Hindawi
Security and Communication Networks
Volume 2020, Article ID 8852124, 9 pages
https://doi.org/10.1155/2020/8852124

Research Article

Modelling Features-Based Birthmarks for Security of End-to-End Communication System

Meilian Li,[1] Shah Nazir,[2] Habib Ullah Khan,[3] Sara Shahzad,[4] and Rohul Amin[5]

[1]*School of Electronic and Electrical Engineering, Anhui Sanlian Univeristy, Hefei, Anhui 230601, China*
[2]*Department of Computer Science, University of Swabi, Ambar, Khyber Pakhtunkhwa 23430, Pakistan*
[3]*Department of Accounting and Information System, College of Business and Economics, Qatar University, Doha 2713, Qatar*
[4]*Department of Computer Science, University of Peshawar, Peshawar, Khyber Pakhtunkhwa 25120, Pakistan*
[5]*Department of Mathematics, University of Peshawar, Peshawar, Khyber Pakhtunkhwa 25120, Pakistan*

Correspondence should be addressed to Habib Ullah Khan; habib.khan@qu.edu.qa

Received 12 May 2020; Revised 30 May 2020; Accepted 4 June 2020; Published 29 June 2020

Academic Editor: Iqtadar Hussain

Copyright © 2020 Meilian Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Feature-based software birthmark is an essential property of software that can be used for the detection of software theft and many other purposes like to assess the security in end-to-end communication systems. Research on feature-based software birthmark shows that using the feature-based software birthmark joint with the practice of software birthmark estimation together can deliver a right and influential method for detecting software piracy and the amount of piracy done by a software. This can also guide developers in improving security of end-to-end communication system. Modern day software industry and systems are in demand to have an unbiased method for comparing the features-based birthmark of software competently, and more concretely for the detecting software piracy and assessing the security of end-to-end communication systems. In this paper, we proposed a mathematical model, which is based on a differential system, to present feature-based software birthmark. The model presented in this paper provides an exclusive way for the features-based birthmark of software and then can be used for comparing birthmark and assessing security of end-to-end communication systems. The results of this method show that the proposed model is efficient in terms of effectiveness and correctness for the features-based software birthmark comparison and security assessment purposes.

1. Introduction

Software piracy is considered to be a foremost anxiety for the industry of software. Software piracy is done due to the large growth of Internet and software industry. Wide-ranging research [1] into the way to do piracy of software detection has encouraged the progress of techniques such as watermarking in software, fingerprints, and recently the birthmark of software. Birthmark of software is inherent characteristic or property of software to be effectively used for theft of software and detection of software piracy. Software watermark and fingerprint have been used for a long time with the realization but these techniques have some limitations. Some of the researchers and practitioners of industry are using forward-looking versions of software watermark [1–12], fingerprints [13, 14], software clone [15, 16], and software birthmark [17–29]. Detection of plagiarism is relevant area to these mentioned software detection methods which are used for source code theft and discovery of similarities among the original and duplicated source codes [30–35]. Watermark of software is used to express the proprietorship of a software. The watermarks add some supplementary code or detail information to the existing software to show the ownership. Software fingerprint is used to find the intellectual property. Cloning of software is done by copy-past of source code of copyrighted software that may be in parts or full in another version of the software. The methods of clone detection of software are used to sense the piracy in such cases. Software birthmark is considered to be the recently used technique for the software piracy detection. Birthmarks of software use the inherent characteristics or software properties to identify the originality of software. Birthmark similarities of two software programs show the extent of piracy done among the software.

The concept of birthmark of software is offered for the similar determination of theft identification of software and detection of piracy. Birthmark of software is till now recognized to be resilient to any obliteration or obfuscation technique(s). Several researches have been accomplished to recognize diverse types of birthmark of software [20, 24–27, 29, 36–40]. Nazir et al. [36] offered the strategy of feature-based software birthmark and a proper estimation process for birthmark of software [37]. Though birthmark of software has been extensively deliberated in research from several viewpoints of the area of software piracy and detection of theft, yet there is no objective measure to compare birthmarks of software efficiently for the detection of piracy and to assess the security of end-to-end communication systems. The aim of the proposed work is to deliver a mathematical model for the purpose of comparison of feature-based birthmark of software and to assess the security of end-to-end communication systems. The proposed model is based on differential equations system and uses the features of birthmark, presented by Nazir et al. [36] and can be assessed for the comparison purpose of features-based birthmark of another (duplicate) software and assessment purpose of the security of end-to-end communication systems. These comparisons will ultimately endorse or reject the piracy performed in software and security changes that occurred in the applications.

The organization of the paper is as follows: Section 2 of the paper presents related work done for software birthmark and detection of piracy. Section 3 gives the details of the mathematical model used for the proposed research, with logic of using mathematical model for birthmark comparisons. This section further provides explanation for the use of differential equations as system model. The results and discussion of the proposed research are discussed in Section 4. This section further discusses the case study of the method. The paper concludes in Section 5.

2. Related Work

Software industry and productions are facing with a dreadful problem of piracy and changes of security in software. On the other hand, the pirates of software make vast sums of money from the trade performed in piracy and changes in security of software. According to the report of Business Software Alliance (BSA) [41] of year 2013, about 43% of software programs that are configured on personnel computer systems in the globe were pirated and not appropriately licensed. The marketable value of these unlicensed software programs was about 62.7 billion dollars. Taking this point further, Myles and Collberg [29] outlined the three foremost threats to industry of software. These threats include the illegal re-selling of the legitimate software, malicious reverse engineering, and software tampering. The industries of software adopt diverse practices to trace the theft of software. Among these practices, the software birthmark is one of the techniques which are used for the detection of pirated software and by the assistance of which the pirated or duplicated version of the software would be traced. The software birthmark types and history could be taken at length. Tamada et al. [42] designed the very first birthmark method which is based on four types of birthmark; these birthmarks are constant values in field variables (CVFV), inheritance structure (IS), sequence of method calls (SMC), and used classes (UC). This technique of birthmark was effectively used by the software industry for the purpose of detection theft of software. Myles and Collberg [29] suggested a method of "Whole Program Path Birthmark." This method is based on the whole control flow of the program. The properties of resilience and credibility were used to assess the effectiveness of the method. The method further reveals that the WPPB is more resilient than the existing methods of birthmark. Zeng et al. [43] proposed a framework of semantic-based abstract interpretation for software birthmark. Mahmood et al. [44] proposed a method-based similarity level for software birthmark. By help of the proposed method, the elements of code and their properties can be found. This method traces the modification occurring in the program. Wang et al. [45] suggested the operand stack dependence-based static software birthmark for the difficulty of semantic lost when mining birthmark with the help of k-gram algorithm.

Moreover, through offering different types of birthmark, several researchers have provided some case studies for the work of their analysis and evaluation they performed. Choi et al. [23] analyzed the static API-based birthmark of software for binary executable of Windows and compared 49 executables. They described that the birthmark used by them can easily distinguish and identify the program copies. The birthmark is checked with the Windows dynamic birthmark and presented to likely suitable for the applications with Graphical User Interfaces. Kakimoto et al. [28] did analysis of the birthmark similarities in Argo UML and then visualized them using multidimensional scale. Park et al. [24] proposed a static API trace birthmark for detection of theft of Java-based programs. This technique assesses the birthmark for the properties of resilience and credibility. Results obtained from their experiment of the proposed method show that the static API birthmark can identify related components of two packages while the other techniques of birthmarks fail to do so. Xie et al. [46] suggested a static birthmark for k-gram and their weights. The weight is computed by analysis rate of change in the k-gram frequency of the actual and modified version of the program. Myles and Collberg [47] accomplished an empirical analysis of the k-gram-based software birthmark by analysis of 111 programs in the Java programming language. Several studies [20, 24–27, 29, 35, 36, 42, 43, 48–51] were explored for the types of birthmark, their analysis, and assessment, but the work of [48, 49] analyzed the birthmark in depth used for different purposes. From most of the studies, it is derived that in majority of the cases only the results of case study and empirical suggestions are provided to support the given studies.

The current research work is endeavouring to propose a mathematical model for the purpose of comparisons of feature-based software birthmark and to evaluate the security of end-to-end communication systems. The model is based on differential equations system and uses the features of birthmark presented in the literature.

3. Methodology

The methodology is described in the following subsections which present the proposed research methodology for the features-based birthmark of software.

3.1. Need for a Mathematical Model. Diverse methods based on mathematics are used by the researchers and practitioners for modelling the real life occurrences. A number of these techniques include exact equations, linear equations, separable variable methods, substitution solution, and numerical method. These techniques are used to solve the first-order differential equations [52].

Software industry is endeavouring to have a policy and strategic independent description for birthmarks of software, which can then be used as proper estimating and comparisons of birthmark of software. This definition and description will ease the industry of software to detect software theft and piracy with further changes in security of end-to-end communication systems. The recommended feature-based software birthmark [36] is currently mathematically modelled to enable the birthmark comparison based on the defined features. This feature-based birthmark comparison will identify the similarities among software programs for the purpose of piracy detection and changes in security of end-to-end communication systems.

In this research work, the essential model is planned in the form of homogeneous linear differential system. For the design of this type of system, generally three methods are used. These methods are repeated Eigen values, distinct real Eigen values, and Complex Eigen values. In the situation of the proposed research, the Eigen values are complex.

Mathematically, if $\lambda_1 = \alpha + i\beta$ and $\lambda_2 = \alpha - i\beta$, where $i = \sqrt{-1}$ are Complex Eigen values of the matrix "A," then the corresponding Eigen vector also contains complex values [52]. This study proposed a mathematical model for the features-based software birthmark to enable the comparisons among the birthmark based on the predefined features.

3.2. Terminologies Used for Modelling Software Piracy Detection. The following subsections briefly discuss the method and terminologies used in this research for modelling features-based birthmark of software.

3.2.1. Differential Model for Software Birthmark. The differential equations have the derivatives of one or more dependent variable(s), with respect to one or more independent variable(s) [52]. Let there be an equation with unknown variables, without any information available about its construction. Such type of an equation (function) can be represented as, for example, $y' = \phi(x)$?

3.2.2. Eigen Values and Eigen Vector. The characteristic polynomial of a square matrix "A" is defined by [53]

$$p(\lambda) = \det(A - \lambda I). \tag{1}$$

If p is the characteristic polynomial of matrix "A", then the roots of p are the Eigen values of matrix "A." If λ is Eigen value of "A" and $x \neq 0$ satisfies $(A - \lambda I)x = 0$, then x is Eigen vector corresponding to the Eigen value λ. In the context of this research, there are three main features (categories), from which a differential system is obtained. This differential system is also called linear differential system. To solve this differential system, we need the Eigen vector for the corresponding Eigen values.

3.3. Model for Comparison of Birthmark for Detection of Software Piracy and Assessment of Security in End-to-End Communication Systems. Diverse approaches have been used in literature in the area of development of healthcare mobile applications. The proposed technique for comparisons of suggested features-based software birthmark is mathematically modelled to enable and facilitate the comparisons of birthmarks and assessment of security of end-to-end communication systems based on the identified features. The features followed by the proposed study are the features that are already identified in the previous research work [36, 48, 49]. This features-based comparison advises the similarity among different modules of the software which can further investigate the changes occurring in the security of end-to-end communication systems. Here, in this study, we considered the four main features that were previously identified [36]. These features include preconditional features, input features, nonfunctional features, and functional features. These categories are further divided into subcategories of features. The preconditional features have three subfeatures categories that are program availability, runnable, and identification of components. These features are significant which can be patterned even for all kinds of programs for detecting the similarities. Figure 1 shows the detail of the feature-based birthmark of software as already defined [36].

After performing the early analysis, the rest of the three features categories are used as the base of mathematical model for the proposed study, while the category of preconditional features is excluded, as this features category can be examined for all types of software while detecting the piracy and changes in security of software. The input feature category is further divided into 17 features that are program context, program contents, internal data structure, program flow, configurable terminologies, program responses, control flow, size of program, interface description, number of statements in program, naming, functions, restriction, limitation and constraints, comprehensive documentation, global data structure, user interface, and internal quality. The nonfunctional feature category is further divided into 12 subfeatures that are automation, ease of use, friendly, scalability, applicability, interface connections, robustness, dependency, portability, scope, standard, and

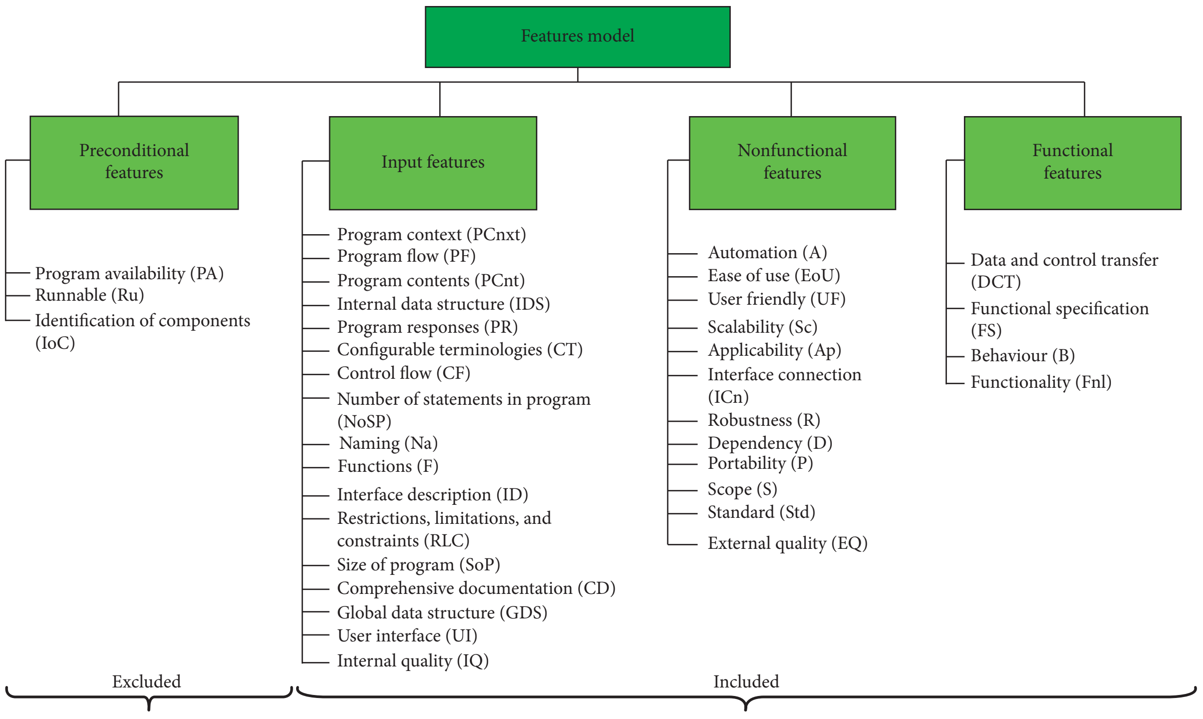

FIGURE 1: Software features and their subfeatures.

external quality. The functional feature category is divided into further four subfeatures that are data and control process, functional specification, behaviour, and functionality. All the categories of these features are combined and then plotted in the form of differential system mathematically as

$$\begin{aligned} x'(f) &= 17x + 12y + 4z, \\ y'(f) &= 4x + 17y + 12z, \\ z'(f) &= 12x + 4y + 17z, \end{aligned} \tag{2}$$

where x, y, and z are the three features. Then, from equation (2), we have

$$X'(f) = AX(f). \tag{3}$$

To find these three features x, y, and z, we need to find the solution of equation (2). For this purpose of finding the exact solution, we have to find the Eigen values and Eigen vectors of the matrix A. The proposed process has been carried out in the following steps.

Step 1. To find Eigen value,

$$\text{Since } A = \begin{bmatrix} 17 & 12 & 4 \\ 4 & 17 & 12 \\ 12 & 4 & 17 \end{bmatrix}. \tag{4}$$

According to Section 3.2.2, by using equation (1), the characteristic polynomial of the matrix "A" is given by det $(A - \lambda I) = 0$. That is,

$$\begin{vmatrix} 17-\lambda & 12-0 & 4-0 \\ 4-0 & 17-\lambda & 12-0 \\ 12-0 & 4-0 & 17-\lambda \end{vmatrix} = 0. \tag{5}$$

After simplification, we have

$$\lambda^3 - 51\lambda^2 + 723\lambda - 4257 = 0. \tag{6}$$

By using syntactic division, we have

$$\begin{aligned} \lambda_1 &= 33, \\ \lambda_2 &= 9 + 6.9282i, \\ \lambda_3 &= 9 - 6.9282i. \end{aligned} \tag{7}$$

Thus, the Eigen values of the matrix "A" are 33, $9 + 6.9282i$, and $9 - 6.9282i$, where λ_1 is real, λ_2 is complex, and λ_3 is complex conjugate of λ_2.

Step 2. To find Eigen vector of corresponding Eigen values, If $\lambda = 33$, then the corresponding Eigen vector is given by $AX = \lambda X$.

$$\begin{bmatrix} 17 & 12 & 4 \\ 4 & 17 & 12 \\ 12 & 4 & 17 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 33 \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \tag{8}$$

By solving this, we have

$$\begin{aligned} -16a + 12b + 4c &= 0, \\ 4a - 16b + 12c &= 0, \\ 12a + 4b - 16c &= 0. \end{aligned} \tag{9}$$

By solving this, we have

$$\begin{aligned} a &= 1, \\ b &= 1, \\ c &= 1. \end{aligned} \tag{10}$$

Thus, the corresponding Eigen vector for Eigen value $\lambda = 33$ is $V1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$.

Similarly, the corresponding Eigen vectors for $9 + 6.9282i$ and $9 - 6.9282i$ are given by

$$\begin{aligned} V_2 &= \begin{pmatrix} \frac{1}{2} i (i + \sqrt{3}) \\ -\frac{1}{2} i (-i + \sqrt{3}) \\ 1 \end{pmatrix}, \\ V_3 &= \begin{pmatrix} -\frac{1}{2} i (-i + \sqrt{3}) \\ \frac{1}{2} i (i + \sqrt{3}) \\ 1 \end{pmatrix}. \end{aligned} \tag{11}$$

Step 3. Thus, the solution of equation (2) is given by

$$\begin{aligned} X = {} & c_1 V_1 e^{\lambda_1 f} + c_2 (B_1 \cos \beta f - B_2 \sin \beta f) e^{\alpha f} \\ & + c_3 (B_2 \cos \beta f + B_1 \sin \beta f) e^{\alpha f}, \end{aligned} \tag{12}$$

where $\lambda = \alpha + i\beta$, $B_1 =$ real part (Eigen vector) and $B_2 =$ imaginary part (Eigen vector). Putting the values in the above equation, we get

$$X = c_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} e^{33f} + c_2 \left(\begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ 1 \end{pmatrix} \cos 6.9282f - \begin{pmatrix} \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} \\ 0 \end{pmatrix} - \sin 6.9282f \right) e^{9f}$$

$$+ c_3 \left(\begin{pmatrix} \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} \\ 0 \end{pmatrix} \cos 6.9282f + \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ 1 \end{pmatrix} \sin 6.9282f \right) e^{9f},$$

$$x(f) = c_1 e^{33f} + c_2 (\cos(6.9282f)) e^{9f} + c_3 (\sin(6.9282f)) e^{4f}. \tag{13}$$

Similarly, we have

$$\begin{aligned} y(f) &= c_1 e^{33f} + c_2 \left[-\frac{1}{2} \cos(6.9282f) + \frac{\sqrt{3}}{2} \sin(6.9282f) \right] e^{9f} + c_3 \left[-\frac{\sqrt{3}}{2} \cos(6.9282f) - \frac{1}{2} \sin(6.9282f) \right] e^{9f}, \\ z(f) &= c_1 e^{33f} + c_2 \left[-\frac{1}{2} \cos(6.9282f) - \frac{\sqrt{3}}{2} \sin(6.9282f) \right] e^{9f} + c_3 \left[\frac{\sqrt{3}}{2} \cos(6.9282f) - \frac{1}{2} \sin(6.9282f) \right] e^{9f}. \end{aligned} \tag{14}$$

Putting the value of $f = 0$ in the above equations and using the initial conditions, we have

$$\begin{aligned} c_1 - \frac{1}{2} c_2 + \frac{\sqrt{3}}{2} c_3 &= 17, \\ c_1 - \frac{1}{2} c_2 - \frac{\sqrt{3}}{2} c_3 &= 4, \\ c_1 + c_2 &= 12. \end{aligned} \tag{15}$$

By solving these equations, we get

$$\begin{aligned} c_1 &= 11, \\ c_2 &= 1, \\ c_3 &= -7.5056. \end{aligned} \tag{16}$$

Thus, the required solution of (2) is given by

$$\begin{aligned} x(f) &= 11 e^{33f} + \cos(6.9282f) e^{9f} - 7.5056 \sin(6.9282f) e^{4f}, \\ y(f) &= 11 e^{33f} + \left[-\frac{1}{2} \cos(6.9282f) + \frac{\sqrt{3}}{2} \sin(6.9282f) \right] e^{9f} - 7.5056 \left[-\frac{\sqrt{3}}{2} \cos(6.9282f) - \frac{1}{2} \sin(6.9282f) \right] e^{9f}, \\ z(f) &= 11 e^{33f} + \left[-\frac{1}{2} \cos(6.9282f) - \frac{\sqrt{3}}{2} \sin(6.9282f) \right] e^{9f} - 7.5056 \left[\frac{\sqrt{3}}{2} \cos(6.9282f) - \frac{1}{2} \sin(6.9282f) \right] e^{9f}, \end{aligned} \tag{17}$$

where $x(f)$, $y(f)$, and $z(f)$ represent the required solution of the differential system (2) for the available features of software birthmark.

For the process of comparisons of birthmark of software for the detection purpose of software piracy and assessment of security of end-to-end communication systems, birthmark(s) of various occurrences of (the same) software application defined over the same features based birthmark [36] can be modelled using the given differential system. If the solutions of both of the resulting differential systems are found the same or nearly the same, then the software is copy of the original software; hence, it is proved to be pirated and changes have occurred in the security of end-to-end communication systems.

4. Results and Discussion

The following subsections briefly discuss the results and discussion section of the paper.

4.1. Experimentation with a Case Study and the Results. The proposed research work based on mathematical model for features-based software birthmark has been validated by performing a case study. The case study was intended to test an Android mobile application for features-based software birthmark. Multiple versions of the application were generated to bear and validate the process of comparison. Copies instances of the Android mobile applications were modified (in parts) to enhance and eliminate a portion of functionality. These modifications were made through third-party developers. This process was done to mimic pirated copies of the test cases application and to assess the security of end-to-end communication systems.

After getting modified copies of the mobile (Android) application, the features-based birthmark of software was individually derived from all of the copies of the application as shown in equation (2). This was performed by extracting each individual copy of the features-based application. The features along with their details were taken into consideration for checking the piracy among the applications and to assess the security of end-to-end communication systems. The features of each copy were then extracted and the birthmarks of pirated copies of the applications were then compared with the features-based software birthmark of the actual application to show that piracy and changes in security were done/or not to further show the similarities and security view among the actual and pirated copies of the application. Figure 2 shows case study performed for features extraction from the actual and pirated version of the software and their comparison process.

A case study of the equations below was taken as an example to show the validity:

$$\frac{\partial u(x,y,z)}{\partial x} = 17x + 15xy, \tag{18}$$

$$\frac{\partial u(x,y,z)}{\partial y} = 17y + 15xz, \tag{19}$$

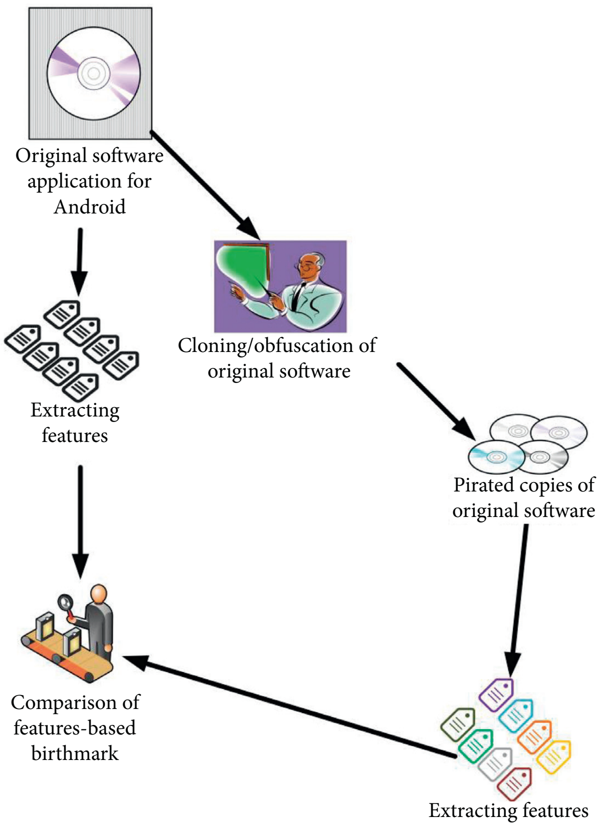
Original software application for Android
Cloning/obfuscation of original software
Extracting features
Pirated copies of original software
Comparison of features-based birthmark
Extracting features

Figure 2: Case study for feature extractions and comparisons process of actual (original) and pirated software.

$$\frac{\partial u(x,y,z)}{\partial z} = 17z + 15xy. \tag{20}$$

And the exact equation of the above system of partial differential equation is

$$u(x,y,z) = \frac{17}{2}\left(x^2 + y^2 + z^2\right) + 15xyz. \tag{21}$$

Equation (21) satisfies equations (18)–(20) and hence shows that the proposed model works well. If equation (21) is put in equations (18)–(20), then the left-hand side is equal to the right-hand side. Equation (21) is the exact solution of equations (18)–(20). So, it will satisfy for all values of the variables x, y, and z. It can be any real number. The threshold can be any real number for the variables x, y, and z.

The proposed features-based model accepts inputs of software for comparison of features of original and pirated software that is fully or partially pirated. This comparison can ultimately show the extent of piracy and changes done in security of end-to-end communication systems. In the current scenario of the case study, features of original software were extracted as shown in the top of Figure 2. Then features from pirated copies of software as shown in the right side of Figure 2 were extracted. A comparison of these features was done which is mathematically shown as equations (18)–(20) and their solution in equation (7). From the above description, it is clear that the proposed model works very well; hence, piracy and changes can be found up to optimal level.

Furthermore, some other examples were tried to show the validity of the proposed method. These examples follow:

$$\frac{\partial x(x,y,z)}{\partial x} = 34xyz + 16y^2z + 2xz^2, \tag{22}$$

$$\frac{\partial x(x,y,z)}{\partial y} = 17x^2z + 32xyz + 2xz^2, \tag{23}$$

$$\frac{\partial x(x,y,z)}{\partial z} = 17x^2y + 16xy^2 + 4xyz. \tag{24}$$

We can find the numerical solution which always contains some error. The proposed mathematical model accepts features as input(s) shown in equation (2) to check the piracy and changes in security of features-based software birthmark. In the context of the current case study, features were extracted from multiple copies of the Android mobile application to show the piracy and security changes among multiple copies of the application.

5. Discussion

Industry of software development and end-to-end communication systems is using diverse approaches and methods to detect and identify the software piracy and assessment of security in end-to-end communication systems. Different techniques like watermarks, fingerprints, and digital signatures were used for showing the originality of the software, but these techniques have some limitations such as with the use of code obfuscation and semantic preserving transformation the watermarks and digital signature can be removed. Due to these limitations, the concept of software birthmark came into existence. The software birthmarks are considered to be of the utmost value and resilient to obliteration, and uniquely identify specific software. Software features are categorised into several categories. A program of software is a combination of several types of features of software. The investigation of code of a program, based on the defined features, resultantly supports the detection of similarity among more than one instance of seemingly the identical application of software. Such detection of similarity will eventually facilitate identifying and detecting the theft and piracy of software. The features-based birthmark of software provides further wide-ranging birthmark and hence representation of a software. The proposed differential-system-based mathematical model in this study using the idea of Eigen values and Eigen vector provides an exclusive solution for the features-based birthmark of software. This exclusive solution provides an unbiased measure for comparisons of features-based software birthmark that can be checked to piracy and assessment of security in end-to-end communication systems.

5.1. Threat to Validity. Software birthmark is the inherent characteristic of software used for the detection of theft in the software and can also be used for other purposes like to show the ownership of the software and detect the level of piracy in the software. So far, the existing literature was searched to analyze the existing efforts made in the area of software birthmark but maybe some work is missed due to the open access and availability of the research work. Validation of the work is also mandatory which is not mostly covered by this research and the work was validated through experts' opinion.

6. Conclusion

The proposed research work has presented a mathematical model based on differential system for comparisons of features-based birthmark of software and assessment of security in end-to-end communication systems. These comparisons of feature-based software birthmark will eventually find piracy and changes in security performed among the end-to-end communication systems. The main objective of the proposed study is to do comparisons of the feature-based software birthmark that was addressed by Nazir et al. [36]. The birthmark of software in terms of feature-based birthmark is categorised into different types. These categories include input features, functional features, and nonfunctional features. These features-based software birthmark categories are jointly known as software birthmark. This paper contributes to present a mathematical model based on differential system for the features-based software birthmark to support the comparisons of software birthmark to be checked for piracy and security assessment of end-to-end communication systems. The solutions of the differential equation as defined by using the idea of Eigen values designed for the feature categories of the birthmarks provide an unbiased measure and an effective means to compare birthmarks of software for the purpose of detecting piracy. Therefore, this comparison of model can make the process of software piracy and theft detection smooth and assesses the security of end-to-end communication systems.

Data Availability

No primary data were collected.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by Qatar National Library, Qatar.

References

[1] R. Thabit and B. E. Khoo, "Robust reversible watermarking scheme using Slantlet transform matrix," *Journal of Systems and Software*, vol. 88, pp. 74–86, 2014.

[2] Y. Zeng, F. Liu, X. Luo, and C. Yang, "Software watermarking through obfuscated interpretation: implementation and analysis," *Journal of Multimedia*, vol. 6, no. 4, 2011.

[3] F. Liu, B. Lu, and X. Luo, "A chaos-based robust software watermarking," in *Information Security Practice and Experience*, vol. 3903, pp. 355–366, Springer, Berlin, Germany, 2006.

[4] C. Collberg and T. R. Sahoo, "Software watermarking in the frequency domain: implementation, analysis, and attacks," *Journal of Computer Security*, vol. 13, no. 5, pp. 721–755, 2005.

[5] G. Myles and C. Collberg, "Software watermarking through register allocation: implementation, analysis, and attacks," in *Information Security and Cryptology–ICISC 2003*, vol. 2971, pp. 274–293, Springer, Berlin, Germany, 2004.

[6] C. Collberg, E. Carter, S. Debray, A. Huntwork, C. Linn, and M. Stepp, "Dynamic path-based software watermarking," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 04)*, pp. 1–10, Washington, DC, USA, June 2004.

[7] G. e. Arboit, "A method for watermarking java programs via opaque predicates," in *Proceedings of the Fifth International Conference on Electronic Commerce Research (ICECR-5)*, pp. 1–8, Montreal, Canada, October 2002.

[8] R. Venkatesan, V. Vazirani, and S. Sinha, "A graph theoretic approach to software watermarking," in *Proceedings of the 4th International Information Hiding Workshop*, pp. 157–168, Pittsburgh, PA, USA, April 2001.

[9] J. P. Stern, G. e. Hachez, F. c. Koeune, and J.-J. Quisquater, "Robust object watermarking: application to code," in *Information Hiding*, vol. 1768, pp. 368–378, Springer, Berlin, Heidelberg, 2000.

[10] A. Monden, H. Iida, K.-i. Matsumoto, K. Inoue, and K. Torii, "A practical method for watermarking java programs," in *Proceedings of the 24th Computer Software and Applications Conference compsac2000*, pp. 191–197, Taipei, Taiwan, October 2000.

[11] C. Collberg and C. Thomborson, "Software watermarking: models and dynamic embeddings," in *Proceedings of the Conference Record of POPL '99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 311–324, San Antonio, Texas, USA, January 1999.

[12] G. Qu and M. Potkonjak, "Analysis of watermarking techniques for graph coloring problem," in *Proceedings of the 1998 in Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design–ICCAD'98*, pp. 190–193, New York, NY, USA, November 1998.

[13] J. Pieprzyk, "Fingerprints for copyright software protection," in *Information Security*, vol. 1729, pp. 178–190, Springer, Berlin, Heidelberg, 1999.

[14] C. S. Collberg, C. Thomborson, and G. M. Townsend, "Dynamic graph-based software fingerprinting," *ACM Transactions on Programming Languages and Systems*, vol. 29, no. 6, p. 35, 2007.

[15] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proceedings of the International Conference on Software Maintenance*, Bethesda, MD, USA, November 1998.

[16] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: a systematic review," *Information and Software Technology*, vol. 55, no. 7, pp. 1165–1199, 2013.

[17] H. T. e. al, "Detecting the theft programs using birthmarks," in *Information Science Technical Report*, Nara Institute of Science and Technology, Ikoma, Japan, 2003.

[18] H.-i. Lim, "Customizing k-gram based birthmark through partial matching in detecting software thefts," in *Proceedings of the IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, pp. 1–4, IEEE, Japan, July 2013.

[19] Z. Xin, H. Chen, X. Wang et al., "attacks: automatically evading behavior-based software birthmark," *International Journal of Information Security*, vol. 11, no. 5, pp. 293–304, 2012.

[20] H. Chen, H.-i. 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.

[21] P. P. F. Chan, L. C. K. Hui, and S. M. Yiu, "Dynamic software birthmark for java based on heap memory analysis," in *Communications and Multimedia Security*, vol. 7025, pp. 94–107, Springer, Berlin, Heidelberg, 2011.

[22] Y. Mahmood, S. Sarwar, Z. Pervez, and H. F. Ahmed, "Method based static software birthmarks: a new approach to derogate software piracy," in *Proceedings of the 2nd International Conference on Computer, Control and Communication*, pp. 1–6, IEEE, Karachi, Pakistan, February 2009.

[23] S. Choi, H. Park, H.-i. Lim, and T. Han, "A static API birthmark for Windows binary executables," *Journal of Systems and Software*, vol. 82, no. 5, pp. 862–873, 2009.

[24] H. Park, S. Choi, H.-i. Lim, and T. Han, "Detecting java theft based on static API trace birthmark," in *Advances in Information and Computer Security*, vol. 5312, pp. 121–135, Springer, Berlin, Heidelberg, 2008.

[25] H. Park, S. Choi, H.-i. Lim, and T. Han, "Detecting code theft via a static instruction trace birthmark for Java methods," in *Proceedings of the 6th IEEE International Conference on Industrial Informatics*, pp. 551–556, Daejeon, South Korea, July 2008.

[26] H.-i. 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.

[27] J. Yang, J. Wang, and D. Li, "Detecting the theft of natural language text using birthmark," in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 1–4, Pasadena, CA, USA, December 2006.

[28] T. Kakimoto, A. Monden, Y. Kamei, H. Tamada, M. Tsunoda, and K.-i. Matsumoto, "Using software birthmarks to identify similar classes and major functionalities," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, Shanghai, China, 2006.

[29] G. Myles and C. Collberg, "Detecting software theft via whole program path birthmarks," in *Information Security*, vol. 3225, pp. 404–415, Springer, Berlin, Heidelberg, 2004.

[30] A. Aiken, *Moss: A System for Detecting Software Plagiarism*, University of California, Berkeley, CA, USA, 1994.

[31] G. Whale, "Identification of program similarity in large populations," *The Computer Journal*, vol. 33, no. 2, pp. 140–146, 1990.

[32] M. J. Wise, "Detection of similarities in student programs: yap'ing may be preferable to plague'ing," in *Proceedings of the 23rd SIGCSE Technical Symposium*, St. Louis, Missouri, USA, March 1992.

[33] S. Schleimer, D. Wilkerson, and A. Aiken, "Winnowing: local algorithms for document fingerprinting," in *Proceedings of 2003 SIGMOD Conference*, San Diego, CA, USA, June 2003.

[34] Z. Tian, Q. Zheng, T. Liu, M. Fan, X. Zhang, and Z. Yang, "Plagiarism detection for multithreaded software based on thread-aware software birthmarks," in *Proceedings of the 22nd International Conference on Program Comprehension*, Hyderabad, India, May 2014.

[35] Z. Tian, Q. Zheng, T. Liu, and M. Fan, "DKISB: dynamic key instruction sequence birthmark for software plagiarism

detection," in *Proceedings of the IEEE International Conference on High Performance Computing and Communications & IEEE International Conference on Embedded and Ubiquitous Computing*, pp. 619–627, Zhangjiajie, China, November 2013.

[36] S. Nazir, S. Shahzad, Q. U. A. Nizamani, R. Amin, M. A. Shah, and A. Keerio, "Identifying software features as birthmark," *Sindh University Research Journal*, vol. 47, no. 3, pp. 535–540, 2015.

[37] S. Nazir, S. Shahzad, S. A. Khan, N. Binti Alias, and S. Anwar, "A novel rules based approach for estimating software birthmark," *The Scientific World Journal*, vol. 2015, Article ID 579390, 8 pages, 2015.

[38] D. Lee, Y. Choi, Y. Choi, J. Jung, J. Kim, and D. Won, "efficient categorization of the instructions based on binary excutables for dynamic software birthmark," *International Journal of Information and Education Technology*, vol. 5, no. 8, pp. 571–576, 2015.

[39] S. Won, S. Shahzad, and S. B. S. Abid, "Selecting software design based on birthmark," *Life Science Journal*, vol. 11, no. 12s, pp. 89–93, 2014.

[40] J. Choi, Y. Han, S.-j. Cho, HaeYoungYoo, and J. Woo, "A static birthmark for MS Windows applications using import address table," in *Proceedings of the Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 129–134, Taichung, Taiwan, July 2013.

[41] BSA, *The Compliance Gap BSA Global Software Survey*, Business Software Alliance, Washington, DC, USA, 2014.

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

[43] Y. Zeng, F. Liu, X. Luo, and S. Lian, "Abstract interpretation-based semantic framework for software birthmark," *Computers & Security*, vol. 31, no. 4, pp. 377–390, 2012.

[44] Y. Mahmood, Z. Pervez, S. Sarwar, and H. F. Ahmed, "Similarity level method based static software birthmarks," in *Proceedings of the High Capacity Optical Networks and Enabling Technologies*, pp. 205–210, Penang, Malaysia, November 2008.

[45] Y. Wang, F. Liu, Z. Zhao, B. Lu, and X. Xie, "Operand stack dependence based java static software birthmark," in *Proceedings of the 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Shenyang, China, July 2013.

[46] X. Xie, F. Liu, B. Lu, and L. Chen, "A software birthmark based on weighted k-gram," in *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent System (ICIS)*, pp. 400–405, Xiamen, China, October 2010.

[47] G. Myles and C. Collberg, "K-gram based software birthmarks," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, New Mexico, 2005.

[48] S. Nazir, S. Shahzad, R. Wirza et al., "Birthmark based identification of software piracy using Haar wavelet," *Mathematics and Computers in Simulation*, vol. 166, pp. 144–154, 2019.

[49] S. Shahzad, S. Shahzad, and N. Mukhtar, "Software birthmark design and estimation- A systematic literature review," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3905–3927, 2019.

[50] K. Fukuda and H. Tamada, "A dynamic birthmark from analyzing operand stack runtime behavior to detect copied software," in *Proceedings of the 2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 505–510, Honolulu, HI, USA, July 2013.

[51] Y. Bai, X. Sun, G. Sun, X. Deng, and X. Zhou, "Dynamic k-gram based software birthmark," in *Proceedings of the IEEE ASWEC 2008 19th Australian Conference*, Perth, WA, Australia, March 2008.

[52] D. G. Zill and M. R. Cullen, *Differential Equation S with Boundary Value Problem*, Brooks/Cole Cengage Learning, Boston, MA, USA, 7th edition, 2009.

[53] R. L. Burden and J. D. Faires, *Numerical Analysis*, Brooks/Cole, Cengage Learning, Boston, MA, USA, 9th edition, 2011.