2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering

Fast Comparison of Software Birthmarks for Detecting the Theft with the Search Engine

Jun Nakamura†, Haruaki Tamada†,
† Faculty of Computer Science and Engineering, Kyoto Sangyo University.

***Abstract*—The software birthmarks were proposed for detecting the software theft, which is suspected a copy. The birthmark system extracts native characteristics of a program. Then, the system computes the similarity between two birthmarks. This paper proposes the reduction method of the comparison time for the birthmark system. The conventional birthmark systems did not care the comparison time. Therefore, the enormous amount of programs could not compare practically for requiring quite a huge comparison time. To solve the problem, we tackled to reduce the comparison time with the search engine. So that we introduce the pre-comparison phase into conventional birthmark procedures to narrow the defendant programs. We implemented the proposed method with Apache Solr search engine and evaluated the method. The result was shown that the method for UC birthmark dramatically reduces the comparison time 81.35%, and narrows the target programs 74.81%. On the other hand, the results were not good in the default threshold $\varepsilon = 0.75$ (high false negative rates $C_{\mathrm{FN}} = 67.01\%$, and low accuracies $M_a = 34.18\%$). However, the results were improved $C_{\mathrm{FN}} = 21.17\%$ and $M_a = 78.56\%$ in average by lowering $\varepsilon = 0.25$.**

I. Introduction

Today, software theft has become a serious problem with BSA global software survey 2016[1]. In order to solve software theft, the software birthmarks were proposed by Tamada et al[1], [2]. The birthmark system extracts native characteristics as birthmarks from binary code. Generally, the stealing is concealed, and the source codes of stolen programs are never exposed. Therefore, birthmarks are proposed for comparing two binaries.

The typical scenario of birthmark is we prepare a plaintiff binary program, and many defendants binary programs. We extract birthmarks from prepared programs. The extracted plaintiff birthmarks are compared to defendant birthmarks, computes similarities. Detected similar programs are, in other words, suspected copied programs.

Birthmarks are good concepts for detecting the theft, however, birthmarks have several problems. One of the problems is that practical use of birthmarks was not discussed. Considering the practical use, birthmarks require fast comparison for scaling up to the huge defendants. The previous birthmarks were focused on the detection accuracy of the suspected copies. This paper focuses on the comparison speed for scaling up defendants' programs.

Tsuzaki et al. proposed to reduce the comparison time of birthmarks with fuzzy hashing[3]. The key idea of their method is employed roughly comparison of fuzzy hashed birthmark information. The background of their idea supposes that the majority of programs are innocent in the real world. Therefore, the rough comparison is enough to narrow defendants. Narrowing the defendants enables to compare the more large scale defendants. This paper shares the background and employs rough comparison method to narrow the defendants. To reduce the comparison time of birthmarks, we introduce the new phase, called pre-comparison phase, between the extraction phase, and comparison phase in the previous birthmark system. The most important purpose of pre-comparison phase is to exclude a clearly innocent program from defendants. The proposed method accepts false positives (incorrectly detected innocent program) since further investigation will be conducted in comparison phase.

The proposed method has several nice side-effects. The one of effect is to define the request and response format. The previous birthmark systems did not define the request and response formats. We employ the search engine in the pre-comparison phase. By using the search engine we post a request and obtain a response. The request and the response formats are defined in the search engine. Then, another effect is easy to automate of the birthmark system. The users of the previous birthmark systems assume human since the request and the response format were not defined. The clients of the search engine allow not human, for example, a bot, another system, and etc. That is, the proposed method can be performed from other services.

The rest of this paper organized as follows. Section II describes the proposed method. Section III shows the experimental evaluation of the proposed method. Finally, section IV summarizes the paper and shows future works.

II. The Proposed Method

A. The Definition of Software Birthmarks

Before describing the proposed method, we explain the definition of software birthmarks. The software birthmarks are defined as follows by Tamada et al[1], [2].

Definition 1 (Birthmark): Let p and q be given programs. Further, let $f(p)$ be a set of characteristics extracted from p by a given method f. If the conditions below are met, $\mathcal{B}(p) = f(p)$ is said to be a birthmark of p.

Condition 1: $f(p)$ is obtained from only program p.

Condition 2: If q is a copy of p, then $\mathcal{B}(p) = \mathcal{B}(q)$.

Condition 1 indicates that a birthmark is an information necessary for running p and is not additional information. In other words, birthmark does not require additional information

[1]http://globalstudy.bsa.org/2016/downloads/studies/BSA_GSS_US.pdf

978-1-5090-4871-7/16 $31.00 © 2016 IEEE
DOI 10.1109/ACIT-CSII-BCD.2016.38

in the method of a software watermark. Condition 2 indicates that same birthmark can be obtained from a copied program. If birthmarks $\mathcal{B}(p)$ and $\mathcal{B}(q)$ are different, this means that q is not a copy of p.

Two properties known as resilience and credibility should also ideally be satisfied.

Property 1 (Resilience): For a p' obtained by an arbitrary equivalent transformation of p, $\mathcal{B}(p) = \mathcal{B}(p')$ is satisfied.

Property 2 (Credibility): When programs p and q that develop independently, $\mathcal{B}(p) \neq \mathcal{B}(q)$ is satisfied.

Resilience property indicates a resistance of birthmark to various types of attacks. Credibility property indicates that programs created completely independently can be differentiated even if their specifications are the same. Because birthmarks that completely satisfy both these properties are difficult to create, in practice, birthmark strength must be set appropriately at the discretion of the user.

The different types of birthmarks are proposed, which focuses on different characteristics of the program, such as control flow[4], opcode sequences[5], structures[6], data flow[7], behaviors[8], and etc.

Additionally, similarity calculation methods were defined for almost birthmarks. The paper did not discuss each similarity calculation method. Therefore, the paper simply shows $\mathrm{sim}(\mathcal{B}(p), \mathcal{B}(q))$, which computes the similarity between $\mathcal{B}(p)$ and $\mathcal{B}(q)$.

The similarity value for 0 means completely independent programs, and the value for 1 means the other programs are strongly suspected a copy. We should define the value of ε to decide the plagiarism or not. If the similarity of $\mathcal{B}(p)$ and $\mathcal{B}(q)$ is greater than ε, then either p or q is plagiarized from the other. A typical value of ε is 0.75[9]. We classify the score from sim into three groups to clarify the result from similarity, shows in equation (1)[10].

$$\mathrm{sim}(\mathcal{B}(p), \mathcal{B}(q)) = \begin{cases} \geq \varepsilon & \textit{copy relation} \\ \leq 1 - \varepsilon & \textit{no copy relation} \\ \textit{otherwise} & \textit{inconclusive} \end{cases} \quad (1)$$

B. Overview of the Proposed Method

The aim of the software birthmark is to detect the software theft, not to prove the theft. Therefore, the birthmark system should find similar programs from the enormous program database as the suspected copies. Then, we will conduct the further investigation to the resultant programs. The previous researches focus on the correctness of the birthmark system. The paper focuses on the speed of the birthmark system process. Because the scale of current birthmarks is a million programs at the most. In the real world, the quite large number of programs exist, and the programs are released day by day. Therefore, to reduce the time for birthmark system is helpful for the practical use.

The procedure of the birthmark system consists of extraction phase and comparison phase. In the typical scenario, the user of the birthmark system posts plaintiff program for detecting the suspected copies. The birthmark system extracts birthmark from plaintiff program and defendant programs. The defendant programs are stored in the database of the system beforehand. Next, the system computes similarities between every pair of extracted birthmarks in the comparison phase.

The system consumes time in both of extraction phase and comparison phase. To reduce the required time for the system, we focus on extraction phase. Fortunately, we can perform the extraction phase beforehand, and extracted birthmark can be stored in a database. Also, we improve the comparison phase to reduce comparison time with a search engine.

C. The Key Idea

The search engine finds the results from enormous data sets. We consider applying the search engine for the comparison phase of the birthmark system. However, the search algorithms in the search engine are different from each birthmark comparison algorithm. Therefore, we introduce new phase, pre-comparison phase, before comparison phase. The pre-comparison phase compares birthmarks with the simple algorithm to narrow down the defendants. The results from pre-comparison phase categorized into suspected copies are still defendant. Further investigations are conducted in the comparison phase. On the other hand, the innocent programs from pre-comparison become innocent, no further investigations are conducted. Because of almost programs in the world almost innocent. The proposed method aims to remove obviously innocent program early from the defendant sets.

Besides, false negatives (undetected copies) are serious problems than false positives (incorrectly detected innocent program). Because false positives will be investigated further method, however, false negatives are never investigated. We should consider the matter to implement the pre-comparison phase.

D. Our Pre-Comparison Phase with Search Engine

In the use of the search engine, we perform a registration step and search step. The registration step stores the extracted birthmarks into the database of the search engine. The search step accepts a request from a user and finds similar data from the database.

In order to perform the pre-comparison with a search engine, we must prepare the database to store the extracted birthmarks. For this, we collect original programs from some software repository. Then, extract birthmarks from them, and store the extracted birthmarks into the database. Recently, software repository is available in each programing language, for example, maven repository[2] for Java, and rubygems[3] for Ruby. Those processes are the registration step, shown in Figure 1.

Figure 2 shows the processes of search step. In the typical use case, a user posts a defendant program p to our system. Next, the system extracts the birthmark from the posted program, and obtain extracted birthmark $\mathcal{B}(p)$. Then, the system post $\mathcal{B}(p)$ to the search engine to find similar birthmarks. The

[2]https://mvnrepository.com
[3]https://rubygems.org

resultant birthmarks of the search engine are the result of the system.

E. Analysis of the Proposed Method

The section formulates the proposed method. At first, we define the scenario of the proposed method and the previous birthmark system.

1) Target Set: Let $P = \{p_1, p_2, ..., p_n\}$ be given programs for the proposed method. Then, let $S_i = \{\mathcal{B}_i(p_1), \mathcal{B}_i(p_2), ..., \mathcal{B}_i(p_n)\}$ be extracted birthmarks from P by certain birthmark extraction method $\mathcal{B}_i (1 \leq i \leq n)$. We store $S^{\forall} = \{S_1, S_2, ..., S_m\}$ into the database of the search engine. For the scenario of the proposed method, the user posts a request $\mathcal{B}_x(q)$ to the search engine, then the search engine compares elements in $S_x (1 \leq x \leq m)$.

2) Scenario of the proposed method: Then, let $R^{\forall}(\mathcal{B}_k(p)) = \{R_1, R_2, ..., R_l\}$ be a result set of the search engine. $R_j = \{c_j, \mathcal{B}_k(p_j), s_j\}$ contains the class name c_j, birthmark of p_j corresponding c_j ($\mathcal{B}_k(p_j)$), and similarity s_j between $\mathcal{B}_k(p)$ and $\mathcal{B}_k(p_j)$ computed by the search engine $(1 \leq j \leq l)$. Then, we obtain $\mathcal{R}(\mathcal{B}_k(p)) = \{R_1, R_2, ..., R_r\}$ by filtering $s_i \geq \varepsilon$ in $R^{\forall}(\mathcal{B}_k(p))(1 \leq i \leq l)$.

3) Scenario of the previous birthmark system: Let $V^{\forall}(\mathcal{B}_k(p)) = \{V_1, V_2, ..., V_n\}$ be a result set of the previous birthmark system. $V_j = \{c_j, \mathcal{B}_k(p_j), v_j\}$ contains the class name c_j, birthmark of p_j corresponding c_j ($\mathcal{B}_k(p_j)$), and similarity v_j between $\mathcal{B}_k(p)$ and $\mathcal{B}_k(p_j)$ computed by the previous birthmark system $(1 \leq j \leq n)$. Then, we obtain $\mathcal{V}(\mathcal{B}_k(p)) = \{V_1, V_2, ..., V_u\}$ by filtering $v_j \geq \varepsilon$ in $V^{\forall}(\mathcal{B}_k(p))(1 \leq j \leq n)$.

4) Metrics for evaluating the proposed method: Accuracy, precision, recall, and f-measure are employed for evaluating the proposed method. Before the definition of those metrics, we assume that the result of the previous birthmark system is correct. The followings are the formulation of TP (true positive), TN (true negative), FP (false positive), and FN (false negative).

For counting TP, we obtained name set $N_{\mathcal{R}}$ by the extracting element of c_i from each R_i of $\mathcal{R}$. $N_{\mathcal{V}}$ is obtained similarly from each V_j of $\mathcal{V}$. TP count is defined as $|N_{\mathcal{R}} \cap N_{\mathcal{V}}|$. Next, we define $\overline{N_{\mathcal{R}}}$ and $\overline{N_{\mathcal{V}}}$. $N^{\forall}$ is collecting names n_i of p_i from P. Then, we formulate $\overline{N_{\mathcal{R}}} = \{c_i \notin N_{\mathcal{R}} | N^{\forall}\}$. Similarly, $\overline{N_{\mathcal{V}}} = \{c_i \notin N_{\mathcal{V}} | N^{\forall}\}$. Then, FP is collecting names appeared in $N_{\mathcal{R}}$, and not appeared in $N_{\mathcal{V}}$ ($|N_{\mathcal{R}} \cap \overline{N_{\mathcal{V}}}|$). Similarly, FN is collecting names not appeared in $N_{\mathcal{R}}$ and appeared in $N_{\mathcal{V}}$ ($|\overline{N_{\mathcal{R}}} \cap N_{\mathcal{V}}|$). Finally, TN is counting names which are not in both $N_{\mathcal{R}}$ and $N_{\mathcal{V}}$ ($|\overline{N_{\mathcal{R}}} \cap \overline{N_{\mathcal{V}}}|$).

Then, accuracy M_a, precision M_p, recall M_r, and f-measure M_f is defined as follows.

$$M_a = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$M_p = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$M_r = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$M_f = \frac{2M_pM_r}{M_p + M_r}$$

III. Experimental Evaluation

A. Experimental Setup

We evaluated the proposed method in the following four perspectives:

1) Time for pre-comparison and comparison,
2) False positives check,
3) False negatives check, and
4) The accuracy of a search engine.

The purpose of the proposed method is to reduce total time, especially comparison time. In the first experiment, we measure the time for the comparison. The comparison time means the time required for pre-comparison phase and comparison phase. However, the accuracy of the proposed method is also important perspectives. Therefore, we also evaluated the proposed method in the perspectives of false positives (incorrectly detected innocent program), and false negatives (undetected copies). In addition, We analyze the accuracy of the proposed method. Because we consider proving accuracy not enough. Therefore, we also evaluated the proposed method in the perspectives of accuracy.

Before experiments, we prepared the experimental environment; We used Apache solr[4] as a search engine on a Mac Book

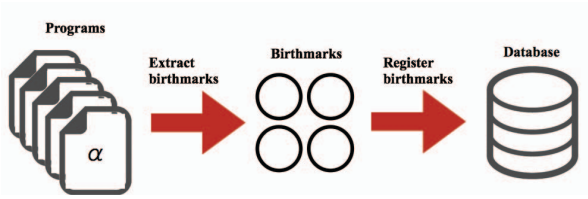

Fig. 1: The registration step of the proposed method

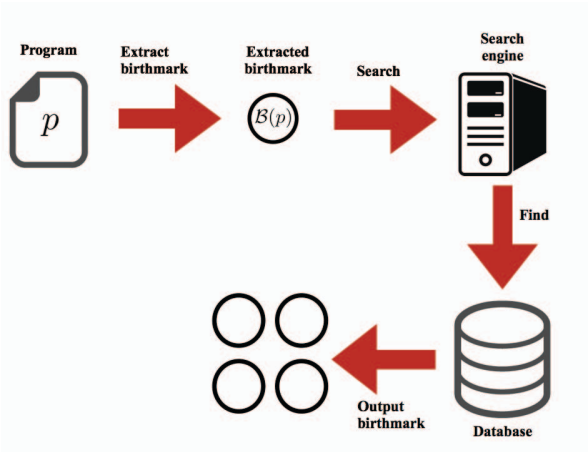

Fig. 2: The search step of the proposed method

[4]http://lucene.apache.org/solr/

Pro (OS X El Capitan with 2.7GHz Intel Core i5 CPU and 16GB RAM). Also, stigmata[5] was employed as the previous birthmark system. The employed birthmark methods are k-gram, and UC, and k values are from 2 to 6[2], [5]. Those birthmark methods are able to compare fast, since they are quite simple, even if we apply the previous method. Besides, ε was 0.75 from the previous researches[9].

B. Workflow of the experiments

The workflow of the experiments is shown in Figure 3 and described s follows.

(1) We choose the search target sets as ascendant 10,000 jar files, obtained from maven repository in alphabetical order[11].
(2) We extracted certain birthmarks from all class files in the jar files (507,707 class files).
(3) Extracted birthmarks were stored in the database of the search engine.
(4) We randomly chose a jar file from the maven repository for the search query. The chosen jar file was not included in the search target sets.
(5) We also extracted the birthmarks from classes in the chosen jar file.
(6) Each extracted birthmark was posted a request iteratively to the search engine.
(7) We obtained birthmarks of the search result with a search engine.
(8) We measure the similarity between posted birthmark and birthmarks of a search result with the previous birthmark system.

C. Comparison time

The section describes the evaluation for measuring comparison times. We measured the spent time for step (6), and (7). Since both steps include the pre-comparison phase, measured time is for pre-comparison phase.

The purpose of the paper is to reduce the total time for comparison. Therefore, we measured the time for comparison phase after pre-comparison phase. Hence, we collect class files from the response of the search engine (7) in Figure 3. Then, we measure the time for comparing the among the collected class files and request class files by the previous birthmark system (steps (6) to (8) in Figure 3).

Table I shows the result of the experiment. The first column represents birthmarks. The second column shows required time using the previous birthmark system. The following columns are required times of the steps (6), and (7), steps (6) to (8), and comparison speed rate. The last column represents the narrowed rate of the defendants by the proposed method.

From the result, the speed rates of k-gram birthmarks are lower than 5.5% except for 2-gram birthmark. The speed rates of 2-gram birthmark and UC birthmark are larger than other k-gram birthmarks. However, the required time for pre-comparison and comparison phases is shorter than the previous birthmark system.

The narrowed rates of k-gram birthmarks are greatly low, less than 4%. Unfortunately, the narrowed rate of UC birthmark is about 75%. The narrowed rate was high, however, the total spent time succeeded to reduce. From the result, the proposed method can reduce the total comparison time.

D. False positives evaluation

The experiment evaluates the false positives of the search result obtained by the proposed method. The false positives are the detection rate of innocent programs. The experimental setup was same as the previous experiment in Section III-C. Then, we obtained a set of class names $N_{\mathcal{R}}$ from the search results. Next, we compared the birthmarks corresponding extracted class names. Finally, we compute the false positive rates $C_{\mathrm{FP}} = \frac{\mathrm{FP}}{N_{\mathcal{R}}} = \frac{N_{\mathcal{R}} \cap \overline{N_{\mathcal{V}}}}{N_{\mathcal{R}}}$ from the comparison results by the previous birthmark system.

Table II shows the result of the experiment. Each line of Table II means each birthmark. The first and second columns show the length of $|N_{\mathcal{R}}|$ and FP ($|N_{\mathcal{R}} \cap \overline{N_{\mathcal{V}}}|$) defined in Section II-E. The final column represents the false positive rate C_{FP} of the proposed method. Also, the false positive rates of each birthmark are shown in Figure 4. From the Figure 4 and Table II, the false positive rates of 2-gram, 3-gram, and UC birthmarks are under 7%. This result shows that

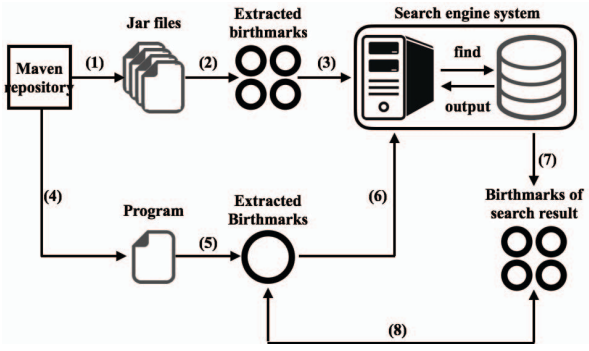

Fig. 3: The workflow of the experiments

[5]https://github.com/tamada/stigmata

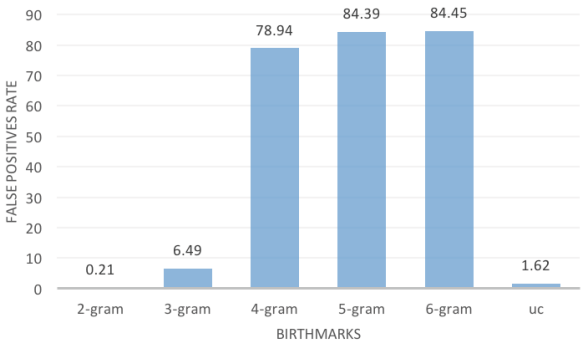

Fig. 4: False positive rates of each birthmark

TABLE I: Spent time for comparison

Birthmark	Conventional method (t^f)	The proposed method (6), and (7) (t^p)	The proposed method (6) to (8) (t^a)	The proposed method Speed rate (t^a/t^f)	Narrowed Rate ($\lvert N_{\mathcal{R}}\rvert/\lvert N^{\forall}\rvert$)
2-gram	297.40	0.95	101.98	34.29%	3.36%
3-gram	724.12	0.92	35.22	4.86%	1.18%
4-gram	919.98	0.95	48.27	5.25%	1.40%
5-gram	1,015.54	0.90	30.50	3.00%	1.07%
6-gram	911.18	0.93	10.56	1.16%	0.60%
UC	934.41	0.43	760.18	81.35%	74.81%

TABLE II: Result of false positives evaluation

	$\lvert N_{\mathcal{R}}\rvert$	$\lvert N_{\mathcal{R}} \cap \overline{N_{\mathcal{V}}}\rvert$	C_{FP}
2-gram	400,580	842	0.21%
3-gram	205,702	13,351	6.49%
4-gram	213,064	168,183	78.94%
5-gram	284,265	239,894	84.39%
6-gram	290,762	245,552	84.45%
UC	357,799	5,782	1.62%

the search engine successfully detects the suspected copies. However, 4-gram, 5-gram, and 6-gram birthmarks are over about 80%. The result means the almost class names from the search engine were innocent programs. Although the false positives are acceptable, the correct rates of these birthmarks are extremely high. For more practical use, we should review the threshold ε.

E. False negatives evaluation

The experiment evaluates the false negatives of the search result obtained by the proposed method. The procedure of the experiment is as follows.

1) We chose 50 jar files randomly from Maven repository[11].
2) we extracted birthmarks from classes of, all of every jar files (6,023 class files).
3) Then, we compared extracted birthmarks by the previous birthmark system. The comparison results are $\mathcal{V}$ by filtering with $\varepsilon = 0.75$.
4) Next, we posted each birthmark of $\mathcal{V}$, and we merge each result for obtaining $\mathcal{R}$.

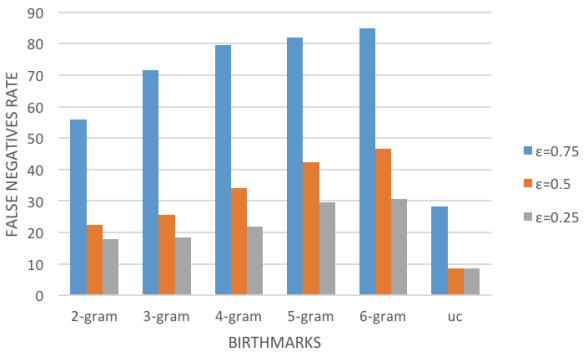

Fig. 5: False negatives rates of each birthmark

5) We investigated the search results $\mathcal{R}$, and calculate false negative rate $C_{\text{FN}} = \frac{\text{FN}}{|N_{\mathcal{V}}|} = \frac{|\overline{N_{\mathcal{R}}} \cap N_{\mathcal{V}}|}{|N_{\mathcal{V}}|}$.
6) In additional, we change threshold ε to 0.75, 0.5, and 0.25 and conduct above procedure.

Table III shows the result of false negatives check. The first column represents birthmarks. The following columns show $|N_{\mathcal{V}}|$, false negatives, and false negative rate C_{FN} of each threshold ε. Also, Figure 5 shows the false positive rates of each birthmark.

From the result, UC birthmark shows good false negative rates, since every C_{FN} are lower than 30%. On the other hand, k-gram birthmarks were not good false negative rates in $\varepsilon = 0.75$, since every C_{FN} is greater than 50%. In $\varepsilon = 0.5$, the false negative rates of 2-gram, and 3-gram birthmarks were lower than 30%. The k-gram birthmarks show good false negative rates in $\varepsilon = 0.25$ because the all of C_{FN} are lower than about 30%. Therefore, false negative values were improved by lowering the threshold ε.

F. Accuracy of the Proposed Method

The goal of the proposed method is to reduce the comparison time. We can accept the slightly worse of the comparison accuracy than the previous method. However, evaluating the performance of the proposed method is important concerns. Therefore, the experiment evaluates the accuracy of the proposed method by the metrics defined in Section II-E. The procedure of the experiment is same as the previous experiment.

Table IV describes the resultant metrics shown in Section II-E. The first column represents birthmarks. The following columns show M_a, M_p, M_r, and M_f in each threshold ε. Precisions M_p of all of birthmarks and threshold shows over 98%, therefore, almost search results are suspected copies. However, the method could not detect actual copies by 5-gram and 6-gram in $\varepsilon = 0.75$ since recalls M_r in are lower than 20%. In $\varepsilon = 0.5$, the results show better the accuracies M_a, recalls M_r, and f-measures M_f than $\varepsilon = 0.75$. Also, in $\varepsilon = 0.25$, the metrics were more improved from $\varepsilon = 0.5$. Therefore, by lowering the threshold ε, the accuracies of the proposed method improved. The appropriate threshold should be decided by the user's discretion. Other methods for improving the accuracies are also important our future works.

IV. Conclusion

This paper proposed the pre-comparison phase to narrow defendants from the enormous amount of programs. Our pre-comparison phase employed the search engine for fast com-

TABLE III: Result of false negatives evaluation

Birthmark	$\varepsilon = 0.75$			$\varepsilon = 0.5$			$\varepsilon = 0.25$		
	$\lvert N_{\mathcal{V}} \rvert$	$\lvert \overline{N_{\mathcal{R}}} \cap N_{\mathcal{V}} \rvert$	C_{FN}	$\lvert N_{\mathcal{V}} \rvert$	$\lvert \overline{N_{\mathcal{R}}} \cap N_{\mathcal{V}} \rvert$	C_{FN}	$\lvert N_{\mathcal{V}} \rvert$	$\lvert \overline{N_{\mathcal{R}}} \cap N_{\mathcal{V}} \rvert$	C_{FN}
2-gram	5,898	3,302	56.00%	5,898	1,317	22.33%	5,898	1,058	17.93%
3-gram	5,898	4,218	71.52%	5,898	1,514	25.68%	5,898	1,087	18.44%
4-gram	5,900	4,687	79.45%	5,900	2,020	34.23%	5,900	1,287	21.81%
5-gram	5,900	4,836	81.97%	5,900	2,498	42.33%	5,900	1,741	29.52%
6-gram	5,900	5,010	84.92%	5,900	2,748	46.58%	5,900	1,812	30.71%
UC	5,910	1,665	28.17%	5,910	510	8.62%	5,910	508	8.59%

TABLE IV: Metrics of the proposed method

Birthmark	$\varepsilon = 0.75$				$\varepsilon = 0.5$				$\varepsilon = 0.25$			
	M_a	M_p	M_r	M_f	M_a	M_p	M_r	M_f	M_a	M_p	M_r	M_f
2-gram	44.99%	99.59%	44.00%	61.04%	77.40%	99.04%	77.67%	87.06%	81.66%	99.04%	82.07%	89.76%
3-gram	29.76%	99.25%	28.48%	44.26%	74.15%	99.04%	74.32%	84.92%	81.17%	99.04%	81.56%	89.45%
4-gram	22.04%	99.30%	20.55%	34.06%	65.87%	99.08%	65.77%	79.06%	77.93%	99.09%	78.19%	87.40%
5-gram	19.57%	99.21%	18.03%	30.52%	57.93%	98.95%	57.67%	72.87%	70.38%	98.99%	70.48%	82.34%
6-gram	16.71%	99.29%	15.08%	26.19%	53.77%	98.87%	53.42%	69.36%	69.25%	99.02%	69.29%	81.53%
UC	72.00%	99.50%	71.83%	83.43%	90.94%	99.34%	91.38%	95.19%	90.97%	99.34%	91.41%	95.21%

parison. We evaluated the proposed method on two existing birthmark types, k-gram and UC birthmarks. The evaluation aspects were comparison time, false positives, false negatives, and accuracy of the proposed method. The result shows that our method reduced the total comparison time dramatically. However, false negative rates were not good except UC birthmarks in default threshold $\varepsilon = 0.75$. By lowering the threshold, the accuracies of the proposed method could improve. In the future works, the appropriate threshold will be found out. Also, the method for improving the accuracies is also the important concern.

REFERENCES

[1] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto, "Design and evaluation of birthmarks for detecting theft of Java programs," in *Proc. IASTED International Conference on Software Engineering (IASTED SE 2004)*, February 2004, pp. 569–575, (Innsbruck, Austria).

[2] ——, "Java birthmarks —detecting the software theft —," *IEICE Transactions on Information and System*, vol. E88-D, no. 9, pp. 2148–2158, September 2005.

[3] T. Tsuzaki, T. Yamamoto, H. Tamada, and A. Monden, "A fuzzy hashing technique for large scale software birthmarks," in *Proc. 15th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2016)*, June 2016, pp. 867– 872.

[4] H. il Lim, H. Park, S. Choi, and T. Han, "A static java birthmark based on control flow edges," in *Proc. the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009)*, vol. 1, 2009, pp. 413–420.

[5] G. Myles and C. Collberg, "K-gram based software birthmarks," in *Proc. the 20th Annual ACM Symposium on Applied Computing*, 2005, pp. 314–318.

[6] P. P. F. Chan, L. C. K. Hui, and S. Yiu, "Heap graph based software theft detection," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 101–110, 2013.

[7] Y.-C. Jhi, X. Wang, X. Jia, S. Zhu, P. Liu, and D. Wu, "Value-based program characterization and its application to software plagiarism detection," in *Proc. the 33rd International Conference on Software Engineering*, 2011, pp. 756–765.

[8] H. il LIM, H. PARK, S. CHOI, and T. HAN, "Detecting theft of java applications via a static birthmark based on weighted stack patterns," *IEICE TRANSACTIONS on Information and Systems*, vol. E91-D, no. 9, pp. 2323–2332, 2008.

[9] D. Schuler, V. Dallmeier, and C. Lindig, "A dynamic birthmark for java," in *Proc. of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*, 2007, pp. 274–283. [Online]. Available: http://doi.acm.org/10.1145/1321631.1321672

[10] Z. Tian, Q. Zheng, T. Liu, and M. Fan, "DKISB: Dynamic key instruction sequence birthmark for software plagiarism detection," in *Proc. 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC)*, November 2013, pp. 619–627.

[11] S. Raemaekers, A. van Deursen, and J. Visser, "The maven repository dataset of metrics, changes, and dependencies," in *Proc. 2013 10th IEEE Working Conference on Mining Software Repositories (MSR 2013)*, May 221–224, p. 2013.