Hindawi
Scientific Programming
Volume 2021, Article ID 5547766, 7 pages
https://doi.org/10.1155/2021/5547766

Research Article

Software Birthmark Usability for Source Code Transformation Using Machine Learning Algorithms

Keqing Guan,[1] Shah Nazir,[2] Xianli Kong,[3] and Sadaqat ur Rehman[4]

[1]*Institute for Big Data Research, Liaoning University of International Business and Economics, Dalian 116052, China*
[2]*Department of Computer Science, University of Swabi, Swabi, Pakistan*
[3]*School of Economics, Dongbei University of Finance & Economics, Dalian 116025, China*
[4]*Department of Computer Science, Namal Institute, Mianwali 42250, Pakistan*

Correspondence should be addressed to Shah Nazir; snshahnzr@gmail.com and Xianli Kong; kongxianli@dufe.edu.cn

Received 14 January 2021; Revised 24 January 2021; Accepted 30 January 2021; Published 9 February 2021

Academic Editor: Sikandar Ali

Source code transformation is a way in which source code of a program is transformed by observing any operation for generating another or nearly the same program. This is mostly performed in situations of piracy where the pirates want the ownership of the software program. Various approaches are being practiced for source code transformation and code obfuscation. Researchers tried to overcome the issue of modifying the source code and prevent it from the people who want to change the source code. Among the existing approaches, software birthmark was one of the approaches developed with the aim to detect software piracy that exists in the software. Various features are extracted from software which are collectively termed as "software birthmark." Based on these extracted features, the piracy that exists in the software can be detected. Birthmarks are considered to insist on the source code and executable of certain programming languages. The usability of software birthmark can protect software by any modification or changes and ultimately preserve the ownership of software. The proposed study has used machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors (K-NN) algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research.

1. Introduction

Source code transformation is performed in a manner in which the source code of a program is transmuted by spotting any operation for creating an alternative or nearly same program. This is mostly performed in situation of piracy where the pirates want the ownership of the software program. From different perspectives, the transformed source code is mostly equivalent to the original program in terms of semantics. For transforming the source into another program, one usually needs the incorporation of whole front end of programming language, data structure of internal program representation, parsing of source code, understanding of the program, meaningful static analysis, and generation of useable source code for representation of program. Software industry is immensely in front of the software piracy issues. This piracy performed in software can badly affect the software business and eventually big loss to the owner organizations. Stoppage of software piracy is extremely important for the rising economy of the software industry. Different methods are used to prevent piracy of software. These methods include techniques of fingerprinting [1, 2], watermarking [3–5], and software birthmarks [6–11]. The watermark has the weaknesses as it can be removed by approaches of code obfuscations and semantic preserving transformation. The similar concerns are existing in the software fingerprints. To overawe these limitations, the idea of birthmark was presented and is broadly acknowledged and known approach for preventing source code transformation and piracy of software.

Birthmark of software is considered as necessary features which can be employed for focusing the identification and uniqueness of software. The common uses of birthmarks are for software theft, identification of transformations in source code, and Windows API. More features of a software birthmark can eventually present the robustness and effectiveness which will further show the precise detection of transformations or theft made in the software or program. Birthmark of software is established on imperative properties, resilience, and credibility [6]. Credibility depicts that the birthmark of software entails that two programs, which is written independently, should be different. Whereas, the resilience should be preserved and not be damaged in any case. Various approaches were considered to show the effectiveness and usability of software birthmark [6, 12–14]. These approaches talk about various applications of software birthmark including source code transformation, code obfuscations, software theft, piracy, and many others. The use of software birthmark can protect software by any adaptation and ultimately preserve the ownership of software.

The proposed study endeavored to use machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research.

This study is divided into different sections. Section 2 represents the related work associated to the existing approaches of source code transformation, software theft, and so on. The research methodology of the proposed study is presented in Section 3. Results and Discussion are given in Section 4. The study is concluded in Section 5.

2. Related Work

Researchers frequently attempt to devise diverse approaches, methods, and solutions to proficiently and successfully analyze the source code transformation and software piracy. Numerous practices have been adopted in software industry to detect and prevent software theft. The idea of software birthmark was presented to overcome the downsides of software fingerprint, watermarks, and digital signature as these can be modified or removed by using approaches of code obfuscation and transformation of semantic preservation. Birthmark of software was established to powerfully recognize the software theft. First, the birthmark was developed by Tamada et al. [15], which extracts four types of birthmarks: inheritance structure, sequence of method calls, constant values in field variables, and the used classes. With the advancements in the field, birthmark was well thought out as a significant measure of the software in serving to identify software piracy. The field was discovered, and lots of researchers tried to grow a strong and further trustworthy birthmark for finding of software piracy. Software birthmark knowledge was initially considered as sole identification of object by Neufeld [16] in 1992. Derrick [17] discovered the idea and gave the importance to the use of birthmark details for "protecting" software. This was later termed as theft protection of software.

The primary software birthmark was allied with software theft which was offered as a birthmark for Java program theft detection [15]. In the same way in 2004, Tamada et al. considered birthmark of software that was used for detecting the theft in Windows applications. Myles and Collberg proposed "whole program path birthmarks" for detecting software theft [18]. That birthmark method was created on the whole control flow of the software program. Diverse categories of birthmark were planned for software theft detection. The proposed study identified a number of important birthmarks which have been proposed by different researchers for different purposes mostly for theft detection. Spafford and Weeber [19] discovered dissecting executable code for analyzing the structure of data, library calls, and system calls. The idea of software forensics was offered for thoughtful source of virus and malware infection. Birthmark was aimed to facilitate detection of transformation in source code and software theft [20–23]. These studies have considered the design of own birthmark according to some defined features of software and then evaluated the effectiveness in term of creditability and resilience of software for identification of theft exists in copies of software.

The authors [24] offered a dynamic program slicing tool built on dynamic birthmark with some inputs; a union of k-gram instruction-sequence sets as birthmark is used for identification of program. Formal description of software birthmarks was offered by Tamada et al. [25] where they proposed an approach of extracting birthmark from the class files of Java. They are sightseen on comparable perception and proposed a framework for evaluating the two significant properties of birthmarks that is resilience and credibility. Zeng et al. [26] devised a framework of semantic-based abstract interpretation for evaluating software birthmark. This model defines two important properties resilience and credibility. The success of the framework is confirmed by static API birthmark and static-gram birthmark. The authors presented a dynamic birthmark for Java that perceives how a program uses objects providing by the Java standard API [27].

For transmuting the source code into an alternative program, commonly, it needs the integration of data structure of internal program representation, whole front end of programming language, meaningful static analysis, parsing of source code, understanding of the program, and generation of useable source code for representation of the program. Various approaches are being used to change the source code. To overcome this issue, the researchers have devised different solutions. The proposed study has used machine learning algorithms for classification of the software birthmarks usability in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks.

3. Research Methodology

Efforts are made to overcome the issues raised from the transformation of source code and software theft.

Table 1: Software birthmark.

S. no.	Ref.	Approaches of software birthmark
1	[8]	Birthmark-based approach for intellectual software asset management
2	[9]	DKISB
3	[10]	Instruction-words based software birthmark
4	[11]	JSBiRTH
5	[29]	API call structure
6	[15]	CVFV, SMC, IS, and UC
7	[18]	Detecting software theft via whole program path birthmarks
8	[30]	Birthmark-based android application filtering
9	[31]	System call dependence graph
10	[24]	Dynamic k-gram-based software birthmark
11	[32]	k-gram-based software birthmarks
12	[33]	Dynamic key instruction sequences
13	[23]	Features-based software birthmark
14	[34]	Method-based static software birthmarks
15	[35]	CHI-based instruction-words based software birthmark
16	[36]	Thread-aware software birthmarks
17	[37]	System call-based birthmarks

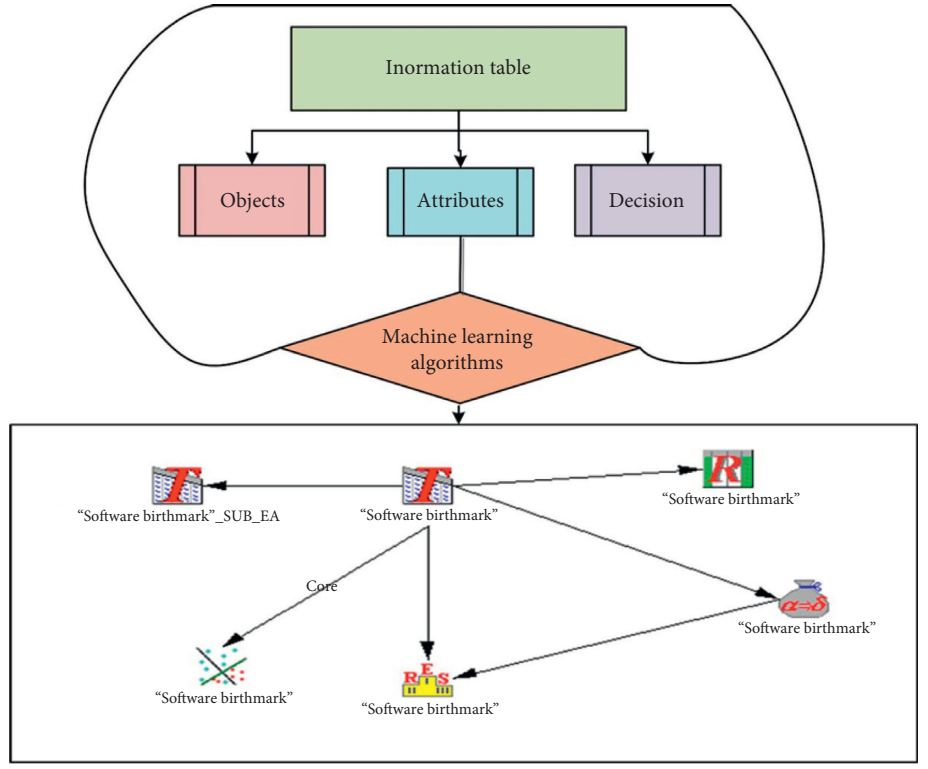

Figure 1: Flowchart of the proposed approach.

Researchers mostly considered software ownership and safety as one of the most priorities under consideration. A lot of research studies have been shown for shaping the idea of software birthmarks. Maximum of the birthmark approaches are related to Java source code, which are used for detecting Java theft. Further significant birthmark approaches and techniques works for Windows API [28], for detecting software theft. One of the significant notions used in describing a software birthmark is the usage of software features. Software can be divided into various parts (mostly features) of software [23]. Together, all these features of the software can deliver a faster and reliable identification of the software and then eventually be used for detection of theft. To detect transformation in source code or software theft, the birthmarks of software applications are matched, and similar birthmark identifies software piracy. A number of birthmarks were identified in the literature. The details are given in Table 1.

Figure 1 represents the flowchart of the approach used in the proposed study for software birthmark usability for source code transformation. The figure represents the information table (dataset) containing objects, attributes,

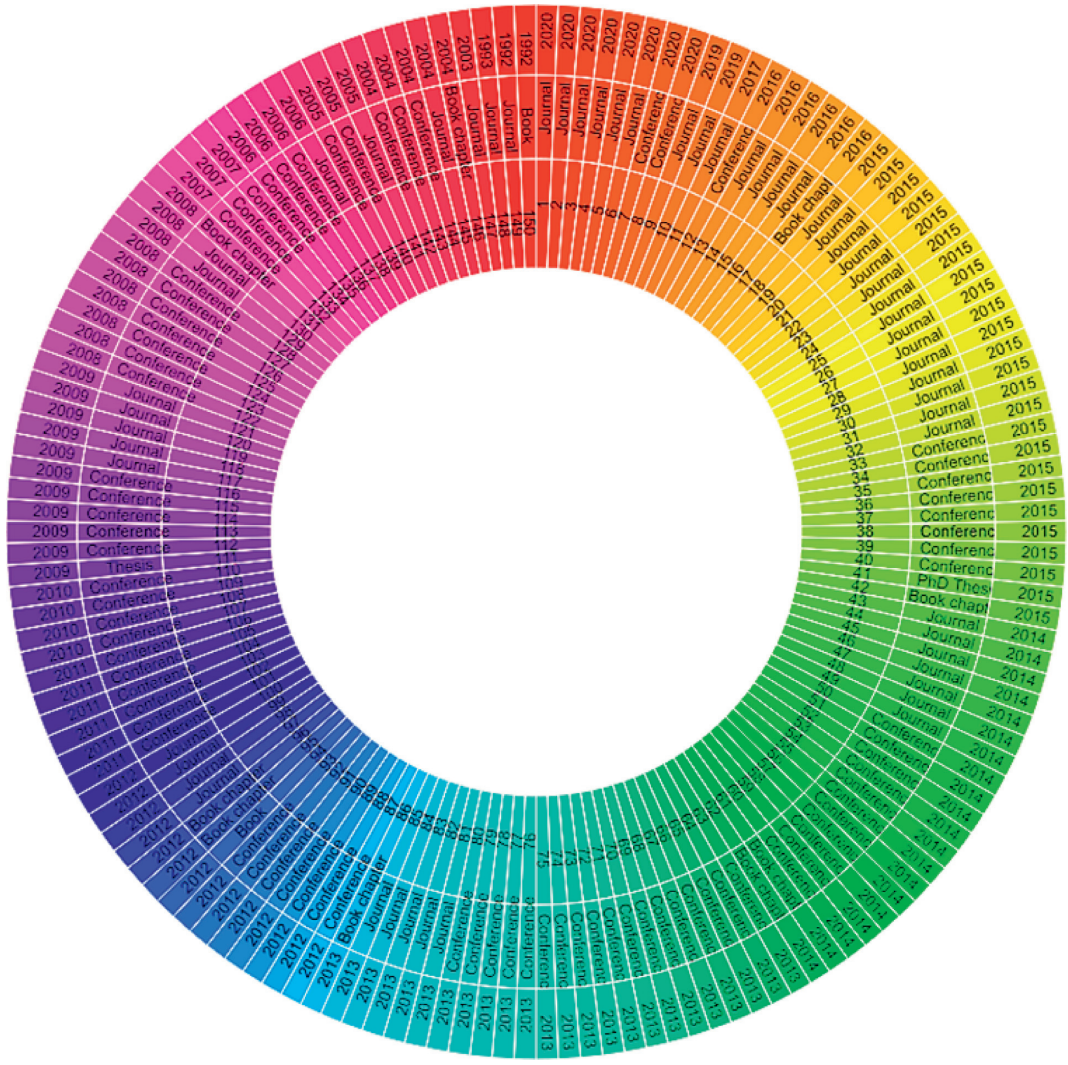

Figure 2: Dataset visualization.

and their decision. After the information table, the machine learning algorithms were applied. The K-NN algorithm was applied for the classification purpose. After that, the algorithms of decision rules, decomposition tree, and LTF-C algorithms were applied as cross-validation algorithms.

The dataset developed during higher studies programme was considered for validation purpose of the proposed research. Total of 150 entries were existing with three features. Figure 2 shows the visualization of the dataset for user understanding.

Once the information table was imported to the proposed system, initially, the reduct was applied. After that, rules set were generated. Figure 3 depicts the publications with the year in the given dataset.

Figure 4 depicts the rules set generated from the proposed study.

After doing this process, in last, the algorithm of K-NN was applied to the proposed research. Some cross-validation algorithms were used which are discussed in the Results and Discussion Section.

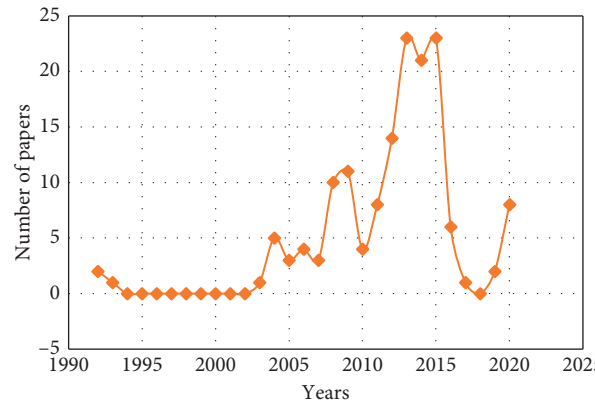

Figure 3: Publications with years based on dataset.

4. Results and Discussion

Several research studies have been conducted for refining software birthmarks for detection of piracy. The existing approaches used for detection of piracy are given as

Rule set: "Software birthmark"

(1-109)	Match	Decision rules
1	8	(Year=2020)=>("Paper type"={Conference[8]})
2	6	(Year=2016)=>("Paper type"={Conference[6]})
3	6	(Year=2011)=>("Paper type"={Conference[6]})
4	4	(Year=2010)=>("Paper type"={Conference[4]})
5	3	(Year=2007)=>("Paper type"={Conference[3]})
6	2	(Year=2019)=>("Paper type"={Journal[2]})
7	1	(Year=2013)&(Reference=63)=>("Paper type"={Conference[1]})
8	1	(Year=2013)&(Reference=64)=>("Paper type"={Conference[1]})
9	1	(Year=2013)&(Reference=65)=>("Paper type"={Conference[1]})
10	1	(Year=2013)&(Reference=66)=>("Paper type"={Conference[1]})

Figure 4: Rules set supported.

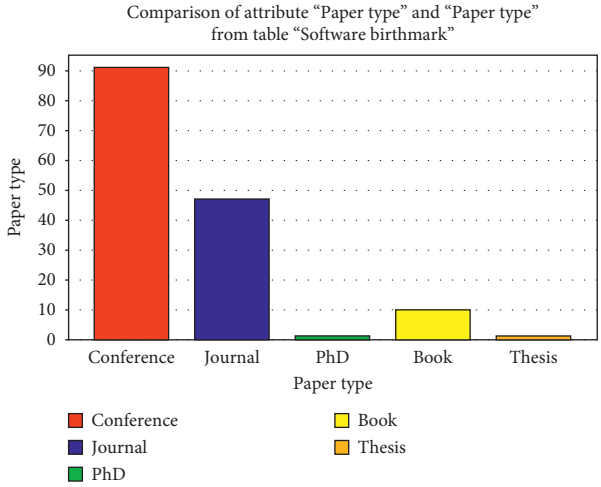

Figure 5: Frequencies of the dataset used.

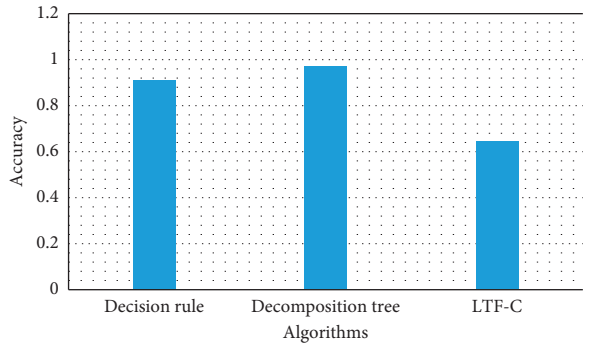

Figure 6: Algorithm used along with the accuracy.

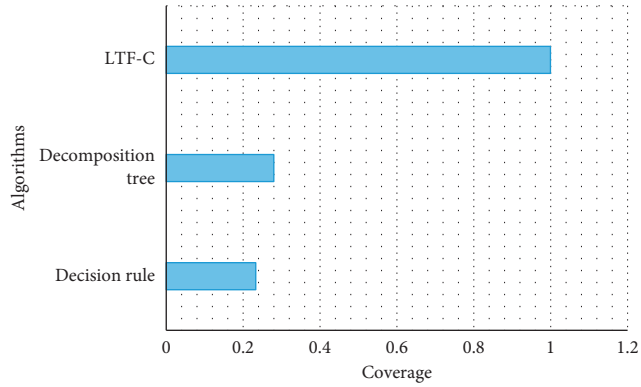

Figure 7: Coverage of the algorithms used.

intellectual software asset management [8], detection of software theft [18], plagiarism detection [38], detecting java theft [39], detecting binary theft [40], semantics-based repackaging detection for mobile apps [41], malware detection [42], detecting code theft [43], detecting the theft of natural language [44], credible, resilient, and scalable detection of software plagiarism using authority histograms [45], detecting plagiarized mobile apps [46], efficient similarity measurement technique of Windows software [47], detecting common modules in Java packages [48], measuring similarity of android applications [49], identify similar classes and major functionalities [50], moreover, for the source code level [48], and so on.

The proposed study has used the application of machine learning for software birthmark usability for transformation of source code. Initially, the K-nearest neighbors algorithm was used for classification of the software birthmarks. The experimental results of K-NN were effective and showed an accuracy of 98%. Figure 5 represents the frequencies of the dataset in term of conference, journals, books, and thesis.

Figure 6 shows the comparisons of the algorithms used in the proposed research. The algorithm decision rule has 0.91%, decomposition tree algorithm is having 0.96%, and the LTF-C algorithm is having 0.64% accuracy.

Figure 7 graphically represents the coverage of the algorithms used.

5. Conclusion

Software industry is growing with the passage of time. New innovations are offered to cater diverse issues of real life. The role of software applications has evidenced the success of software industry. Pirates are engaged with code transformations and gaining profit from the code obfuscation, transformation of source code, and piracy of software. This is mostly carried out in situations of piracy where the pirates want the ownership of the software program. Various approaches are being practiced for source code transformation

and code obfuscation. Among the present approaches, software birthmark was one of the approaches developed with the aim to detect software piracy exists in the software. Birthmarks are considered to insist on the source code and executable of certain programming languages. The proposed study has used machine learning algorithms for classification of the usability of existing software birthmarks in terms of source code transformation. The K-nearest neighbors algorithm was used for classification of the software birthmarks. For cross-validation, the algorithms of decision rules, decomposition tree, and LTF-C were used. The experimental results show the effectiveness of the proposed research. The algorithm decision rule has 0.91%, decomposition tree algorithm is having 0.96%, and the LTF-C algorithm is having 0.64% accuracy.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was sponsored in part by the Research Fund for the Doctoral Program of Liaoning University of International Business and Economics (2019XJLXBSJJ002).

References

[1] C. Gottschlich, "Curved-region-based ridge frequency estimation and curved gabor filters for fingerprint image enhancement," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 220–227, 2012.

[2] 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.

[3] 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.

[4] Y. Zeng, F. Liu, X. Luo, and C. Yang, "Software watermarking through obfuscated interpretation: implementation and analysis," *Journal of Multimedia*, vol. 6, no. 4, pp. 329–340, 2011.

[5] 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.

[6] 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.

[7] T. Yokoi and H. Tamada, "A beforehand extraction method for dynamic software birthmarks using unit test codes," in *Proceedings of the 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 169–175, Busan, South Korea, June 2018.

[8] D. Kim, J. Moon, S. J. Cho et al., "A birthmark-based method for intellectual software asset management," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, Siem Reap, Cambodia, January 2014.

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

[10] L. Ma, Y. Wang, F. Liu, and L. Chen, "Instruction-words based software birthmark," in *Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security*, Nanjing, China, November 2012.

[11] P. P. F. Chan, L. C. K. Hui, and S. M. Yiu, "JSBiRTH: dynamic JavaScript birthmark based on the run-time heap," in *Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference*, Munich, Germany, July 2011.

[12] S. Nazir, S. Shahzad, and L. S. Riza, "Birthmark-based software classification using rough sets," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 859–871, 2016.

[13] S. Nazir, S. Shahzad, R. B. Atan, and H. Farman, "Estimation of software features based birthmark," *Cluster Computing-The Journal of Networks Software Tools and Applications*, vol. 21, no. 1, pp. 1–14, 2017.

[14] S. Nazir, 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.

[15] H. Tamada, M. Nakamura, and A. Monden, "Design and evaluation of birthmarks for detecting theft of java programs," in *Proceedings of the IASTED International Conference on Software Engineering*, pp. 17–19, Innsbruck, Austria, February 2004.

[16] G. Neufeld, "Descriptive name resolution," *Computer Networks and ISDN Systems*, vol. 23, no. 4, pp. 211–227, 1992.

[17] G. Derrick, *Protection of Computer Software: Its Technology and Application*, p. 224, Cambridge University Press, New York, NY, USA, 1992.

[18] G. Myles and C. Collberg, "Detecting software theft via whole program path birthmarks," in *Proceedings of the Information Security: 7th International Conference, ISC 2004*, Palo Alto, CA, USA, September, 2004.

[19] E. H. Spafford and S. A. Weeber, "Software forensics: can we track code to its authors?" *Computers & Security*, vol. 12, no. 6, pp. 585–595, 1993.

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

[21] S. Nazir, "Design and estimation of features based software birthmark," Ph. D. Thesis, University of Peshawar, Peshawar, Pakistan, 2015.

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

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

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

[25] H. Tamada, M. Nakamura, A. Monden, and K.-I. Matsumoto, "Detecting the theft of programs using birthmarks," Technical Report NAIST-IS-TR2003014, Nara Institute of Science and Technology, Ikoma, Japan, 2003.

[26] 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.

[27] D. Schuler, V. Dallmeier, and C. Lindig, "A dynamic birthmark for java," in *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*, Atlanta, GA, USA, November 2007.

[28] H. Tamada, K. Okamoto, M. Nakamura, A. Monden, and K.-I. Matsumoto, "Dynamic software birthmarks to detect the theft of windows applications," in *Proceedings of the International Symposium on Future Software Technology*, Xi'an, China, October 2004.

[29] S. Choi, H. Park, H.-I. Lim, and T. Han, "A static birthmark of binary executables based on API call structure," in *Proceedings of the 12th Asian Computing Science Conference on Advances in Computer Science: Computer and Network Security*, Doha, Qatar, December 2007.

[30] S. Kang, H. Shim, S.-J. Cho, M. Park, and S. Han, "A robust and efficient birthmark-based android application filtering system," in *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, Towson, MD, USA, October 2014.

[31] K. Liu, T. Zheng, and L. Wei, "A software birthmark based on system call and program data dependence," in *Proceedings of the 2014 11th Web Information System and Application Conference*, Tianjin, China, September 2014.

[32] G. Myles and C. Collberg, "k-gram based software birthmarks," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, NM, USA, March 2005.

[33] Z. Tian, Q. Zheng, T. Liu, M. Fan, E. Zhuang, and Z. Yang, "Software plagiarism detection with birthmarks based on dynamic key instruction sequences," *IEEE Transactions on Software Engineering*, vol. 41, no. 12, pp. 1217–1235, 2015.

[34] 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 Computer, Control and Communication, 2009. IC4 2009*, Karachi, Pakistan, February 2009.

[35] Y. Wang, F. Liu, D. Gong, B. Lu, and S. Ma, "CHI based instruction-words based software birthmark selection," in *Proceedings of the Fourth International Conference on Multimedia Information Networking and Security*, Nanjing, China, November 2012.

[36] 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, June 2014.

[37] X. Wang, Y.-C. Jhi, S. Zhu, and P. Liu, "Detecting software theft via system call based birthmarks," in *Proceedings of the Computer Security Applications Conference*, pp. 149–158, Honolulu, HI, USA, December 2009.

[38] D.-K. Chae, S.-W. Kim, J. Ha, S.-C. Lee, and G. Woo, "Software plagiarism detection via the static API call frequency birthmark," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, Coimbra, Portugal, March 2013.

[39] H. Park, S. Choi, H.-I. Lim, and T. Han, "Detecting java theft based on static API trace birthmark," in *Proceedings of the Advances in Information and Computer Security, Third International Workshop on Security, IWSEC 2008*, Kagawa, Japan, November 2008.

[40] S. Park, H. Kim, J. Kim, and H. Han, "Detecting binary theft via static major-path birthmarks," in *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, Towson, MD, USA, October 2014.

[41] Q. Guan, H. Huang, W. Luo, and S. Zhu, "Semantics-based repackaging detection for mobile apps," in *Proceedings of the Engineering Secure Software and Systems: 8th International Symposium, ESSoS 2016*, London, UK, April 2016.

[42] S. Vemparala, F. D. Troia, V. A. Corrado, T. H. Austin, and M. Stamo, "Malware detection using dynamic birthmarks," in *Proceedings of the 2016 ACM on International Workshop on Security and Privacy Analytics*, New Orleans, LA, USA, March 2016.

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

[44] 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*, Pasadena, CA, USA, December 2006.

[45] D.-K. Chae, J. Ha, S.-W. Kim, B. Kang, E. G. Im, and S. Park, "Credible, resilient, and scalable detection of software plagiarism using authority histograms," *Knowledge-Based Systems*, vol. 95, pp. 114–124, 2016.

[46] D. Kim, A. Gokhale, V. Ganapathy, and A. Srivastava, "Detecting plagiarized mobile apps using API birthmarks," *Automated Software Engineering*, vol. 23, no. 4, pp. 591–618, 2015.

[47] P. Daeshin, J. Hyunho, P. Youngsu, and H. JiMan, "Efficient similarity measurement technique of windows software using dynamic birthmark based on API," *Smart Media Journal*, vol. 4, no. 2, pp. 34–45, 2014, http:////KJD:ART002028547, in Korean.

[48] H. Park, 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, 2009, in English.

[49] J. Ko, H. Shim, D. Kim et al., "Measuring similarity of android applications via reversing and k-gram birthmarking," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, Montreal, Canada, October 2013.

[50] 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, May 2006.