INFORMATION TECHNOLOGY

UDC 004.056
DOI: 10.18372/2306-1472.67.10431

Denys Samoilenko

PERMUTATION-BASED POLYMORPHIC STEGO-WATERMARKS FOR PROGRAM CODES

National University of Shipbuilding after Admiral Makarov, Mykolaiv, Ukraine
3, Lenin ave, Mykolaiv, 54010, Ukraine,
E-mail: DenNikSam@gmail.com

Abstract

Purpose: *One of the most actual trends in program code protection is code marking. The problem consists in creation of some digital "watermarks" which allow distinguishing different copies of the same program codes. Such marks could be useful for authority protection, for code copies numbering, for program propagation monitoring, for information security proposes in client-server communication processes.* ***Methods:*** *We used the methods of digital steganography adopted for program codes as text objects. The same-shape symbols method was transformed to same-semantic element method due to codes features which makes them different from ordinary texts. We use dynamic principle of marks forming making codes similar to be polymorphic.* ***Results:*** *We examined the combinatorial capacity of permutations possible in program codes. As a result it was shown that the set of 5-7 polymorphic variables is suitable for the most modern network applications. Marks creation and restoration algorithms where proposed and discussed. The main algorithm is based on full and partial permutations in variables names and its declaration order. Algorithm for partial permutation enumeration was optimized for calculation complexity. PHP code fragments which realize the algorithms were listed.* ***Discussion:*** *Methodic proposed in the work allows distinguishing of each client-server connection. In a case if a clone of some network resource was found the methodic could give information about included marks and thereby data on IP, date and time, authentication information of client copied the resource. Usage of polymorphic stego-watermarks should improve information security indexes in network communications.*

Keywords: data protection; digital watermarks; information security; program code; steganography.

1. Introduction

One of the most actual trends in program code protection is code marking problem. Main idea of code marking consists in creation of some digital "watermarks" (DW) which allow distinguishing different copies of the same program codes (means the same by functionality). Such DW could be useful for authority protection, for code copies numbering, for program way monitoring from a creation lab to finite user.

For information security proposes aforementioned DW could be used as hidden data in client-server communication processes. Suppose, that server has a mechanism of different DW creation for each client request. It means that the same network information resource (NIR), site, portal, etc. – for different users has the same visual face, but different codes which forms the face. In such case appears a possibility to establish date, time, IP and, possible, some additional information of origins for the concrete client code.

For example, security service of some corporation finds in Network a clone of its own NIR. It is rather obvious that clone was made from the real NIR by codes coping. So, in clone should be DW. Restoration of DW number and selection of its record in access database (DB) gives information stored for the excess investigation start point.

2. Analysis of the latest researches and publications

Technologies of hidden data marking are widely discovered in modern digital steganography [1]. Basic principle of information concealing is creation of the "subliminal channel" [2]. Such channel means the way of hidden information restoration. For example, it could be each first letter in word or the less bit in pixel's color. Term subliminal means that only two communicating sides know this way. Some schemes of subliminal channel construction for authentication and signature problems are shown in works [3-6].

If we want to use DW for protection proposes, we should accent some features. First, there is no second side in communications. Creation and restoration of DW information occurs at the same place. So, the subliminal channel becomes a subliminal procedure known only for its author. Second, DW should become an integral part of its container. It limits types and technologies of DW creation. DW designed for images could not be used (in the same form) for texts. Digitally generated DW for printed issues [7] are unsuitable for recorded sounds and so on.

Choosing of web-client program (software) codes (WPCs) as main container for DW required emphasizing following features:

- WPCs are text containers. So we should look for appropriate methods in text steganography [8-10]. Designed for other containers methods are unacceptable.
- WPCs do not allow methods, related with symbol exchanging (combining symbols with different codes but same faces) or with words disposition [9-10].
- Although manipulations with fonts, sizes, styles etc. are acceptable in code editor, they will have no effect when WPCs include in HTML document.
- WPCs allow most popular method of "whitespace watermarking" which involves words separation with different amount of whitespaces. But tends of web-traffic minimization require functions declaration in shortest way. In this case additional whitespaces will reveal DW. Besides, WPCs size will be different for different DW whereby secretiveness of DW will decrease.
- WPCs allow changes in variables and user-defined functions and classes names in case if changes are synchronic in all code places. Also, declaration of variables allows permutations.

The last WPCs feature is least of all discussed in scientific issues. Hereinafter we will consider developing specified DW methodic.

3. Research tasks

In the paper, the methodic of digital watermarks including in and restoring from program codes is carried out. Aimed to be adopted for web-oriented codes methodic requires speed-optimized calculation algorithm so it will be described. It will be also shown that appropriate results could be provided for codes used only 5-7 variables.

4. Researches results

Consider V to be a set of variables used in a specified WPC. Let v – is a size of V.

Consider S to be a set of symbols that are valid for variable names. It is possible to include in S all possible valid literals but for implementation aforementioned principle of traffic minimization it will be better to use only single symbols for variable names. For non-web-oriented software S set could be expanded. Let s – is a size of S.

The number of permutations in variable declaration order (Np) depends on number of variables only:

$$N_P = v!$$

Different symbols selection for variable names allow NS combinations (combinatorial partial permutations):

$$N_S = s \cdot (s-1) \cdot \ldots \cdot (s-v+1) = \frac{s!}{(s-v)!}$$

The whole number of DW will be a product of described parts by virtue of its independence:

$$N_{DW} = N_P \cdot N_S$$

Suppose we will use only alphabet symbols (small and capital) for variable names. So,

$$N_{DW} = \frac{52! \cdot v!}{(52-v)!},$$

where 52 is a size of set which includes 26 small and 26 capital letters. Table 1 shows values of N_{DW} for different variable values v.

For the estimation of shown numbers let's consider one from most popular NIR – google.com. By statistics it processes approximately $4.5 \cdot 10^9$ requests per day (by *worldometers.info*). So, DWs in basis of 5 variables allow distinctive marking all server answers during a week. Usage of 6 variables prolongs this period for 6 years, 7 variables – for millennium. For less popular NIRs conclusions could be freely forced 100-1000 times.

Table 1. Number of different DW (N_{DW}) in dependence on number of variables (v)

v	N_{DW}
1	52
2	5304
3	795 600
4	155 937 600 (~10^8)
5	37 425 024 000 (~10^{10})
6	10 553 856 768 000 (~10^{13})
7	3 398 341 879 296 000 (~10^{15})

It may seem that permutations in declaration order could contradict to variables names changes. For 3-variable example: declaration order is 1-2-3 and names are 1-A, 2-B, 3-C produce code like "var A,B,C". At the same time, for order 3-2-1 and names 1-C, 2-B, 3-A it will be produced exactly the same code.

Really, semantic variable order could be restored by program instructions analysis. For the shown example instruction "print(A,B,C)" in first case transforms to "print(C,B,A)" in second because semantic role of the first variable (named A in first case) passes directly to first variable (named C in second case) independently on order of its declaration. For the simplification of order restoration in practice, it is possible to place variable initialization instructions nearly its declaration like "var A,B,C; C=0; B=0; A=0;". Order of variables initialization establishes its semantic order while declaration order corresponds to DW.

The next problem of WPC marking is developing a methodic for DW numbering. Means that by known DW number we can construct WPC and vice versa, we can restore DW number from given WPC.

The methodic of DW construction for the given number N could be divided on follow procedures:

1. Checking general possibility: DW could be constructed for $N< N_{DW}$
2. Decomposing N on basis (N_p, N_s) i.e. representing N in form $N = p \cdot N_s + s$
3. Constructing permutation with number p
4. Constructing symbol combination with number s

Inverse methodic for restoring DW number from given code means descending usage of shown sequence.

First two steps are rather obvious for detailed description necessity. So, let's focus on the last two.

Numbering of permutation is well-known problem with plenty of solutions [12]. We'll use one of them just adopted for PHP language. PHP language is chosen as one of most popular. Program codes for simpler problem of number restoration will look like

```
$G = array(2,0,3,5,4,6,1);
//the permutation G, here 7-element example
$g = count($G);          //size detection
$P = array();            //array for reference sequence
for($i=0;$i<$g;$i++) $P[]=$i;  //reference sequence
$p=0;                    //variable for perm. number
for($i=0;$i<$g;$i++){//cycle for elements
  $m=$G[$i];             //the element of G
  $k=$P[$m];             //its order in P
  for($j=$m+1;$j<$g;$j++) $P[$j]--;
//order decreasing for subsequent elements
  $p=$p*($g-$i)+$k;
//number forming by Horner's method
}
```

Printing the $p variable shows 1479.

Permutation construction bit more complex problem. Codes could be outlined like

```
$w=1479;                 //the permutation number
$g=7;                    //the size of sequence
$P = array();            //array for reference sequence
for($i=0;$i<$g;$i++)                    $P[]=$i;
                         //reference sequence forming
$G=array();              //array for permutated sequence
$pb=array();             //array for basis
$pb[$g-1]=1;             //last basis weight is 1
for($i=2;$i<=$g;$i++)                   //by the cycle form basis as n!
  $pb[$g-$i]= $pb[$g-$i+1]*$i;
//so previous element multiplies for $i
$pd=array();             //array for decomposition
for($i=1;$i<$g;$i++){//cycle for sequence
  $pd[$i-1]=(int)floor($w/$pb[$i]);  //a "digit" in basis
  $pr=$w-$pd[$i-1]*$pb[$i];
//residue for subsequent decomposition
  $w = $pr;              //renaming for cycle repeating
}                        //
$pd[]=0;                 //the last "digit" always 0
for($i=0;$i<$g;$i++){//cycle for sequence
  $m=$P[$pd[$i]];    //extracting element by "digit"
  for($j=$pd[$i]+1;$j<$g;$j++)
//shifting reference sequence
    $P[$j-1]=$P[$j];  //after extracted element
  $G[]=$m;               //adding element to G sequence
}
```

Printing the G array shows 2035461 sequence which corresponds to 1479 number from the example above.

Algorithm for symbol combination (partial permutations), as a rule, derives from permutations algorithm. But for network application it is better to imply some optimizations, because the main complexity of permutations algorithm is to shift

array elements after extraction one of them (bold comment in previous codes).

As it was considered before, the number of variables in DW is small (about 5-7). So arrays transforming has rather low complexity. Whereas symbols array 10 times longer (52 letters) and it seems to be inappropriate where performs near 52 shift operations for each symbol.

For constructing optimized algorithm consider in details partial permutations. For the small example consider S to be a set of 5 elements: S={1,2,3,4,5} from which names of 3 variables are formed in its semantic order. Tab. 2 shows all possible partial permutations.

Table 2. "3 from 5" partial permutations

R \ C	1	2	3	4
1	123 124 125	132 134 135	142 143 145	152 153 154
2	213 214 215	231 234 235	241 243 245	251 253 254
3	312 314 315	321 324 325	341 342 345	351 352 354
4	412 413 415	421 423 425	431 432 435	451 452 453
5	512 513 514	521 523 524	531 532 534	541 542 543

We can see 5 (s) rows, 4 (s-1) columns and 3 (s-2) items in each table cell. Number of row is considered to be R (1-5), column – C (1-4), item – I (1-3). Some regularity in shown sequence could be observed:

- first element n_1 is a number of row R and n_1 changes at every (s-1)(s-2)=12 elements
- second element n_2 is a constant for every (s-2) items. But second element could not be equal to first element, so column number C could be defined from n_2 as:

$$C = \begin{cases} n_2, n_2 < n_1 \\ n_2 - 1, otherwise \end{cases}$$

- the last (third) element could not be equal either first or second element. The item number in a cell could be calculated as:

$$I = \begin{cases} n_3, n_3 < n_3 \wedge n_3 < n_3 \\ n_3 - 1, (n_3 < n_1 \wedge n_3 \geq n_2) \vee (n_3 < n_2 \wedge n_3 \geq n_1) \\ n_3 - 2, otherwise \end{cases} \quad (1)$$

Using these interrelations we can define a location of given permutation in Tab. 2. For example lets analyze permutation "423" – first digit "4" means 4th row without additional conditions (R=4); second digit "2" is less than first, so it means 2nd column (C=2); third "3" is less than first but greater than second, so it means "3-1" = 2nd item in the cell (I=2).

In software programming, as a rule, array's elements are calculated from reference point "0". Being intended on program algorithm construction lets reduce by 1 all counter-like elements: R, C and I. In such case, the relative number of "423" permutation will be N=(R-1)(s-1)(s-2)+(C-1)(s-2)+(I-1) = 3·12+1·3+1=40. If the first item in Tab. 2 has number 0, the 40*th* item, as one can freely check, is "423".

Inverse problem is to find a permutation with relative number n. For this problem we should use the numerical-basis decomposition. The basis is formed by quantities of digit positions. First digit is responsible for (s-1)(s-2) quantity and so on. Finally, we should find residues of n divided by basis numbers.

For the chosen example, first digit has weight 12, second – 3, third – 1. Let's decompose quantity 40 for this basis: 40 = 3·12 + 1·3+1. So, n digits are "3"-"1"-"1". Considering starting point at "0" and symbol skipping with described rules will have "4"-"2"-"3" combination.

With a whole S set (52 symbols "a-zA-Z") let's construct 3-variable DW which is 12345th by number. First, define a basis: 2550-50-1. Value 2550 is a product (52-1)(52-2). In this basis given number decomposes like 12345=4x2550 + 42·50+45. Thus basic digits are 4, 42 and 45.

First digit (4) without conditions is the 4th symbol in S set. Referring that set origin is 0 we should consider the 5th element – the "e" letter. Second digit should be corrected by increasing for 1 because it greater than first one. Origin correction leads us to 44th letter – "R". The third digit should be increased for 2 because it greater than first and greater than second digits. Considering origins we'll have 48th letter – "V".

Summing up, the DW for 12345th names permutation is "eRV". It means that first variable should be declared with "e" name, second and third variables – as "R" and "V" respectively.

DW's number restoration in the mentioned example could be shown in the following way. We define a letters (symbols) numbers in S set. For "eRV" DW they will be 5, 44 and 48, as written above. Next we should correct these numbers by decreasing for 1. It means that first row contains

numbers with formal zero first digit – less than 2550 numbers. The same property has the rest digits. So, we'll have 4, 43 and 47 digits. Next we decrease digits if they greater than previous ones. Thus, 43 is decreased by 1 (it is greater than one previous digit) and 47 is decreased by 2 (it is greater than 2 previous digits). Corrected digits are 4, 42 and 45. Finally, we combine digits with basis and obtain $4 \cdot 2550 + 42 \cdot 50 + 45 = 12345$.

Described methodic could be realized as particular program. So far as DWs are focused on network client-codes it is expedient to use server-side programming language for the realization. DW creating codes will look like

```
$S = "abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ";
//S – the set of DW symbols
$s = strlen($S);          // the size of S
$V = "";                  // V – the set for DW symbols
$v = 3;                   // number of variables
$N=12345;                 // the DW number
$b = array();             // array for basis numbers
$b[$v-1] = 1;             // the last basis element is 1
for($i=$v-2;$i>=0;$i--)                    // loop for
basis
  $b[$i] = $b[$i+1]*($s-$i-1); //basis numbers
computing
$n = array();             // array for DW "digits"
$r = $N;                  // variable for residue storage
for($i=0;$i<$v;$i++)  // loop for V-set
{  $d = (int)floor($r / $b[$i]); // current digit
   for($j=0;$j<$i;$j++)                    //loop for
correction (1)
      if($d >= $n[$j]) $d++; // the correction
   while(in_array($d,$n))
// if corrected digit matched already calculated ones
      $d++; // it should be additionally increased
   $n[$i] = $d; // finally corrected digit is added to
array
   $V .= $S[$d]; // the n-th symbol is added to V-set
   $r %= $b[$i];              // for the next cycle we
use N
// modulo current basis number
}
```

If we finish the codes by print($V) command we will see the "eRV" output which corresponds to 12345 number used in codes.

Codes for given DW's number restoration could be formed in following way.

First, we need a function that returns a number of given symbol in S-set. Some string-type function could be used, but for demonstration purposes we'll construct the new one.

```
function indexOf($a,$A)
// computes index of $a symbol in $A set
{   $i = -1;              // value for return
   $lim = strlen($A);  // loop limit
   do ++$i; while($a!=$A[$i] && $i<$lim);
// searching for $a in $A
   if($i==$lim) return -1;// if $i exceeds limit
returns -1
   else return $i;       // otherwise index returns
}
```

Second, using constructed function we could define DW's number:

```
$V = "eRV";               // DW symbols
$n = array();             // array for symbols numbers
$c = array();             // array for corrected numbers
(1)
for($i=0;$i<$v;$i++)  //loop for DW symbols
   $n[$i] = indexOf($V[$i],$S);//symbol's index
for($i=0;$i<$v;$i++)  //correction loop
{  $c[$i] = $n[$i];
//first, corrected number is equal to symbol's index
   for($j=0;$j<$i;$j++)                    // but
      if($n[$i] >= $n[$j])
// if a number greater than previous ones
         $c[$i]--;       // it should be decreased by 1
}                         // end of correction loop
$N = 0;                   // variable for the DW number
$b = 1;                   // basis number
for($i=$v-1;$i>=0;$i--)                    // loop for
basis
{  $N += $b*$c[$i];  // number increasing by
corresponding basis
   $b *= $s-$i;           // next basis number
}
```

Similar to previous codes adding print($N) command will show "12345" output.

It could be simple observed that in symbol selection codes all cycles have limits that no longer than v – the number of variables. In comparison with algorithm based on permutations numbering (with full-array cycles) the shown codes have approximately 10 times less amount of array tracing

operations. For client-server communications such optimization allows to serve more requests for the same time.

Finally, for all stages example, let's construct DW for number N=654321 using 3 variables with single-letter names. For letters choosing we have a set of 52 symbols.

1) We should compute basis: Ns=52·51·50=132600, Np=3!=6, and general DW limit: N_{DW}=Ns·Np=795600. The given DW number (N) should be checked to be strictly less than the limit N_{DW}, otherwise DW construction is impossible. In the given case $N< N_{DW}$, so we pass to next step.

2) The DW number N should be decomposed on basis computed previously: $N = p \cdot Ns + Np$. For the example number we'll have decomposition in form: 654321 = 4·132600 + 123921. It means that we should construct 4^{th} permutation in declaration order and choose the 123921^{st} combination (partial permutation) of symbols for names.

3) To construct the p^{th} permutation we should decompose p on (n-1)! permutation basis. For 3-element sequence the permutation basis will be 2-1-0. In this basis p=4 is decomposed like 4=2·2+0·1+0·0. On other words, in permutation basis p has digits 2-0-0. The digits allow us to find the p^{th} sequence if we have the reference (0^{th}) sequence. Consider "1-2-3" to be the reference sequence. First p's digit (2) means that we should extract 2^{nd} element from reference sequence. Because of zero counting origin we should really extract 3^{rd} element – "3". After each extraction reference sequence should be left-shifted for filling extracted position. So, after first extraction we'll have "1-2" residue. Next we'll extract second digit (0) which means the first element "1". After left-shifting the sequence will be "2". This last symbol corresponds to last digit. Finally, we have "3-1-2" sequence for 4^{th} permutation.

4) To choose symbols for variables names we should find Np^{th} partial permutation. First, it is necessary to decompose Np=123921 onto partial basis. For v=3 partial basis will be 2550(51·50)-50-1. Np decomposition in partial basis will look like 123921 = 48·2550 + 30·50+21. With described above algorithm we can select symbols. As far as all "digits" are in descending order (48-30-21) there will be no digit corrections. So we take 48^{th}, 30^{th} and 21^{st} symbols that are "W"-"E"-"v".

5) By collection of names symbols and declaration order sequence we can produce a program code fragment with given DW number. As it was marked before, the reference permutation sequence must be used in initialization order. Because of choosing "1-2-3" reference, the initialization codes will look like: "W=0;E=0;v=0;" in accordance with symbols sequence obtained in par.4). Variable declaration must include other sequence which corresponds to permutation order p. Found in par.3) sequence "3-1-2" could be reflected in codes like "var v,W,E;" – on first place the third variable "v" is declared, the next are first "W" and second "E" ones. Since declaration codes should be placed before initialization the codes with 654321^{st} DW will look like "var v,W,E;W=0;E=0;v=0;". As it follows from par.1) such small program fragment of 22 printed symbols could be written in 795600 different ways without any semantic violations.

It's useful to note that reference and permutated sequences could be included in codes in different way. It is possible to use reference sequence in declarations and permutated one in initializations. In such case codes will look like "var W,E,v;v=0;W=0;E=0;". There are no fundamental differences in two shown schemes. So programmer could select one of them by own preferences.

5. Conclusions

It was proposed a program codes stego-marking methodic adopted for client-server communications. The methodic is based on full and partial permutations in variables names and its declaration order.

It was shown that amount of 5-7 variables is enough for marking purpose for most network information resources even for single-symbol variable names and symbols limitation by alphabet letters.

As a methodic part it was proposed the optimized algorithm for partial permutations. The algorithm has the less complexity in compare with classic schemes due to usage of correction cycles limited by partial set size. For the selected problem calculation speed gains approximately 10 times.

It was outlined some PHP code fragments and examples that realize described methodic.

References

[1] *Cachin, C.* Digital Steganography [WWW document]/ Christian Cachin // IBM Research Zurich Research Laboratory CH-8803 Rüschlikon, Switzerland cca@zurich.ibm.com February 17, 2005 – Access from: https://www.zurich.ibm.com/~cca/papers/encyc.pdf

[2] *Simmons, G. J.* The Prisoners' Problem and the Subliminal Channel [Text] / G. J. Simmons // Advances in Cryptology. – 1985, Vol. 209, P. 364–378.

[3] *Bäcker, C.* Subliminal Channels in Cryptographic Systems [WWW document]/Christian Bäcker/ – Access from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.392.8043&rep=rep1&type=pdf

[4] *Simmons, G. J.* Authentication Theory / Coding Theory [Text] / G. J. Simmons // Advances in Cryptology. – 1985. – Vol. 196. – P. 411–431.

[5] *Pei, D. Y.* Authentication Schemes [WWW document] / D. Y. Pei. // Singapore: Institute for Mathematical Sciences.– 2001. – 36 p. – Access from: www2.ims.nus.edu.sg.

[6] *Zhang, F., Lee, B., Kim, K.* Exploring Signature Schemes with Subliminal Channel / Fangguo Zhang, Byoungcheon Lee, Kwangjo Kim // SCIS 2003 The 2003 Symposium on Cryptography and Information Security Hamamatsu,Japan, Jan. 26-29, 2003 The Institute of Electronics, Information and Communication Engineers

[7] *Samoilenko D., Miroshnichenko O., Popov D.* Fractal images usage for printed production protection. [Text] / Kvalilohiya book : Coll. Science. pr. / Eng. Acad. printing., 2010, № 2 (18), p. 77-81

[8] *Michaud, E.* Current Steganography Tools and Methods (Use offense to inform defense. Find flaws before the bad guys do.) [WWW document] / Erin Michaud / SANS Institute, As part of GIAC practical repository, 2003 Access from: https://cyber-defense.sans.org/resources/papers/gsec/current-steganography-tools-methods-104695

[9] *Agarwal, M.* Text steganographic approaches: a comparison [Text] / Monika Agarwal // International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1, January 2013

[10] *Sagan, C.* 10 Data Hiding in Text [WWW document] Access from: http://www.springer.com/cda/content/document/cda_downloaddocument/9780387003115-c10.pdf

[11] *Garg, M. A.* Novel Text Steganography Technique Based on Html Documents [Text] / International Journal of Advanced Science and Technology Vol. 35, October, 2011

[12] *Wagner D. J.* The Combinatorics of the Permutation Enumeration of Wreath Products between Cyclic and Symmetric Groups. PhD Thesis in Mathematics. [WWW document] / University of California, San Diego Access from: https://math.ucsd.edu/~thesis/thesis/jwagner/jwagner.pdf

Received 11 February 2016

Д. М. Самойленко. Переставні поліморфні стеганографічні водяні знаки для програмних кодів

Національний університет кораблебудування імені адмірала Макарова, пр. Леніна, 3, м. Миколаїв, Україна, 54010
E-mail: DenNikSam@gmail.com

Мета: Однією з найактуальніших тенденцій у захисті програмних кодів є їх маркування. Завдання полягає у створенні специфічних цифрових «водяних» знаків, які дозволяють розрізняти копії однієї програми. Такі знаки можуть знайти використання для захисту авторських прав, нумерування копій програмних кодів, моніторингу розповсюдження програм, а також для задач інформаційної безпеки у процесі клієнт-серверних комунікацій. **Методи**: Використано методи цифрової стеганографії, адаптовані до програмних кодів як текстових об'єктів. Метод знаків однакового нарису трансформовано до методу елементів однакової семантики для можливості використання у текстах програм, оскільки вони мають відмінності від звичайних текстів. Вжито принцип динамічного формування міток, що надало програмам ознак поліморфності. **Результати**: Досліджено комбінаторну ємність вибірок та перестановок, застосовних до програмних кодів. Показано, що набір з 5-7 поліморфних змінних є достатнім для більшості сучасних мережних завдань. Наведено та обговорено алгоритми створення та відновлення міток. Алгоритми ґрунтуються на вибірках імен змінних та перестановках у порядку їх оголошення. Алгоритм вибірок було оптимізовано щодо обчислювальної складності. Наведено фрагменти програм мовою PHP, що реалізують запропоновані алгоритми. **Дискусія**: Запропонована у роботі методика дозволяє розрізняти усі з'єднання клієнта із сервером. У разі виявлення клону мережного ресурсу, використання запропонованої методики може надати інформацію про включені мітки, а відтак, дані про IP, дату та час, автентифікаційні дані клієнта, який здійснив копіювання ресурсу. Впровадження поліморфних стеганографічних міток має покращити показники інформаційної безпеки мережних комунікацій.

Ключові слова: захист даних; інформаційна безпека; програмні коди; стеганографія; цифрові водяні знаки.

Д.Н. Самойленко. Перестановочные полиморфные стеганографические водяные знаки для программных кодов

Национальный университет кораблестроения имени адмирала Макарова, пр. Ленина, 3, г. Николаев, Украина, 54010
E-mail: DenNikSam@gmail.com

Цели: Одним из наиболее актуальных трендов в защите программных кодов является их маркировка. Задача заключается в создании специальных цифровых «водяных» знаков, позволяющих различать копии одной программы. Такие знаки могут найти применение для защиты авторских прав, нумерации копий программных кодов, мониторинга распространения программ, для задач информационной безопасности в процессе клиент-серверных коммуникаций. **Методы**: Использованы методы цифровой стеганографии, адаптированные для программных кодов как тестовых объектов. Метод знаков одинаковой формы преобразован в метод элементов одинаковой семантики для возможности применения в текстах программ, поскольку они все-таки отличаются от обычных текстов. Был использован принцип динамического формирования меток, вследствие чего программы приобрели признаки полиморфности. **Результаты**: Была исследована комбинаторная емкость перестановок и подстановок, применимых к программным кодам. Показано, что набор из 5-7 полиморфных переменных достаточен для большинства современных сетевых приложений. Приведены алгоритмы построения и восстановления меток, указаны их особенности. Основой алгоритма является процесс подстановки имен переменных и перестановки порядка их объявления. Алгоритм нумерации подстановок был оптимизирован по вычислительной сложности. Приведены фрагменты программ на языке PHP, реализующие предложенные алгоритмы. **Дискуссия**: Предложенная в работе методика позволяет различать все соединения клиента и сервера. В случае обнаружения клона сетевого ресурса, применение разработанной методики может предоставить информацию о внедренных метках и, следственно, данные о IP, дате и времени, персональных данных клиента, клонировавшего ресурс. Использование полиморфных стеганографических меток должно привести к улучшению показателей информационной безопасности при сетевых коммуникациях.

Ключевые слова: защита данных; информационная безопасность; программные коды; стеганография, цифровые водяные знаки.

Samoilenko Denys. PhD, Associate Professor.
Ship Electrical Equipment and Information Security Department,
National University of Shipbuilding after Admiral Makarov, Mykolaiv, Ukraine.
Education: Taras Shevchenko National University of Kyiv, Ukraine (2000).
Research area: information security of network resources.
Publications: 64.
E-mail: DenNikSam@gmail.com.