Identifying Software Theft Based on Classification of Multi-Attribute Features

Ye Guo
School of information, Xi'an University of finance and economics, China
Email: guoyexinxi@126.com

Mingyu Wang
School of information, Xi'an University of finance and economics, China
Email: wmyufe@163.com

Yangxia Luo
School of information, Xi'an University of finance and economics, China
NWU (China) -Irdeto Network-Information Security Joint Laboratory (NISL), Xi'an, China
Email: yxluo8836@163.com

***Abstract*—Due to the low performance caused by the traditional "embedded" watermark and the shortages about low accuracy and weak anti-aggressive of single-attribute birthmark in checking obfuscated software theft, a software identification scheme is proposed which is based on classification of multi-dimensional features. After disassembly analysis and static analysis on protecting software and its resisting semantics-preserving transformations, the algorithm extracts features from many dimensions, which combines the statistic and semantic features to reflect the behavior characteristic of the software, analyzing and detecting theft based on similarities of software instead of traditional ways depending on a trusted third party or alone-similarity threshold. Through giving the formal description about the algorithm, depicting the algorithm realization, after comparisons and analysis from the qualitative and quantitative, theoretical and experimental aspects, the results show that the algorithm contributes to the resistance to attacks, as well as the robustness and credibility, and has advantages compared with similar methods.**

***Index Terms*—software theft, classification learning, multi-dimension features, software birthmark**

I. INTRODUCTION

Malicious software attacks and software piracy already are an indisputable fact, rampant on a global scale and has seriously hampered the software industry sustainable development [1]. The present software protection is responsible by software developers, through encryption[2], sequence number, Key File, software dog, and so on, , these software protection methods have difficulty realizing the piracy tracking and try to provide a uniform legal basis because of different technologies. Watermark, such as Ref. [3] and Ref.[4] and birthmark in [5] were a good attempt in copyright identification and protection.

At present, the software watermarks can be divided into "embedded" watermarks and "constructed" watermarks by whether or not it is changed from the original program. The embedded watermarks are inevitable to affect program load and performance, such as dynamic path-based software watermarking [6], generating the watermark through encoding instruction sequences or memory address in program; Threading software watermarks [7], encoding watermark by thread competition, adding multithread to improve robustness will lead to reducing efficiency of program execution due to the introduction of a large number of threads; Dynamic data structure watermark [8], [9]: the watermark information hidden in the memory stack states or global variables domain of the program that will take up memory and change the global variables.

In order to prevent performance degradation when watermark is embedded into a program, researches are conducted mostly to limit the amount of loaded information, such as the typical system：Sandmark [10]. If there was more loaded watermark information, invisibility will decrease; if less information was embedded, the copyright authentication is insufficient, especially for a meaningful watermark (need to embed more information). In order to prevent the watermarks from being attacked, many technologies: encryption and tamper-resistant are also applied to consolidate them, but those will bring more running load inevitably [8], [11]. For the above summary, the "embedded" watermarking has two problems: (1) the software performance will be reduced; (2) It could be removed as time as possible and affect the copyright authentication. So whether don't insert information or the imbedded information is zero, and can also identify copyright.

In fact, image watermarks are also facing similar problems above, some scholars have proposed a new concept: "constructed" watermark—image zero-watermarking [12], [13] to solve that. Subsequently, the

doi:10.4304/jsw.9.6.1401-1411

text zero watermark [14] and database zero-watermark [15]. The study on the three types of zero-watermarking technology shows that they have a same characteristic that extract features of the carrier, and then use the features to identify carrier while didn't embedded information into carrier. Just think of software birthmark (unique feature of software). In this paper, software features and a scheme of the constructed software zero-watermark with features are researched. Whether or not the features are anti-aggressive and representative, or the copyright identification scheme constructed by features is reasonable, is related to the accuracy of software copyright discrimination. The paper focuses on software multi-dimension features, first using classification learning in pattern recognition to build the judgment model and identify unknown software in order to improve the accuracy and objectivity of the copyright identification.

The remainder of this paper is organized as follows. In Sections II: through the comparison and discussion on the related work, indicating the significance and value of this paper work; Section III gives the concepts, definitions and formal descriptions about software features and copyright identification in order to accurately understand; Section IV studies the overall model of the proposed software copyright identification based on multi--dimension features, its formal description, design and implementation; Simulation results, comparison with other conventional algorithms, and the experiment and analysis are given in Section V; Finally, the conclusion and outlook are drawn in Section VI.

II. RELATED WORK

For improving the embedded watermarking, researchers hope to find a kind of feature is a unique feature which is birthmark. The extracted objects and copyright identification schemes based on the objects are the key problems of correct judgment copyright.

A. *Software Feature Object*

The research on software features initiated the source-code plagiarism detection system, MOSS [16],[17], but, sometimes the source code is not available and can easily be changed; H. Tamada and the other three members [18] worked on executable code (binaries or bytecode) rather than source code, and took java bytecode set as software features; API calls [19] were choosen to take software features because API calls of a program can be unique and difficult to be forged for an adversary and later were improved for the dynamic API call-level and call-frequency [20] as software features; Heewan Park [21] used the operation code and the static stack as features to identify software, the christian collbergr team and other researchers [22-24], introduced n-gram and dynamic extraction methods to improve resistance to compression, encryption and packers attacks of features; Control flow edge was also used for software features by Hyun-il Lim [25]; Hunan University [26], researched component dependency graphs characteristics of the software, summarized in Table I:

TABLE I
FEATURE EXTRACTION ALGORITHMS

algorithms	Extraction methods	Extraction Object
$ssSWA_{window}$	static	Source-code
TNMMsmc	static	Bytecode
TNMMuc	static	API call set
TNMMis	dymamic	API call sequence
HHSTosb	static	operation code
$HHST_{Java}$	static	Control flow graph
$ZSSY_{.NET}$	Static/dymamic	Dependency graphs

The Table1 shows the shortages in software identification that was based on one type of feature objects (a single attribute or only one judgement condition), lack of attack resistance. For example software confusion attacks [27],[28]: rename confusion attack will impact on the class name in source; adding useless verbs (dead code)attack affects the frequency and order of API call; smoothing control flow attacks would affect the extracted edges and vertices of control flow graph, so using only one kind of attribute object is insufficient in representation of copyright and weak in resistance to different attacks.

In view of above shortage, one contribution of the paper is to improve the recognition accuracy by increasing the type of feature objects (multi-attribute features or multiple discriminate conditions), and give fully consideration of the software diversity to make features (or birthmark) with the attack resistance and representation.

B. *Copyright Identifications Based on Features*

There were two ways to identify software copyright based on extracted features: 1) proving copyright based on a trusted third party [23]; 2) judging software copyright based on a threshold [24].

Copyright proof based on a trusted third party is to use the extracted features and there is a watermark (owner information) to generate registration information stored in a third party (That can be called the zero-watermark embedding process). When need to prove the software copyright, calculating register information of the third party with the extracted features to retrieve the watermark (the zero-watermark extracting process). Shown in Fig.1 and Fig.2:

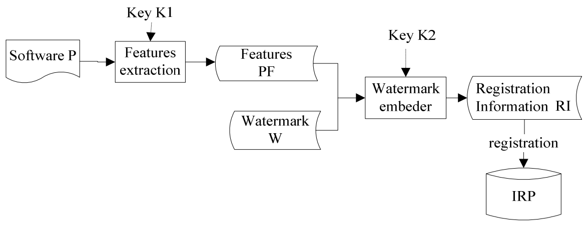


Fig.1. The embedding process based on the third party

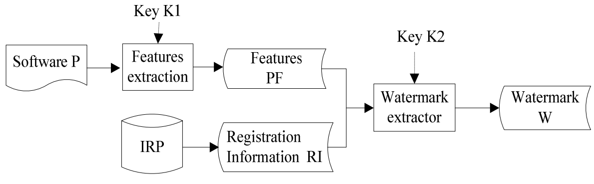


Fig.2. The extracting process based on third party

A shortage of the scheme (Shortage (2)) is a non-blind extraction process that needs to save features of the original software in a trusted third party which is potentially unsafe.

Software copyright identification is based on a threshold which measures similarity of features between the original software and test software. When the similarity is larger than a given threshold, it is judged to be the same copyright, and when the calculated similarity is less than a given threshold, it is judged as a different copyright, shown in Fig. 3.

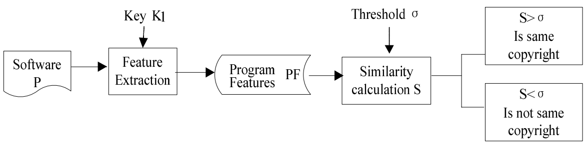


Fig.3. The copyright judgment based on threshold

A shortcoming of the scheme (Shortage (3)) is to depend on a threshold, if the experimental threshold is accurate, the result is correspondingly accurate, however if the threshold is not accurate or incorrect, certainly the result of the copyright identification has no credibility.

Due to diversity of software and dynamic behavior of a running program, how to identify and classify multi-attribute features of software based on class hierarchy of structured program, Fuzzy Classification in the field of data mining and pattern recognition provides a favorable support. Fuzzy division and fuzzy measure provide methods for analysis features from qualitatively to quantitatively. Supervised learning of classification can provide a comprehensive evaluation result for copyright identification. Classification-based identification has been applied in network intrusion detection, image analysis, medicine, biology and other fields and has achieved significant gains. Just thinking of the present software application, the recognition method based on the classification has been used in the software code detection[29],[30], analyzing the characteristics of malicious code and identifying malicious code through the support vector machine (SVM) , but as for the software copyright identification, there is still none of this kind of research at present.

Aiming at Shortages (2) and (3), another contribution of the paper is to apply classification methods to build a software copyright recognition model, to learn similarity of multi-attributes of software features, then use the learned rules to discriminate whether or not the unknown software is the same copyright in order to improve objectivity and accuracy of copyright identification instead of depending on a trusted third party or only one threshold.

III. DEFINITIONS AND FORMAL DESCRIPTION

In order to accurately understand and research the software copyright identification, software features, feature attributes, and so on, formal definitions are as follows:

A. *Features and Feature Attributes*

DEFINITION 1 (Software Features, SF):Refers to all the information about program P, including class hierarchy, opcode, API call frequency, thread execution sequence, control flow, data flow, space and time structure of software , and other series of information.

In accordance with software as a black box, after entering I, output O, so software features can be divided into Input Software Feature(ISF)， Output Software Feature(OSF) and Self Software Feature(SESF),That is $SF \leftarrow ISF \cup OSF \cup SESF$.

According to the number of types of extraction objects, dividing software features into single attribute features and multi-attribute features.

At present, most studies focus on the static and single-attribute features, in order to improve the anti-attack of features and the identification accuracy, the paper studies the dynamic and multi-attribute features.

DEFINITION 2 (Feature Attributes, FA)：Refers to the types of the extracted object, for example, source and API call belong to different feature types. For copyright identification, that is what kinds of judgment copyright conditions, that is $SF_P \leftarrow FA(a_1, a_2, \cdots, a_i)$ $(i \geq 1)$.

DEFINITION 3 (Single Attribute, SA):Refers to the feature type of extraction objects has only one or the identification condition is only one. That is $SF_P \leftarrow FA(a_1)$, like other studies take one of the dynamic opcode, API frequency, or class inheritance relationship as software feature.

DEFINITION4 (Multi-Attributes, MA): Many types of software features or the copyright judgment conditions are more than one type. That is, $SF_P \leftarrow FA(a_1, a_2, \cdots, a_i)$ $(i>1)$. These multiple attributes can come from different forms of software features, such as the source code, binary code, API features, tree-based features, graph-based features. And Multi-Attributes also include different forms of the same type, such as different forms of source-code [27]: programming layout, programming style and programming structure. The comprehensive performance of different types can be taken as software features or identification criteria.

B. *Software Transformation and Feature Measurement*

Attacks against software feature are mainly based on equivalence semantic transformations, such as optimizing software, obfuscation, encryption, shelling and so on, but software function is not changed, from the view of software features, equivalent semantic transformation is defined as follows:

DEFINITION 5 (Equivalent Semantic Transformation): Program P_i is a change of P , exists $ISF_P=ISF_{Pi}$ and $ISF_P=ISF_{Pi}$, but $SESF_P \neq SESF_{Pi}$, this change is called equivalent semantic transformation.In this paper P and P_i after equivalent semantic transformation are the same copyright.

Due to equivalent semantics transformation of software, it is difficult to distinguish between general features and birthmarks, so in the paper, using the word “ featrue” instead of “birthmark”. In fact, It is necessary to measure similarity of software by similarity of software features.

DEFINITION 6 (Software Feature Similarity, Sim): Similarity (P,Q) is defined as a similar degree of software P and Q, getting features description of P and Q by the same extraction algorithms: Extrace(P,I) → SF_P, Extrace(Q,I)→SF_Q when I=Φ, for static feature extraction; when I≠Φ, for dynamical feature extraction, thus the software Similarity (P,Q) is equivalent to the similarity of features Sim(SF_P,SF_Q)

$$\text{Similarity}(P,Q) \equiv \text{Sim}(SF_P,SF_Q) = \frac{|SF_P \cap SF_Q|}{|SF_P \cup SF_Q|} \in [0,1] \quad (1)$$

PROPERTY 1: The different program attack methods will lead to software and feature diversity, set program Q, P and P≠Q, under the same extraction method: Extrace(P,I)→SF_P, Extrace(Q,I)→SF_Q , then $SF_P \neq SF_Q$.

DEDUCTION1: From Property 1, set programs P and Q, under same input I and same extraction method, then if SF_P=SF_Q , then there will be P=Q.

PROPERTY2:When software Q and P are the same, that is, if P=Q, then Similarity(P,Q)=1 .

DEDUCTION2: From Property 2, if the software similarity isn`t 1, software P and Q must not be the same programs: Similarity(P,Q)≠1 ,then P≠Q.

PROPERTY3:When P and Q are completely different, then Similarity(P,Q)=0.

DEDUCTION3: From Property 3, if Similarity(P,Q) ≠0, must be $SF_P \cap SF_Q \neq \Phi$.

PROPERTY4: As a result of compiling, running and implementing , programs there are some similarities and software common characteristics, that is $SF_P \cap SF_Q \neq \Phi$, 0<Similarity<1 . And the more similar the software, the greater the number of feature fragments , the bigger of the value Similarity (p,q); the less similar the software is, the fewer the number of feature fragments, the smaller the value of Similarity (p, q).

C. Software Copyright Identification System Based On Single-Attributes

The identification process is based on threshold a single attribute of software features including: feature extraction, similarity computation and feature-based copyright identification, defined as follows::

DEFINITION 7 (Software Copyright Identification System based on Single Attribute, $SCIS_{SA}$): Given programs P and Q , input I, *Extract()* is a feature extraction method, feature similarity is expressed *Sim()*,$SCIS_{SA}$ shown as follows:

1) Extrace(P,I)→SF_P, Extrace(Q,I)→SF_Q ;
2) Sim(SF_P,SF_Q)∈ [0,1] ;
3) ∃μ, ξ, 0≤μ≤ξ≤1;
   a) When Sim(SF_P,SF_Q)≥ξ, P and Q belong to the same copyright;
   a) When μ≤Sim(SF_P,SF_Q)<ξ, cannot decide whether P and Q are the same copyright;
   b) When Sim(SF_P,SF_Q)≤μ, P and Q belong to the different copyright.

In $SCIS_{SA}$, how to decide μ and ξ will affect the robustness and credibility of copyright identification. Some researches [16] set μ and ξ for 0.2 and 0.8 in program plagiarism detection; some taking the instruction order as the birthmark features [22], using the K-gram results [24], μ and ξ are set for 0.6 and 0.8.

Setting μ and ξ is subject to the following factors: (1) type of extraction object; (2) size of fragments split; (3) calculation similarity method. Due to these effects, μ and ξ, in identification system of a single attribute, are difficult to draw and unify, and will affect the practicality of a single attribute judgment algorithm.

IV. COPYRIGHT IDENTIFICATION BASED ON CLASSIFYING MULTI-ATTRIBUTES

Due to the software diversity resulting from equivalent semantics transformation, for the purpose of increasing the accuracy by enhancing the dimension, using classification techniques in pattern recognition to identify unknown software which analyzes the changes of software multi-dimensional features and get the identification rules, instead of depending on a trusted third party or a threshold. The formal description about the algorithm and the realization are as follows.

A. Definition of Dynamic Multi-Attributes Identification System

DEFINITION 8 (Software Copyright Identification System based on Classifying Multi Attributes, $SCIS_{CMA}$):

Original program P_0 (need to be protected), programs $P_1,P_2\ldots P_i$ (Some programs from equivalence semantic transformations for P), and reference programs $Q_1,Q_2\ldots Q_j$ (with the same function, but belong to different copyright with P_0), the legitimate input I (I≠Φ:dynamic,I=Φ:static), $A_1,A_2\ldots A_k$ represent the k different attributes, C is a set of classification methods, the algorithm $SCIS_{CMA}$ as following:

1) Extaract(Px{$A_1,A_2\ldots A_k$,I})→{$SF_{Px\{A1\}}$, $SF_{Px\{A2\}}$ … $SF_{Px\{Ak\}}$ }, Px∈ { P_0, P_i , Q_j };
2) Sim($SF_{P0\{Ak\}}$,…,$SF_{Py\{Ak\}}$) → a_k,P_y ∈ {P_i,Q_j} a_k ∈ [0,1];
3) ∃ T[a,b]←{a_1, a_2,…,a_k ,class{yes, no}}, a=i+j, b=k+1;
4) ∀C_i∈ C,using a classified method C_i to learn the above data T[a,b], get the rules R_i, expressed as $\delta = \partial$ (T[a,b], R_i, C_i);
5) An unknown program X, circulating 1~4 steps, measure similarity on same k attributes with P0, get T[1,k+1] the value of the last attribute CLASS is null, using the rule Ri to identification X and P_0 whether or not is same copyright, output the value of CLASS, expressed as ∂^{-1} (T[1,k+1], R_i, C_i) → Class{yes:no}.

The detailed description and application of $SCIS_{CMA}$ is as follows.

B. Algorithm Description of $SCIS_{CMA}$

The process of $SCIS_{CMA}$ is divided into seven steps, as shown in Fig. 4, the outputs from different steps in order as follows: software fragments, data sets , two-dimension table with 1-0 values, learning rules and the judgement results, the implementation for key steps is described in next section, the description of $SCIS_{CMA}$ as follows:

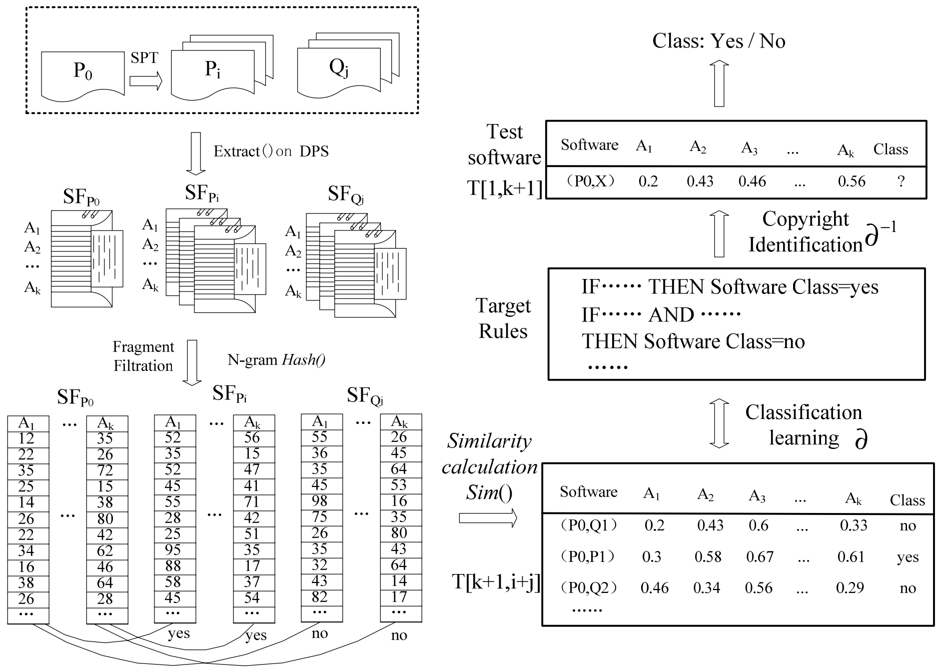


Fig.4. Software copyright identification based on multi- attribute features

1) **Software Transformation(SPT)**: The software P_0 is transformed to different software $P_1,P_2 \ldots P_i$, by equivalent semantic transformation such as confusion, optimization, compression etc using tools or simulation, that is P_i=SPT(P_0), P_i and P_0 belong to the same copyright. Reference software $Q_j=(q_1, q_2, \cdots, q_j)$ are different copyright from P_0 , but are the same function;
2) **Software Fragment collection (Extract())**: Based on dynamic program slicing (see IV.C (1)) ,software P_0, P_i and Q_j are sliced, tracked and collected respectively on the k different attributes, The fragments can be collected: Extract(P_i) $=\{SF_{P_i\{A_1\}}, SF_{P_i\{A_2\}}, \cdots SF_{P_i\{A_k\}}\}$;
3) **Fragment Filtration**: Pretreatment collected fragments: (1) removal of the features likely to be attacked [19]; (2) removal of the common characteristics [23], as possible to retain the personality characteristics;
4) **N-gram Feature Description(H)**: This step is the process of feature fragments into integer values (see IV.C (2)): *n* is an input parameter (sliding window size), dividing the collected fragment $SF_{Px\{Aj\}}$ into substrings in accordance with parameter *n*, build *hashes ()* indexes for each substring. (using string matching algorithm, Karp-Rabin [22],[23]), a substring is mapped to a corresponding integer value by hash function, can get integer value sets of the jth attribute of the software P_i ,that is $D_{P_i\{A_j\}} = H(SF_{P_i\{A_j\}})$;
5) **Similarity Calculation(Sim)**: Similarity calculation for obtaining a two-dimensional table between different attributes value set (see IV.C (3)), such as of $D_{Px\{A_j\}}$ of P_x, $D_{Q_y\{A_j\}}$ of Q_y and the corresponding attributes value set $D_{P0\{A_j\}}$ of the original program P_0;
6) **Classification Learning(∂)**: Classify training based on SVM(see IV.C (4)), input the two-dimensional table T[a,b], constructed classifier C_i, output classification rules R_i, that is $\delta = \partial$ (T[a,b], R_i, C_i);
7) Copyright Identification(∂^{-1}): Calculation similarity of K attributes between the unknow software X and the original software P0, get the table T[1,k+1] of the last attribute value is null, using the rule Ri gotten from classification learning to judge and identify the unknown software whether or not the same copyright, output the value of the last identity attribute CLASS, that is Class{yes:no}=∂^{-1} (T[1,k+1], R_i, C_i)).

C. Essential Algorithm Implementation of $SCIS_{CMA}$

(1) Software Fragment Collection

Extracting multi-attribute feature fragments based on dynamic program slicing.When entering one of legitimate inputs, I, slicing、tracking and collecting fragments of software P_0, P_i, Q_j. Obtaining fragment sets of four types of attributes: { A_{i1}, A_{i2}, A_{i3}, A_{i4}}= {FunctionName, Opecode, Bytecode, API}. The algorithm of dynamic program slicing is as follows:

1) *Construct slicing criterion C=<V，N，I> of software, I is a legitimate input, N is one of the program running points ,V a variable or set of variables;*
2) *According to control flow graph of program to structure control dependency graph (CDG) and data dependency graph(DDG);*
3) *Find all reachable nodes of the node n, when the input variable v ∈ V. Taking these nodes as a starting point to traverse PDG，In the ergodic process, correspondence sentences of all visited nodes are constituted to slice S;*
4) *Inputting I , to run the program and track execution states,*

can get the actual execution paths can be up to;

5) *Mapping in PDG and traversing the dependent edges,and marking all the traversed nodes, the sentences corresponding to the traversed nodes are the slices you want S' ($S' \subseteq S$);*
6) *Repeat steps 3-6, deleting duplicate nodes in S'.When the program execution path in line with the specified path, then getting the final slicing S'' ($S'' \subseteq S'$).*

(2)N-gram Feature Description

N-gram Feature Description is the process of transforming software fragments into the integer-value sets. Improving feature description algorithm of one-dimension attribute to the description of multi-attribution types, specifically expressed as follows:

N-gram（Fr_{Aij}, *N, W*）

INPUT：Fr_{Aij} *; //Fragment Set of the jth attribution of software* P_i*;*

N ; //is the length of each shingle;

W; //is the size of the winnowing window;

OUTPUT： *Set of integer values of the fragment attribute* D_{ij}*;*

PROCESS：

1) L← length[Fr_{Aij}] ;*; //calcuting the length of* Fr_{Aij}*;*
2) *Construct a list shinglesconsisting of L-N+1,* j∈ [1,L-N+1] , length[Y_j]=N*;*
3) Hashes(Y_j) *; // building hashes() indexes， Karp-Rabin [27] ;*
4) *Construct a length* |hashes|-W+1 *list windows by sweeping a size* W_j , j∈ [1,|hashes|-W+1] *;*
   a. *Let* D_{ij} ← Ø; *;*
   b. For each W_j , W_j ∈ windows do
   c. Min ← W_j *smallest hash value in the W;*
   d. *If Min wasn't already selected in the previous window then;*
   e. D_{ij} ← D_{ij} ∪ { Min };
5) Return Dij.

The three principles for feature description are as follows：

1) Comparing two fragments must be based on the same attribute type and same slice address;
2) For different feature attributes, different size of the sliding window. Need to do experiments to decide the size of the window for a higher accuracy;
3) Filtering fragment to retain as much of the atomic features as possible which are not easy to add or delete.

(3)Similarity Calculation

The similarity between two programs is measured by similarities of multi-attribute features. The existing problems of direct measurement [22] is that individual characteristics of software is not obvious and that common characteristics account for a large proportion in the fragment description, especially when some samples or description sets are relatively small. In order to highlight the individual characteristics and reduce the impact of common characteristics, to fuzzily measure the common attribute description of software features:(Common Attributes,CA), each software attributes description (SFp{A}) equals all fragment descriptions (SFrp{A}) of the software minus the common attribute description (SF_{CA})of software features, that is SFp{A}=SFrp{A}-SF_{CA}. For multi-attributes measurement, the metric result of the similarity between two software is not only one [0,1]-value, but rather a set of values. Metric formula is as follows:

$$\begin{aligned}\mathrm{Sim}(SF_p,SF_Q) &=\{\mathrm{sim}(SF_{p\{A\}}, SF_{Q\{A\}})\}\\ &=\{\mathrm{sim}(SF_{p\{Ai\}}, SF_{Q\{Ai\}})\}\\ &=\{\mathrm{sim}((SFr_{p\{Ai\}}-SF_{GA}), (SFr_{Q\{Ai\}}-SF_{GA}) \}\\ &=\frac{|(SFr_{P\{Ai\}}-SF_{GA})\cap(SFr_{Q\{Ai\}}-SF_{GA})|}{|(SFr_{P\{Ai\}}-SF_{GA})\cup(SFr_{Q\{Ai\}}-SF_{GA})|}\end{aligned} \quad (2)$$

How to select CA is very important,but it is difficult to accurately describe, only to weaken conditions, and fuzzily descript, such as CA expressed for the initialization codes resulting from the same function library or editor.

TABLE II

(A)THE TRAIN DATA OF MULTI-ATTRIBUTE SIMILARITIES

Relation: softwareCharacter_train

No.	software Nominal	Function-name Numeric	OpeCode Numeric	Bytecode Numeric	API Numeric	IF-Software Nominal
1	(P0.Q1)	0.26	0.4	0.6	0.33	no
2	(P0.Q2)	0.43	0.34	0.56	0.28	no
3	(P0.P1)	0.3	0.58	0.64	0.48	yes
4	(P0.P2)	0.53	0.33	0.67	0.61	yes
5	(P0.P3)	0.83	0.52	0.71	0.46	yes
6	(P0.Q3)	0.34	0.42	0.16	0.36	no
7	(P0.P4)	0.63	0.45	0.37	0.4	yes
8	(P0.Q4)	0.4	0.36	0.72	0.37	no
9	(P0.P5)	0.26	0.81	0.64	0.62	yes
10	(P0.P6)	0.45	0.61	0.24	0.39	yes
11	(P0.P7)	0.34	0.42	0.84	0.55	yes
12	(P0.P8)	0.52	0.23	0.91	0.64	yes
13	(P0.P9)	0.61	0.57	0.35	0.49	yes
14	(P0.Q5)	0.46	0.34	0.76	0.29	no

(B) TEST DATA OF UNKNOWN SOFTWARE

softwareCharacter_new.arff

Relation: softwareCharacter

No.	Function-name Numeric	OpeCode Numeric	Bytecode Numeric	API Numeric	IF-Software Nominal
1	0.45	0.6	0.42	0.57	

The result of similarity calculation is a two-dimension table with the similarity between(P_0,P_i) and (P_0,Q_i) with four numeric attributes and one nominal attribute (class label). The training data and test data shown in Table II and the .arff file of two-dimension table for classification analysis.

(4) Classification Learning based on SVM

Data classification is a supervised learning to build classiifer by similarity calculation of multi-attributions. A tuple, X, is represented by an n-dimensional attribute vector X={ $x_1, x_2, \cdots, x_n$ }, dipicting n measurements made on the tuple from n software attributes, respectively A_1, A_2, ⋯ ,A_n,. Each tuple, X is assumed to belong to a predefined class, though classification algorithm C_i builds classifier and classification rule R_i analyzing or "learning" from a training set made up of data tuples and their associated class labels(yes/no same copyright). If the accuracy of the classification rule is acceptable, then using the rule on new data tuple, shown in Fig.5.

When constructing classifiers, SVM[30] is adopted for training learning. Every sample is an n-dimensional vector. Find the optimal partitioning plane w.x+b=0 between the two types. The ith classifier decision function is f(x). When training, adjust the size of slide window, and optimize the objective function according to

$$\max W(a_i)=\sum_{i=1}^{m} a - \frac{1}{2}\sum a_i a_j y_i y_j k(x_i, x_j) \quad (3)$$

With constraint $\sum_{i=1}^{m} a_i y_i = 0$, Lagrange multiplier $a_i \in [0,C]$, where C is a positive constant. Decision function is defined as

$$f(x) = \mathrm{sgn}\,(w \cdot x+b)=\mathrm{sgn}\,(\sum_{i=1}^{m} a_i y_i k(x_i, x) + b) \quad (4)$$

The kernel function $k(x_i, x_j)$ satisfies Mercer condition, as an inner product of some transformed space [12]. In training, use the radial base kernel function $k(x_i, x) = \exp(-\frac{|x_i - x|^2}{\delta^2})$, where δ is an argument. Use quadratic programming to solve equation (3), to find optimal Lagrange coefficient a; threshold b is found by solving equation (4).

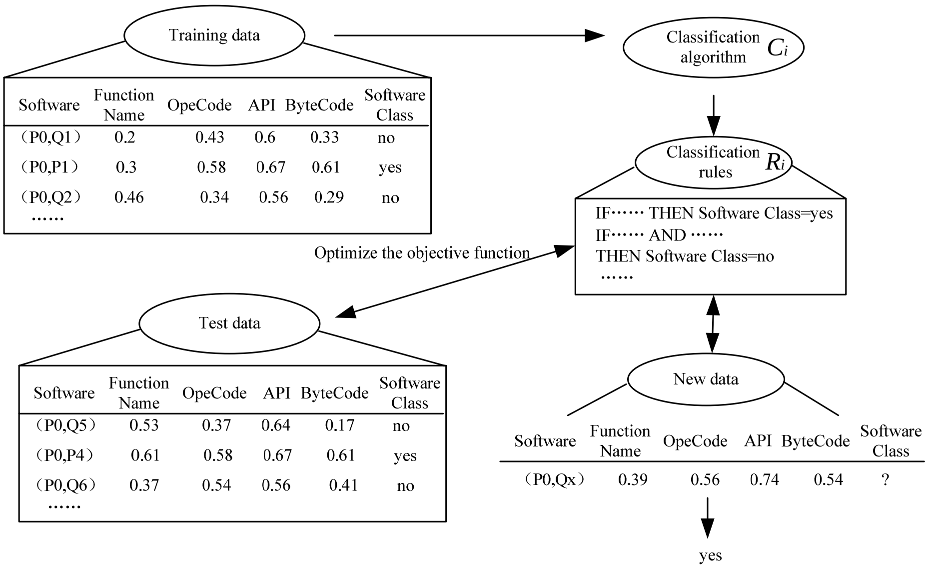


Fig.5. classification process of multi-attribute features

V. EXPERIMENT AND ANALYSIS

For single attribute algorithms, the evaluation of software features and feature extraction system is mainly from robustness and credibility [18]; for the multi-attribute algorithm, $SCIS_{CMA}$, besides the above, need assessments from classification correct rate and accuracy.

DEFINITION 9 (Robustness): Set programs p changed into p' by equivalent semantic transformation, then the similarity between p and p' need to be as large as possible. Feature extraction from p and p' by different algorithms $Extract_1$ and $Extract_2$, respectively, get the similarity of two software features Sim_1(SFp,SFp') and Sim_2(SFp,SFp'). If Sim_1(SFp,SFp')>Sim_2(SFp,SFp'), the algorithm $Extract_1$ is considered to have better robustness.

DEFINITION10(Credibility)：Set two different programs p and q, have the same function, then the similarity between p and q need to be as small as possible. To extract feature by different algorithms $Extract_1$ and $Extract_2$ respectively, and get the similarity of two software features Sim_1(SFp,SFq) and Sim_2(SFp,SFq). If Sim_1(SFp,SFq) < Sim_2(SFp,SFq), the algorithm $Extract_1$ is considered to have higher credibility.

DEFINITION11 (Correct Rate): Correct Rate of a classifier C_i on given test set is the percentage of test set tuples that are correctly classified by the classifier, expressed as Corr(C_i) or 1-Error(C_i) , Error(C_i) is incorrect rate or misclassification rate of a classifier . When training for the small-sample and numerical data sets of multi-attributes, using *Stratified K-Fold Cross-Validation*，that is the overall number of correct classifications from the *K* iterations ,divided by the total number of tuples in the initial data.

DEFINITION12(Accuracy): The accuracy of a classifier can be measured by *Sensitivity, Specificity* and *Precision*. *Sensitivity* is also referred to as the *true positive*(recognition) rate, (that is , the proportion of positive tuples that are correctly identified) ; While *Specificity* is the *true negative rate* (that is , the proportion of negative tuples that are correctly identified); *Precision* is the percentage of positive tuples. These measures are defined as

$$\text{Sensitivity} = \frac{t\_pos}{pos} \quad (5)$$

$$\text{Specificity} = \frac{t\_neg}{neg} \quad (6)$$

$$\text{Precision} = \frac{t\_pos}{(t\_pos + f\_pos)} \quad (7)$$

These datas come from a confusion matrix of the positive and negative tuples, as shown in Table III.

TABLE III CONFUSION MATRIX

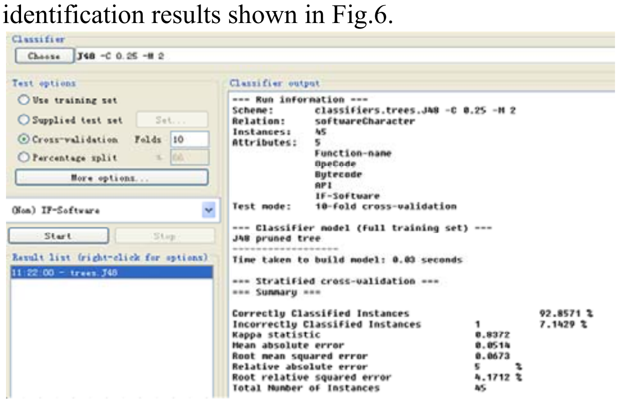
Actual class	Predicted class: C1	Predicted class: C2
C1	true_positives	false_negatives
C2	false_positives	true_negatives

Besides confusion matrix, the real rate and sensitivity can also be measured by ROC(receiver operating characteristic) curve，that shows the trade-off between the true positive rate or sensitivity(proportion of positive tuples that are correctly identified) and the false-positive rate (proportion of negative tuples that are incorrectly identified as positive)for a given model. The area under the ROC curve is a measure of the accuracy of the model, the closer the area is to 0.5, the less accurate the corresponding model is .A model with perfect accuracy will have an area of 1.0 .

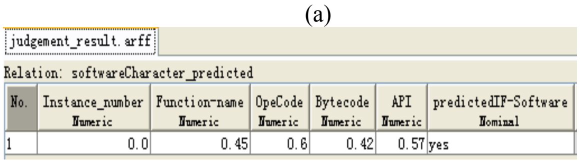
In addition to the above, the classification time is one of the measured criteria.

A. *Experimental Statistical Result of Multi-Attribute Identification Algorithm*

Because the analyzed data is of 0-1 numeric type, the choice for classification algorithm is limited. Application of existing classification algorithms, and taking into account the numeric types and classification efficiency. In the experiment using the decision tree algorithm J4.8（0.25,2）. Confidence factor for pruning is 0.25 （less than this value to be trimmed）,the minimum number of instances is 2. The test environment is as follows: Windows XP; Operating System: Intel (R) Core (TM)2 Duo CPU E8400 3.0GHZ; Memory:3.25G ， the classification study statistical results and output identification results shown in Fig.6.



(a)



judgement_result.arff

Relation: softwareCharacter_predicted

No.	Instance_number Numeric	Function-name Numeric	OpeCode Numeric	Bytecode Numeric	API Numeric	predictedIF-Software Nominal
1	0.0	0.45	0.6	0.42	0.57	yes

(b)

Fig. 6 Interface of Classifier Learning (a) and Output Identification Result (b)

The algorithm of multi-attribute identification ($SCIS_{CMA}$) relys on the classification learning strategies, judging unknown software whether or not it is the same copyright from comprehensive aspects of different attributes. Instead of depending on the third parties to save the extracted features, or only one threshold to determine, and enhances the identification objectivity.

B. *Comparison of Robustness*

Anti-attack simulation comparison between single-attribute features and multi-attribute features under different attacks. Testing similarity changes on single-attribute algorithms : $ssSWA_{window}$ in [16], TNMMsmc in [19] , Mckgram in [24] and multi-attribute algorithms $SCIS_{CMA}$ under three kinds of attacks: rename confusion, adding useless predicates (dead code) ,the instruction confusion on the same program *fibonacci*, the results as Table IV.

TABLE IV
SIMILARITY CHANGES OF THE DIFFERENT ALGORITHMS IN DIFFERENT ATTACKS

algorithm	$ssSWA_{window}$	TNMMsmc	MCkgram	$SCIS_{CMA}$
Extraction object	class name in source	call order of API	opcode set	multi-attribute /their combined features
Fragment size (Byte)	6546	3500	2980	11026
Types of attacks	Similarity changes after attack（%）			
Rename confusion	0.9%	17.3%	72.6%	30.2%
Adding dead code	96.1%	0.55%	1.31%	32.7%
Instruction confusion	87.1%	63.2%	0.64%	50.31%

Seen from the table, different kinds of attacks have a greater impact on the different objects, such as renaming confusion to the source, adding dead code to the API call order, instruction confusion to opcode set. Therefore, if we rely on one kind of features to identify the software copyright, when it meets a targeted attack, its robustness is obviously poor; The reference change of the $SCIS_{CMA}$ in the Table 4, is an average value of similarity of various features, due to its features is the combined changes of different software and different attributes, it appears general against a specific attack, but has stronger robustness and stability against the different attacks.

C. *Comparison of Credibility*

Comparing credibility between two program P and Q , with the same function but different copyright. In the experiment, using the different visions about program Fibonacc , Fig. 7（a）Fibonacci function written in java,（b）is a confused program of (a) , defined as the same copyright with (a), (c) is the same function as (a), but another different version written in another language.

```
int  fib (int n) {
    if (n<=1)
      return  n;
    else
      return  fib(n-1)+fib(n-2) ;
}
```

(a)

```
public long fibonacci (long n)
{
 if(n==0||n==1)
  return n;
 else
  return fibonacci(n-1)+fibonacci(n-2);
}
```

(b)

```
#include<stdio.h>
int main()
{
    int f1 = 1;   int f2 = 1;
    int sum = 0;   int n;
    scanf("%d",&n);
    int i;    int temp;
       if (n == 1)
             {  sum = f1;
             }
       else   if (n == 2)
             { sum = f1+f2;
             }
           else
             { sum = f1+f2;
               for (i=2;i<n;i++)
                 { temp = f1+f2;
                   sum += temp;
                   f1 = f2;
                   f2 = temp;
                 }
             }
       printf("sum = %d",sum);
       return 0;
}
```

(c)

Fig.7.The programs Fibonacci with same and different copyrights

To extract fragments in IDA using algorithm $ssSWA_{window}$ in [16], Mckgram in [24] and multi-attribution algorithm, $SCIS_{CMA}$, , then to fuzzily measure. The similarity results by comparison as follows Fig.8.

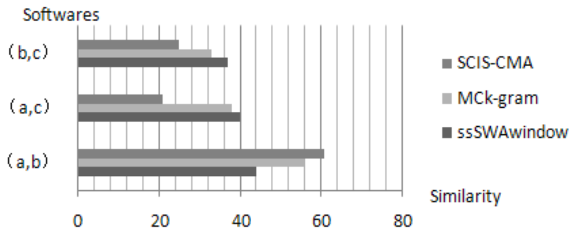


Fig.8 Comparison of Crediblity in different algorithms

Seen from the experiment, the similarity of multi-attribute features, $SCIS_{CMA}$, is the smallest in the different versions programs :(a, c),and (b, c) , and is the biggest in the same version of programs:(a, b) . From the credibility definition, the credibility of $SCIS_{CMA}$ is comparatively higher. The reason is analyzed because using the different feature types, using various collection points make the collected feature fragment increase the number and representatives, so the higher credibility than the other two algorithms.

D. *Experimental Analysis of the Correct Rate*

Using the Decision Tree j48 (confidence factor c=0.25, the minimum number of instances m=2, trim algorithm: C4.5) and Support Vector Machine, SVM_SMO to classify (supervised learning) the experimental data in the two-dimensional table like in Table 2, testing the correct rate separately based on the training set and 10-fold cross study, the datum in the table come from the output of the experimental correctness rate in Fig.6, shown in Table V.

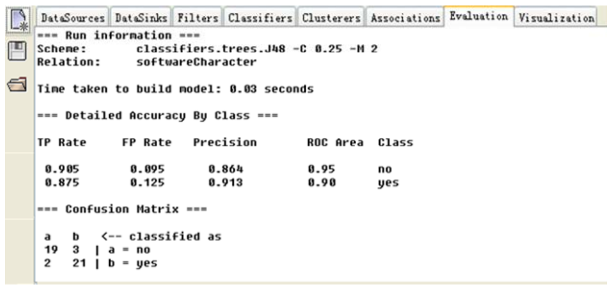
TABLE V
COMPARISON OF CLASSIFICATION CORRECT RATE

Test programs	Correct rate of classification algorithms			
	Decision Tree J48		SVM-SMO	
	Using training set	Repeated Cross-validation	Using training set	Repeated Cross-validation
Hello.exe	92.8571%	81.3487%	83.4210%	79.5421%
Fibonacci.exe	93.6010%	84.6351%	85.0725%	82.6438%
Md5.exe	95.6248%	86.3157%	87.6425%	84.3572%
Notepad.exe	97.2145%	88.9425%	88.6274%	85.3475%
Ping.exe	96.5413%	88.0243%	88.0364%	86.4225%

Seen from the table, the classification correct rates of the two algorithms are higher ,and the correct rates on 10-fold cross training is smaller than the results on the training set. Analyzing reason is that the correct rate on the old data set (training set) can not be fully representative of the correct rate on the new data (unknown software). The classification by 10-foldcross study is more stringent for small sample and limited data, so the accuracy is generally smaller than the results on the training set.

E. *Experimental Analysis of the Accuracy*

In order to verify the accuracy of classification models, on the platform of Weka3.5.8, using classification algorithm J48 (c 0.25, m2) ,to measure the sensitivity, the specificity and the precision by the confusion matrix,shown in Fig.9.



```
DataSources | DataSinks | Filters | Classifiers | Clusterers | Associations | Evaluation | Visualization
=== Run information ===
Scheme:       classifiers.trees.J48 -C 0.25 -M 2
Relation:     softwareCharacter

Time taken to build model: 0.03 seconds

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision     ROC Area    Class

 0.905      0.095      0.864         0.95        no
 0.875      0.125      0.913         0.90        yes

=== Confusion Matrix ===

 a   b   <-- classified as
 19  3  | a = no
 2   21 | b = yes
```

Fig.9 Accuracy Test on the classification model

From confusion matrix, the sensitivity of the model can be calculated for 0.905, the specificity is 0.875， the precision respectively is no:0.95, yes: 0.90, the ROC area is no: 0.95, yes: 0.90. Relatively higher accuracy can be seen, so the model can be used to identify the unknown software. Comparison of ROC area between the different number of tuples (95,45,15) and the classic numerical iris data , as follows Fig. 10.

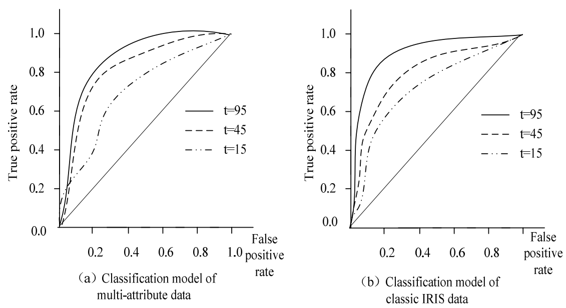


Fig.10 The ROC curves of different tuples

The greater number of test tuples t, the farther away from the diagonal line (reference line) of the ROC curve in Fig.9 ,and the higher accuracy of the model. Copmare with the classic IRIS data, the changes in the ROC curve have the same trend, therefore, it should maximize the number of tuples as possible in the experiment that depends on changes of the software by the actual software transformation tools and existing software attack methods, so you can increase the number of tuples by studying the relevance of the different attribution features of dynamic software to simulate software diversity.

VI. CONCLUSION AND OUTLOOK

Software features and zero-watermarking identification scheme are researched to avoid the embedded watermarks degradation of software performance. In order to improve the accuracy of the copyright identification and enhance the attack resistance of software features, this paper proposed a software copyright identification scheme based on classification learning multi-dimensional features, that classification learning in pattern recognition is used in copyright identification and software security with obvious advantages in improving the judgment accuracy and robustness, specifically including: (1)Improving the extraction object from only one feature to multi-attribution feature to enhance the robustness of features; (2)Making judgments method more objective, as a result of identification based on classified learning rules, not the sole threshold value; (3)Improving from non-blind judgment to blind judgment, removing reliance on third-party, to make the judgment more secure.

Prospects and plans: (1)To introduce rough set theory to analyze the relevance of the different attribution features. (2)Classification of a small sample data depending on choice of classification algorithm and further development of the classification algorithm, *bagging* (multiple combination forecast model strategies) and *adaboost* (complementary classifier based on weight) can be used to reduce dependency on the classification algorithm to further improve accuracy.

On application, the scheme is suitable not only for software recognition, but also for images, databases, and document discrimination. Meanwhile it provided a new method which decentralizes detection points for checking virus and software confusion assessment, and give full consideration to the diversity of software and other carriers.

ACKNOWLEDGMENT

This work was supported in part by the Scientific Research Program Funded by Shaanxi Province Soft Science Research Project (2012KRM84); in part by Ministry of Education Computer Basic Teaching Reform Project (JZW201128); in part by Shaanxi Provincial Education Department (2013JK1200); in part by 13xcj05 of University of Finance & Economics, Xi′an,China.

REFERENCES

[1] BSA. " IDC. The Eighth Annual BSA Global Software Piraey Study [EB/OL]." 2012; http://portal.bsa.org/globalpiracy2010/.

[2] Z. Xiaoqiang, Z.Guiliang, W. Weiping, "New Public-Key Cryptosystem Based on Two-Dimension DLP," *Journal of computers*, vol.7, no.1,pp. 169-178, 2012.

[3] Z. Yu, C. Wang, C. Thomborson, J. Wang, S. Lian, and A. V. Vasilakos, "A novel watermarking method for software protection in the cloud," *Software: Practice and Experience*, 2010, vol.42, no.4 :pp. 409-430.

[4] A. Rial, J. Balasch, and B. Preneel, "A Privacy-Preserving Buyer-Seller Watermarking Protocol Based on Priced Oblivious Transfer," I*nformation Forensics and Security, IEEE Transactions on*, vol.6, no.1, pp. 202-212, 2011.

[5] Y. Mahmood, S. Sarwar, Z. Pervez, and H. F. Ahmed, "Method based static software birthmarks: A new approach to derogate software piracy." *IEEE 2nd International Conference*. 2009: pp. 1-6.

[6] C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu, C. Linn, and M. Stepp, "Dynamic path-based software watermarking," *ACM Sigplan Notices*, vol. 39, no. 6, pp. 107-118, 2004.

[7] J. Nagra, and C. Thomborson, "Threading software watermarks." *Information Hiding. Springer Berlin Heidelberg*, 2005: pp. 208-223.

[8] J. Zhu, Y. Liu, A. Wang, and K. Yin, "H Function based Tamper-proofing Software Watermarking Scheme," *Journal of Software*, vol. 6, no. 1, pp. 148-155, 2011.

[9] Yangxia Luo, Dingyi Fang. "Dynamic watermarking algorithm based on chaotic optimization". *Chinese Jorunal of University of Science and Technology*, 2012.42(1):77-84;

[10] " Sandmark System ," http://sandmark.cs.arizona.edu, .

[11] C. Collberg and J. Nagra, "Surreptitious software: Obfuscation, Watermarking and Tamperproofing for Software Protection," *Addison-Wesley Professional*, 2010.

[12] X. Li, and G. He, "A new audio zero-watermark algorithm for copyright protection based on audio segmentation and wavelet coefficients mapping." IEEE *Digital Content, Multimedia Technology and its Applications,*2011: pp. 22-26.

[13] Z. Li, and X. W. Chen, "Adaptively imperceptible video watermarking algorithm using entropy model," *Ruan Jian Xue Bao(Journal of Software),* vol. 21, no. 7, pp. 1692-1703, 2010.

[14] M. Yingjie, G. Liming, L. Mingwen. " Chinese Text Zero-Watermark Based on Three-dimensional Space Model." *Journal of Computers,* Vol 7, No 8, 2063-2070, 2012.

[15] Y. J Meng,Y. Shi,L Si and J. Y Zhang, "Zero-watermarking algorithm for database based on chaotic sequences," *Chinese Computer Engineering and Applications,* vol. 44, no. 29, pp. 168-170, 2008.

[16] A. Aiken, S. Schleimer, J. Auslander, D. Wilkerson, A. Tomasic, and S. Fink, "Method and apparatus for indexing document content and content comparison with World Wide Web search service," *US Patents*, 2004.

[17] A. Aiken, "Moss: A system for detecting software plagiarism," University of California–Berkeley. http:// www. cs. berkeley. edu/aiken/moss. html, 2005.

[18] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, "Design and evaluation of birthmarks for detecting theft of java programs." *Software Engineering (IASTED SE 2004).* 2004: pp. 569-575.

[19] G. Myles, and C. Collberg, "Detecting software theft via

whole program path birthmarks," *Information Security,* pp. 404-415, 2004.
[20] H. Tamada, K. Okamoto, M. Nakamura, A. Monden, and K. Matsumoto, "Dynamic software birthmarks to detect the theft of windows applications." *International Symposium on Future Software Technology*. Vol. 20. No. 22. 2004
[21] H. Park, H. Lim, S. Choi, and T. Han, "Detecting Common Modules in Java Packages Based on Static Object Trace Birthmark," *The Computer Journal*, vol. 54, no. 1, pp. 108-124, 2011.
[22] Y. Bai, X. Sun, G. Sun, X. Deng, and X. Zhou, "Dynamic k-gram based software birthmark." *IEEE ASWEC 2008. 19th Australian Conference.* 2008: 644-649.
[23] X. Xie, F. Liu, B. Lu, and L. Chen, "A software birthmark based on weighted k-gram." *Intelligent Computing and Intelligent Systems (ICIS),*IEEE, 2010:pp. 400-405.
[24] G. Myles, and C. Collberg, "K-gram based software birthmarks." *Proceedings of the 2005 ACM symposium on Applied computing. ACM,* 2005: pp. 314-318.
[25] H. Lim, H. Park, S. Choi, and T. Han, "A static java birthmark based on control flow edges." *Computer Software and Applications Conference, 33rd Annual IEEE International.* 2009, 1: pp. 413-420.
[26] X. Zhou, X. Sun, G. Sun, and Y. Yang, "A combined static and dynamic software birthmark based on component dependence graph." *Intelligent Information Hiding and Multimedia Signal Processing,* IEEE, 2008: pp. 1416-1421.
[27] YJ Zhao, ZY Tang, N Wang, DY Fang, YX Gu, "Evaluation of code obfuscating transformation." Chinese *Journal of Software*, 2012,23(3):700−711.
[28] Z. Xin, H. Chen, X. Wang, P. Liu, S. Zhu, B. Mao, and L. Xie, "Replacement attacks on behavior based software birthmark," *Information Security,* pp. 1-16, 2011.
[29] DG Kong, XB Tan, HS XI et al."Obfuscated Malware Detection Based on Boosting Multilevel Features." *Chinese Journal of Software*, vol.22, no.3, pp. 522−533, 2011.
[30] Z. Junpeng, H. Jianfeng,M. Lei, L.Jiuyong . "ORPSW: a new classifier for gene expression data based on optimal risk and preventive patterns," *Journal of computers*, vol. 6, no. 6, pp. 1191-1197, 2011.

Ye Guo born in 1961. Master instructor at school of information, Xi'an University Finance and Economics, China. Her main research interests include data mining and system security.

Mingyu Wang, born in 1963. Associate Professor, PhD degree, His main research interests include theories and methods of decision-making, optimize.

Yangxia Luo born in 1974, PhD degree. Her current research interests include software watermarking, computer security and data mining.