the machine implementation of the algorithm, were identified. Based on the investigation of existing watermark systems, methods of attack on them, as well as the requirements put forward for the developed model, a licensing security algorithm was formed based on the watermark system. Conclusions: for the first time, algorithms for safe transition in GERT-networks were developed, which are used as a graph of the control logic of a software product. This logic is implemented depending on the identification or serial number; a model of a licensed security system has been developed, which has not only empirical, but also theoretical substantiation of resistance to attacks by an intruder.

Keywords: software security, GERT model, license security.

Introduction

Formulation of the problem. The software licensing security system is a special component of a software or hardware/software system, with the help of which copyright protection for software is carried out as an object of intellectual property.

Thus, prevention of illegal software product, as well as, if necessary, proof of the author's ownership of this intellectual object.

One of the promising means of protecting this kind of software is protection using digital watermarks [1–3]. This allows you to prevent theft of intellectual property, and, if this happens, to prove your rights to its ownership. Currently, there are a number of approaches that allow in practice to embed digital watermarks in software [4–5]. Large IT companies such as StarForce Protection Systems use digital watermarks as the backbone of their commercial copy protection software.

Analysis of recent research and publication. The task of implementing digital watermarks in software can be formulated as follows. It is necessary to embed the data structure S in the program P so that S can be found in P and extracted from it, even if P has undergone some modification (transformation, optimization, packing, etc.). In this case:
- the structure S must have a mathematical property that makes it possible to assert that its presence in P is the result of deliberate actions.
- the structure S can be large and must be hidden in P;
- a program P with an already built-in structure S should not lose performance;

© Semenov S., Zhang Lijiang, Cao Weiling, Davydov V., 2021
struct, in the order of independent expressions, in the order of instructions for working with the stack (push and pop), and also in the program control logic graph.

It should be noted that it is very difficult to make static digital watermarks more persistent with the help of any transformations that increase stability or hide. So, for example, the study [7] describes a method according to which not just a digital watermark is embedded into the image (which is placed in the static data section of the program), but a fragment of the program's executable code. During program execution, this fragment is retrieved and executed. However, executing the code "on the fly" unmasks the digital watermarks and allows the enemy to localize it.

In [9], the concept of using a digital watermark for protecting software from illegal use was formulated and algorithms were proposed that use intermediate bytecode generated by a Java compiler for the JVM virtual machine as a steganographic container.

The foundations of the systems of digital watermarking on dynamic graphs were studied in [10–12]. The main idea is to embed the digital watermark into the graph topology. Thus, the vertices of the graph are linked by pointers, which makes the analysis of this data structure somewhat difficult.

The aim of the paper is the development of algorithms for the safe transition and coding of digital watermarks in GERT networks, resistant to the existing threats to the licensed security of software products.

The main task includes the development of a model of a software licensed security system based on the constructed algorithms using GERT networks.

Statement of basic materials

Requirements for the considered model. The considered licensing security model is based on the theory of complexity. Traditionally, problems of complexity theory are used to synthesize cryptographic algorithms [10], however, the peculiarity of the proposed model is to use the main problems of complexity theory to protect software from illegal use. At the same time, the licensed security system designed within the framework of the model will meet the requirements of the set construction task.

Consider the basic requirements of the model for the algorithms synthesized within its framework:

1. Availability of data coding mechanisms by means of the graph topology of the control logic of the algorithm (allows embedding digital watermarks into the graph).

2. Availability of mechanisms for increasing the complexity of the control logic graph (allows you to design various software security algorithms).

3. Theoretically substantiated undecidability of specific mathematical problems on the graph and the presence of the properties of a digital watermark in the graph (allows us to speak about the elements of the theoretical stability of the created system).

Within the framework of this model, requirements are also imposed on the machine implementation of the algorithm, which must:

1. Be based on dynamic data structures (elements of empirical robustness).

2. Use a multithreaded implementation of the control logic graph (additional elements of empirical robustness).

The advancement of these requirements for the software implementation of the algorithm is due to the empirical stability of software systems protection tools that use dynamic data structures and several streams at once.

The empirical stability of such systems is based on the fact that it is much more difficult for a researcher to analyze the code of a program that allocates its data in a dynamic memory area, and also has several threads working with this data. This is due, firstly, to the "fragility" of dynamic data. Those, if a person analyzing the program does not have time to track operations with data "in the heap" (heap areas), then as the program is executed, this data will be destroyed (other data will be written in their place). Secondly, the analysis of the code of a program using several threads, each of which works with dynamic data, requires constant switching of the execution context of the process (thread) and analysis of both the executable code and the data area. From an empirical point of view, it is much easier to analyze a program that has one (main) thread and uses only static data types.

Fulfillment of these requirements allows us to ensure that the algorithms built within the framework of the model will be able not only to protect software using active methods (secure verification of serial numbers), but also to prove the rights to own the system as an object of intellectual property:

− the algorithms synthesized with the help of the model must have mechanisms for coding data by means of the topology of the graph of the control logic of the algorithm. This allows long integers to be embedded in the structure (topology) of the graph;

− according to the above model, the designed software licensing security system must have mechanisms for increasing the complexity of the control logic graph. This, first of all, creates an empirical justification for robustness, since a branched and long control flow graph is much more difficult to analyze in practice. Note that the computing resources available today on standard personal computers make it possible to put up with the specially overestimated complexity of the control logic graph;

− the model requires the licensing security system to use the theoretically substantiated insolubility (within a reasonable time) of specific mathematical
problems on the graph and the presence of the properties of a digital watermark in the graph. This requirement is key within the framework of the presented model, since it allows us to speak about the theoretical stability of the developed licensing security system, while the rest of the requirements relate only to empirical stability and passive copyright protection (using digital watermarks).

It is important to note that the licensed security system designed within the framework of the model does not include many infrastructural elements. Those do not address issues such as obtaining serial numbers, automating means of encoding unique identifiers into a control logic graph, automatically increasing the complexity of this graph, etc.

**Development of the licensed security algorithm model.** Based on the investigated watermark systems, methods of attack on them, as well as the requirements put forward for the developed model, a licensing security algorithm was formed based on the watermark system. This algorithm is shown in Fig. 1.

According to the presented algorithm, the licensing security process can be described as follows.

- **Step 1.** The input sequence (bits) of the license key is fed to the system input.

- **Step 2, Step 4.** Some of the bits set to "1" are selected and the action necessary to continue the system operation is performed for them. In the simplest case - the formation of a mutex, in which information is written that a certain block of bits is successful. In the advanced case, the execution of a certain code, without which the further operation of the system is impossible. The difference between Step 2 and Step 4 is the correspondence of the checked bits and the number of called code blocks.

- **Step 6.** Some of the bits set to "0" are selected and the action necessary to continue the system operation is performed for them.

- **Step 3, 5, 7.** Checking the correctness of the conditions of the developed procedures.

According to the presented materials, a GERT network of the software licensing security process was developed, shown in Fig. 2. A distinctive feature of this model is the presence of States 5, 8 and 9. The transition to these states occurs only when the system was in all the previous states of the previous step.

It should be noted that the considered model in Fig. 2 is a software licensing process using a 4-bit license key.
Table 1

<table>
<thead>
<tr>
<th>№</th>
<th>branch</th>
<th>( W )-function</th>
<th>Transition probability</th>
<th>Probability Density Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>(0, 1)</td>
<td>( W_{01} )</td>
<td>( P_1 )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>2.</td>
<td>(0, 2)</td>
<td>( W_{02} )</td>
<td>( P_2 )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>3.</td>
<td>(0, 3)</td>
<td>( W_{03} )</td>
<td>( P_3 )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>4.</td>
<td>(0, 4)</td>
<td>( W_{04} )</td>
<td>( P_4=1-P_1-P_2-P_3 )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>5.</td>
<td>(1, 5)</td>
<td>( W_{15} )</td>
<td>( P_5 )</td>
<td>( k=k_2; \ 0=0_2 )</td>
</tr>
<tr>
<td>6.</td>
<td>(2, 5)</td>
<td>( W_{25} )</td>
<td>( P_6 )</td>
<td>( k=k_2; \ 0=0_2 )</td>
</tr>
<tr>
<td>7.</td>
<td>(5, 9)</td>
<td>( W_{59} )</td>
<td>( P_7 )</td>
<td>( k=k_3; \ 0=0_3 )</td>
</tr>
<tr>
<td>8.</td>
<td>(5, 0)</td>
<td>( W_{50} )</td>
<td>( P_8=1-P_7 )</td>
<td>( k=k_4; \ 0=0_4 )</td>
</tr>
<tr>
<td>9.</td>
<td>(3, 9)</td>
<td>( W_{39} )</td>
<td>( P_9 )</td>
<td>( k=k_5; \ 0=0_3 )</td>
</tr>
<tr>
<td>10.</td>
<td>(3, 0)</td>
<td>( W_{30} )</td>
<td>( P_{10}=1-P_9 )</td>
<td>( k=k_4; \ 0=0_4 )</td>
</tr>
<tr>
<td>11.</td>
<td>(4, 6)</td>
<td>( W_{46} )</td>
<td>( P_{11} )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>12.</td>
<td>(4, 7)</td>
<td>( W_{47} )</td>
<td>( P_{12}=1-P_{11} )</td>
<td>( k=k_1; \ 0=0_1 )</td>
</tr>
<tr>
<td>13.</td>
<td>(6, 8)</td>
<td>( W_{68} )</td>
<td>( P_{13} )</td>
<td>( k=k_2; \ 0=0_2 )</td>
</tr>
<tr>
<td>14.</td>
<td>(7, 8)</td>
<td>( W_{78} )</td>
<td>( P_{14} )</td>
<td>( k=k_2; \ 0=0_2 )</td>
</tr>
<tr>
<td>15.</td>
<td>(8, 9)</td>
<td>( W_{89} )</td>
<td>( P_{15} )</td>
<td>( k=k_3; \ 0=0_3 )</td>
</tr>
<tr>
<td>16.</td>
<td>(8, 4)</td>
<td>( W_{84} )</td>
<td>( P_{16}=1-P_{15} )</td>
<td>( k=k_4; \ 0=0_4 )</td>
</tr>
<tr>
<td>17.</td>
<td>(9, 9')</td>
<td>( W_{99} )</td>
<td>( P_{17} )</td>
<td>( k=k_4; \ 0=0_4 )</td>
</tr>
<tr>
<td>18.</td>
<td>(9, 0)</td>
<td>( W_{90} )</td>
<td>( P_{18}=1-P_{17} )</td>
<td>( k=k_4; \ 0=0_4 )</td>
</tr>
</tbody>
</table>
− (5, 0): Return to initial state. This can happen for two reasons: in the case of processing the bits have passed or the processing of bits has not yet been completely finished (for example, only the 0th bit has been processed and it is necessary to wait for the processing of the 1st bit);
− (5, 9): informing the system that the group of bits has successfully passed the validation;
− (5, 0): Return to initial state. This can happen due to invalid input data;
− (3, 9): informing the system that the group of bits has successfully passed the validation;
− (4, 6), (4, 7), (6, 8), (7, 8): processing of the high-order bit of the license key, during which the useful code is executed and transitions to the corresponding states;
− (8, 4): return to the initial state. This can happen for two reasons: in the case of processing the bits have passed or the processing of bits has not yet been completely finished (for example, only the 0th bit has been processed and it is necessary to wait for the processing of the 1st bit);
− (8, 9): informing the system that the group of bits has successfully passed the validation;
− (9, 0): Return to initial state. This can happen for two reasons: if the bit processing has passed or the bit processing is not yet completely finished;
− (9, 9): All bits passed the test successfully. The license key is correct. Transition to the state of regular operation of the software product.

Research [11] shows that the W-function of the transition between states \( i, j \) is determined by the expression:

\[
W_{ij}(x) = p_{ij} \int_{-\infty}^{x} e^{ix} \zeta_{ij}(x) dx, \tag{1}
\]

where \( \zeta_{ij}(x) \) – state transition probability density \( i, j \);\n
\( p_{ij} \) – the probability of transition from state \( i \) to state \( j \).

As part of the study, we will accept the hypothesis that the use of the Gamma distribution in modeling as a key one in describing probabilistic transitions from state to state will allow us to achieve unification of the model of the software licensing security process. Unification lies in the fact that a decrease or increase in the number of obfuscation operations will slightly change the simulation results while the structural architecture of the model remains unchanged.

Thus, in the considered GERT network, the transition probability density function is defined by the Gamma distribution with variable coefficients \( k \) and \( \theta \):

\[
\zeta(x) = \frac{x^{k-1} e^{-x/\theta}}{\theta^k \Gamma(k)}. \tag{2}
\]

As the result, \( W \)-function looks like:

\[
W_E(s) = \frac{W_E'(s)}{1 - W_E(s)};
\]

\[
W_E'(s) = W_{o0}W_{10}W_{50}W_{90} + W_{00}W_{25}W_{90} + W_{03}W_{50}W_{90} + W_{04}W_{40}W_{90} + W_{04}W_{47}W_{90};
\]

\[
W_E(s) = W_{00}W_{15}W_{50} + W_{02}W_{30}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{89} + W_{04}W_{47}W_{78}W_{90};
\]

\[
W_E(s) = W_{01}W_{15}W_{50} + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{84} + W_{04}W_{47}W_{84} + W_{01}W_{15}W_{50} + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{89};
\]

The generated table of characteristics of the branches considered in the GERT-model of the branches and distribution parameters is presented in Table 1.

Using the described expression in [11], personifying the product of \( W \)-functions describing the successful and unsuccessful execution of algorithms, we obtain the resulting expression for calculating the equivalent transfer functions:

\[
W_E(t) = \sum_{i=1}^{5} \left( \frac{1.5,7,17 + 2.2,2,17 + 3.9,17 + 4,12,14,15,17 + 5,1,13,15,17 + 6,1,2,3,3}{1,1,2,3,3} \right) \times
\]

\[
1 - \frac{1.5,8 + 2.6,8 + 3,10 + 4,11,13,16 + 5,1,12,4 + 6,1,12,4 + 7,1,12,4 + 8,1,12,4 + 9,18 + 10,11,13,15,18 + 11,12,14,15,18 + 12,1,13,3,4 + 13,1,13,3,4 + 14,1,13,3,4}{1,1,1,2,4} \times
\]

\[
W_{o0}W_{10}W_{50}W_{90} + W_{00}W_{25}W_{90} + W_{03}W_{30}W_{50} + W_{04}W_{46}W_{68}W_{89} + W_{04}W_{47}W_{84} +
\]

\[
W_{01}W_{15}W_{50} + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{89};
\]

Using the probability density distribution (4), we obtain the graph of the probability density distribution, shown in Fig. 4. In this case, the probabilities were chosen as follows:

\[
P_1 = P_2 = P_3 = P_4 = 0.25; P_5 = P_6 = 1.0; P_7 = 0.3; P_8 = 0.5; P_9 = P_{10} = 1.0; P_{11} = 0.3; P_{12} = 0.2.
\]

The network can be constructed in such a way that if in some state \( i \) the start of one of several subsequent operations is possible, then the probabilities of starting \( p_{ij} \) of any of these operations form a complete group of inconsistent events:

\[
\sum_j p_{ij} = 1, \forall i. \tag{3}
\]

In this case, the probability of completing the entire network from source to drain is 1.

To show that all nodes satisfy condition (3), we calculate the probability of the entire process, which is calculated by the expression:

\[
p_E = W_E(s) - M_E(s) \bigg|_{s=0} = W_E(0) = 1, \tag{4}
\]

where \( M_E(s) \) – generating function, while,

\[
M_E(s) \bigg|_{s=0} = \frac{\int_{-\infty}^{\infty} f_E(x) dx}{\int_{-\infty}^{\infty} f_E(x) dx} = \int_{-\infty}^{\infty} f_E(x) dx = 1, \tag{5}
\]

where \( f_E(x) \) – distribution density function.
In Fig. 3 shows a graph of the distribution density of the time of execution of the entire process of ensuring software licensing protection with \( k = [2, 4, 4, 3] \) and \( \theta = [2.7, 1.8, 1.8, 3.5] \). By integrating the probability distribution density, we obtain the distribution function, the graph of which is shown in Fig. 4.

![Fig. 3. Graphs of the distribution density of the execution time of the process of ensuring the licensed security of a software product using the GERT network](image)

![Fig. 4. Diagrams of distribution functions of the process of ensuring the licensed security of a software product using a GERT network](image)

The mathematical expectation and variance of the obtained functions are calculated according to the expressions:

\[
\mu = \frac{W_E(t) dt}{\theta} = 3.52 \quad (6)
\]
\[
\sigma^2 = \frac{W_E(t) d^2 t}{\theta} - \mu^2 = 117.01. \quad (7)
\]

**Increasing the complexity of the GERT network.** After examining the developed GERT network, you can see that in order to find the correct key (i.e., the correct initial marking), you need to iterate over only 16 values. According to the use of keys that are less than 256 bits long, today it is insufficient to provide security. Consequently, the proposed GERT network is not suitable for practical use. To increase the key length when using GERT networks, it is proposed to increase (scale) the given elementary network.

Within the framework of the study, it was decided to use 2 types of scaling: horizontal (Fig. 5) and vertical (Fig. 6). When scaling vertically, the transitions to the first states (for example, (0-1), (0-2)) are modified by adding new branches. In this case, the overall probability of transition from state 0 decreases in proportion to the length of the key. In this case, the complexity of the graph remains similar to the original one.

With horizontal scaling, part of the key is wedged into the processing of specific bits. For example, from Fig. 5, you can see that we will go to state 5 in the case when not only bits 0 and 1 are valid, but also when bits 4-7 are valid (states 1\(^{-}\), 4\(^{-}\)).
**Conclusions**

1. Algorithms for safe transition in GERT-networks have been developed, which are used as a graph of the control logic of a software product. This logic is implemented depending on the identification or serial number.

2. A model of a licensed security system has been developed, which has not only empirical, but also theoretical grounds for resistance to attacks by an intruder.

Further research is aimed at studying the behavior of the GERT network during horizontal and vertical growth.
Fig. 6. Developed GERT network of software licensing security process at vertical scaling

References


Розробка математичної моделі захисту програмного продукту від неліцензійного копіювання на основі GERT методу

С.Г. Семенов, Чжан Ліцзян, Цао Вейлінь, В.В. Давидов

Предметом дослідження є методи і алгоритми побудови GERT-мереж, що забезпечують прийомний перехід по глядах графа керуючої логіки і можливість кодування цифрових водяних знаків у вуках існуючих загроз ліцензійної безпеки програмних систем. Метою роботи є розробка системи ліцензійної безпеки програмного продукту на основі алгоритмів прийманого переходу в GERT-мережах, що несуть у собі властивості цифрових водяних знаків. У статті варіюється наступне завдання: розробка моделі системи ліцензійної безпеки програмного забезпечення на основі побудованих алгоритмів, що використовують GERT-мережі. Використовуються методи математичного моделювання, чисельного експерименту, теорії слідності і криптографії. Отримані наступні результати: на основі проведенного аналізу існуючих моделей забезпечення ліцензійної безпеки програмного забезпечення були виділені основні вимоги до синтезуваних в рамках розробленої моделі алгоритмів, а також до машинної реалізації алгоритму. На основі досліджених систем водяних знаків, методів атаки на них, а також висунутих вимог до розробленої моделі, був сформований алгоритм ліцензійної безпеки на базі 4-бітових ліцензійних ключів, заснований на системі водяних знаків. Відмінною рисою цього алгоритму є стійкість до існуючих загроз ліцензійної безпеки через використання прийманого переходу. Продемонстровано розвиток розробленої моделі шляхом її горизонтального та вертикального масштабування для використання ліцензійних ключів більшої довжини. Висновки: вперше розроблено алгоритми прийманого переходу в GERT-мережах, що використовуються в якості графа керуючої логіки програмного продукту. Дана логіка впроваджується в залежності від ідентифікаційного або серійного номера; розроблена модель системи ліцензійної безпеки, що має не тільки емпіричну, а й теоретичну обґрунтуваність стійкості до атаки хуляминника.

Ключові слова: безпека програмного забезпечення, GERT модель, ліцензійна безпека.

Разработка математической модели защиты программного продукта от нелегального копирования на основе GERT метода

С.Г. Семенов, Чжан Личжан, Цао Вейлин, В.В. Давидов

Предметом исследования являются методы и алгоритмы построения GERT-сетей, обеспечивающие скрытый переход по ветвям графа управляющей логики и возможность кодирования цифровых водяных знаков в условиях существующих угроз лицензионной безопасности программных систем. Целью работы является разработка системы лицензионной безопасности программного продукта на основе алгоритмов скрытого перехода в GERT-сетях, несущих в себе свойства цифровых водяных знаков. В статье решается следующая задача: разработка модели системы лицензионной безопасности программного обеспечения на основе построенных алгоритмов, использующих GERT-сети. Используются методы математического моделирования, численного эксперимента, теории сложности и криптографии. Получены следующие результаты: на основе проведенного анализа существующих моделей обеспечения лицензионной безопасности программного обеспечения были выделены основные требования к синтезируемым в рамках разрабатываемой модели алгоритмам, а также к машинной реализации алгоритма. На основе исследований систем водяных знаков, методов атаки на них, а также выдвинутых требований к разрабатываемой модели, был сформирован алгоритм лицензионной безопасности на базе 4-битных лицензионных ключей, основанный на системе водяных знаков. Отличительной особенностью данного алгоритма является стойкость к существующим угрозам лицензионной безопасности благодаря использованию скрытых переходов. Выводы: впервые разработаны алгоритмы скрытого перехода в GERT-сетях, использующихся в качестве графа управляющей логики программного продукта. Данная логика внедряется в зависимости от идентификационного или серийного номера; разработана модель системы лицензионной безопасности, имеющая не только эмпирические, но и теоретические обоснования стойкости к атакам злоумышленника.

Ключевые слова: безопасность программного обеспечения, GERT модель, лицензионная безопасность.