Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 17, 2018

A REAL-TIME PLAGIARISM DETECTION TOOL FOR COMPUTER-BASED ASSESSMENTS

Heimo J. Jeske	Tshwane University of Technology, Pretoria, South Africa	JeskeHJ@tut.ac.za
Manoj Lall*	Tshwane University of Technology, Pretoria, South Africa	LallM@tut.ac.za
Okuthe P. Kogeda	Tshwane University of Technology, Pretoria, South Africa	KogedaPO@tut.ac.za

* Corresponding author

ABSTRACT

Aim/Purpose	The aim of this article is to develop a tool to detect plagiarism in real time amongst students being evaluated for learning in a computer-based assessment setting.
Background	Cheating or copying all or part of source code of a program is a serious concern to academic institutions. Many academic institutions apply a combination of policy driven and plagiarism detection approaches. These mechanisms are either proactive or reactive and focus on identifying, catching, and punishing those found to have cheated or plagiarized. To be more effective against plagiarism, mechanisms that detect cheating or colluding in real-time are desirable.
Methodology	In the development of a tool for real-time plagiarism prevention, literature review and prototyping was used. The prototype was implemented in Delphi programming language using Indy components.
Contribution	A real-time plagiarism detection tool suitable for use in a computer-based assessment setting is developed. This tool can be used to complement other existing mechanisms.
Findings	The developed tool was tested in an environment with 55 personal computers and found to be effective in detecting unauthorized access to internet, intranet, and USB ports on the personal computers.
Recommendations for Practitioners	The developed tool is suitable for use in any environment where computer-based evaluation may be conducted.

Accepted by Editor Keith Willoughby | Received: September 10, 2017 | Revised: December 14, 2017; January 19, 2018 | Accepted: February 13, 2018.
Cite as: Jeske, H. J., Lall, M., & Kogeda, O. P. (2018). A real-time plagiarism detection tool for computer-based assessments. *Journal of Information Technology Education: Innovations in Practice, 17,* 23-35.
https://doi.org/10.28945/3963

Recommendation for Researchers	This work provides a set of criteria for developing a real-time plagiarism prevention tool for use in a computer-based assessment.
Impact on Society	The developed tool prevents academic dishonesty during an assessment process, consequently, inculcating confidence in the assessment processes and respectability of the education system in the society.
Future Research	As future work, we propose a comparison between our tool and other such tools for its performance and its features. In addition, we want to extend our work to include testing for scalability of the tool to larger settings.
Keywords	plagiarism detection, plagiarism detection tools, computer-based assessment, quality education, dishonesty

INTRODUCTION

Over the last decade, the universities in South Africa have experienced a paradigm shift, in terms of higher education, from a focus on teaching to a focus on learning. Many mission statements can be found that claim their institution to be student-centered or learner-centered institutions. Besides this rhetoric, possibly the most notable change that can be observed is a greater emphasis on skills development. The requirement of skills development placed on institutions of higher learning purposefully demands that appropriate skills be identified by analyzing its demand in the workplace. One set of skills that is in great demand by the software industry in South Africa is programming skills (Balwanz & Ngcwangu, 2016). In a quest to meet the computer programming skills shortage, many national and private institutions have either started offering such programs or increased their enrolment into such programs. Although this approach tries to address the issue of quantity of such a workforce, it also places responsibility on such institutions to ensure that the quality of the students being prepared is not compromised and meets the expectations of such a work environment (Dey & Sobhan, 2006).

In order to evaluate the quality of their students, a common mechanism used in many institutions of higher learning is to conduct assessments of learning, using mechanisms such as tests and examinations, in a supervised and controlled environment (Astin, 2012; Gibbs & Simpson, 2005). The controlled environment is to ensure that true capabilities and competencies of students are reflected. An important aspect of assessment of learning is to serve as a means of communication between the world of education and the society at large. For this communication to be publically accepted as a code of quality, the tests and examinations process must instill confidence in the marks obtained in those assessments. Establishing trustworthiness in an assessment of learning results (marks or grades) is directly related, though not limited, to minimizing academic dishonesty amongst the students (Biggs & Collis, 2014; Martins, Fonte, Henriques, & da Cruz, 2014). In this article, we present a tool to detect plagiarism in real time amongst students being evaluated for learning in a computer-based assessment setting. In the context of this research, computer-based assessments refer to all assessments conducted using a computer. This form of assessment is of particular importance when evaluating the learning of computer programming skills such as coding, debugging, and testing, which are difficult to assess using traditional paper based approaches. Additionally, computer-based assessment of learning builds confidence in the minds of prospective employers concerning the ability of the student to meet the skills-set required of a computer programmer. However, applying computer-based assessment brings with it additional challenges not commonly found in a traditional paper based assessment settings (Simon, Cook, Sheard, Carbone, & Johnson, 2013). For instance, source codes widely available on the Internet can easily be downloaded and used covertly, or files containing source code of one student can be shared between students on the same intranet. Figure 1 shows a typical computer-based assessment setting. It can be observed from the figure the ease with which a student could insert a memory stick into the computer's USB port and copy relevant

files and the difficulty an invigilator would face in identifying such illegal activities taking place. Having a tool for detecting such illegal activities in real-time would go a long way in discouraging students with intentions to copy during the assessment. A tool such as the one proposed by us will also assist in reducing the burden on teaching staff to perform a similarity check on the works handed in for evaluation.

Figure 1. A typical computer-based evaluation setting

LITERATURE REVIEW

In an effort to minimize cheating or copying all or part of a program source code from some sources and submitting as the student's own, institutions are constantly under pressure to implement ways to address such activities (Oberreuter & VeláSquez, 2013). Many academic institutions try to address them through policies and tools (Hodgkinson, Curtis, MacAlister, & Farrell, 2016; Martins et al., 2014). Tools often serve as plagiarism detection mechanisms. Many tools such as MOSS, JPlag, Covet, and CloneDr are commonly used for source code plagiarism detection. However, they are not very suitable for source codes plagiarism detection especially from the perspective of Rapid Application Development tools, which automatically supply a major part of the anticipated solution (Simon et al., 2013). Additionally, in a corpus of student assignments, source-code files may have a similar semantic meaning. This is inevitable since they are developed to solve a common programming problem and share common programming language keywords. Plagiarism detection is a reactive mechanism, which highlights similarities between various code snippets well after the commitment of dishonest actions (Agrawal & Sharma, 2016; Joy & Luck, 1999).

Policies and procedures, on the other hand, are a more proactive mechanisms aimed at discouraging plagiarism by highlighting punitive measures. They serve as a plagiarism prevention mechanism. In order to maintain high standards of academic integrity, it is essential to know and respect the policies concerning plagiarism, and to seek and foster a learning environment that encourages the development of academic skills that are appropriate for each discipline. The limited effectiveness of policies as a deterrent to plagiarism can be observed by a perpetual increase in incidence of academic dishonesty amongst students.

Prevention of plagiarism, if applied effectively, has a very strong impact but does not prove to be watertight to rule out all plagiarism or, more specifically, illegal source code copying (Devlin & Gray, 2007; Wilcox, Cameron, & Reber, 2015). On the contrary, plagiarism detection can narrow down potential cheaters, but it is not sophisticated enough to separate the cheaters from the non-cheaters (Lukashenko, Graudina, & Grundspenkis, 2007). All these mechanisms are either proactive or reactive and focus on identifying, catching, and punishing those found to have cheated or plagiarized. To be more effective against plagiarism, there is a need for mechanisms that detect cheating or colluding in real time, that is, during an assessment session. This will assist in implementing immediate preventive actions in accordance with the institutional policies, for instance, getting a confession letter

signed by the student thereby minimizing lengthy disciplinary hearing processes. In this article, we propose a software-based tool to detect copying or sharing of program codes during a practical computer-based programming skills assessment session by monitoring, sending alerts, and logging access to external resources. These external resources include source codes available on the internet, intranet, and external devices such as USB memory sticks. The monitoring, sending alerts to a manned computer, and logging of unauthorized activities are all done in a transparent manner to avoid disturbance to other students being assessed. Having such a system in place will enhance trust in our academic system by prospective employers of our students.

REQUIREMENTS OF THE SOFTWARE TOOL FOR COMPUTER-BASED ASSESSMENT

For a tool to be useful during assessment, it must fulfil the following functions.

Support monitoring of unauthorized access to external resources: Students may try to access resources that they think may be useful to them in answering the questions posed. These resources could be available on the internet or resources that the students have saved on the intranet with an intention to access them during the assessment session. Another common place to store information is on a USB memory stick. These devices, being physically small, are very easy to smuggle into the venue where the assessment is taking place and easy to insert into the workstations for accessing source codes covertly.

Support ease of identifying workstations accessing illegal resources: Accessing resources from the internet, intranet, or a tiny memory stick is very difficult to detect in an assessment setting where there are many students busy typing in their codes. It is of paramount importance that invigilators or any other person assigned the responsibility of ensuring the integrity of the assessment process is able to identify the exact workstation on which suspicious activity is taking place and take appropriate actions.

Support transparent logging of the illegal activities: An important aspect of policies on plagiarism are the punitive measures to be applied in cases of their violation. These punitive measures are a form of deterrent to perpetrators of these rather serious activities. It is important that that mechanism be in place to log all unauthorized accesses to be used as proof in an event of disciplinary hearing against the students.

There are a few commercial tools, such as NetOp School, Securexam Remote Proctor (SRP), and LanSchool that may be used for the monitoring of student activities on the network (Hage, Rademaker, & van Vugt, 2010). The SRP was initiated by Troy University for conducting trusted examinations at distant learning sites worldwide (Kitahara & Westfall, 2007). The SRP uses biometrics, such as fingerprints and facial photographs to authenticate the identity of the candidates. SRP works by allowing access to the computer-based examination and locking down the operating systems of the desktop to prevent access to all other non-examination materials or applications (Anderson & Gades, 2017). Although SRP could be successful in preventing illegal access to external resources, it is not suitable for use in a practical computer-based assessment session of programming skills where the students have to access the compilers for compiling the source codes. This has a limiting effect on the kind of skills assessment that academic institutions can conduct on its students.

LanSchool is another popular tool used by academic institutions for managing computer-based practical sessions (LanSchool, 2016). Some of the key features of LanSchool to prevent cheating during the assessment session is the keyboard monitoring and logging ability, and network tempering control (includes alert for disconnection) capabilities. However, it lacks the features to monitor external drive activities and tracking of network communication between peers on a network. Although 2-D mapping of the classroom is supported, its usefulness is limited due to not being able to explicitly send alerts to the manned computer when external resources are being accessed.

Another popular tool is the NetOp School (NetOp Vision, 2015) used for managing teaching related activities in a computer laboratory settings. It supports, amongst others, activities for sharing teachers' computer screen, observing students' computer screens, broadcasting information to all or selected students' screens, and permitting internet access. It also supports mirroring the physical computer layout in the computer laboratory for facilitating ease of identification. It however, lacks features for monitoring internal and external data access. No data flow alerts for local network traffic between peer stations are supported. A logging feature, though available, is limited in the sense that it is not possible to log external data access with information such as timestamp and workstation ID for use as proof at a later stage.

Based on the above discussion, it is observed that although there exist commercial tools that could be used for detecting cheating during an assessment session, they all seem to lack certain features. In the next section, we propose the conceptual model that defines the structure, behavior, and high-level views of our proposed tool (E-Proctor) that aims to minimize these limitations.

THE SOFTWARE TOOL AND ITS DESIGN

Based on the discussion presented above, for a system to be useful in preventing cheating or colluding during a computer-based assessment session, it must include the following features:

- Auto start and transparent monitoring
- Uniquely identify each computer on the network
- Provide fast (near real time) alerts to the monitoring station
- Log the details of transgression taking place
- Initiate actions on the client machine in accordance with the institutional rules.

Figure 2 depicts a high-level view of system functions performed at the clients and the monitoring station.

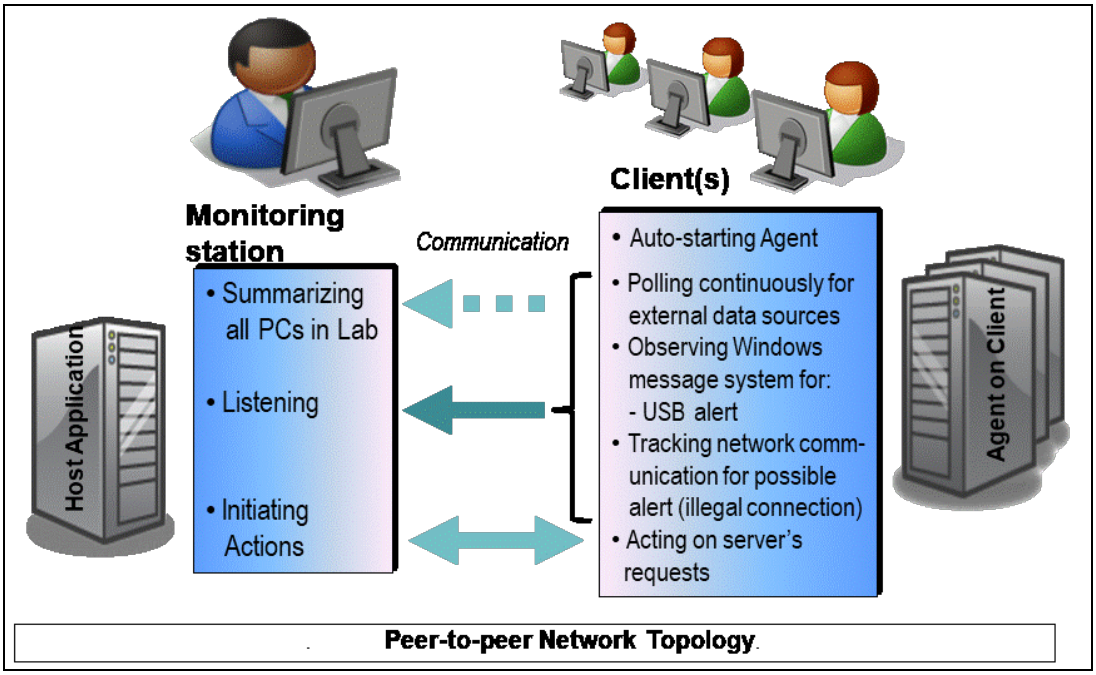

Figure 2. A high-level functional view of the E-Proctor system

Both the monitoring station and the client software were written in Delphi programming language using Indy components. The Indy components are based on the Microsoft's Winsock.dll library in

form of wrappers and are freely available. A high-level view of the various components of the E-Proctor is shown in Figure 3.

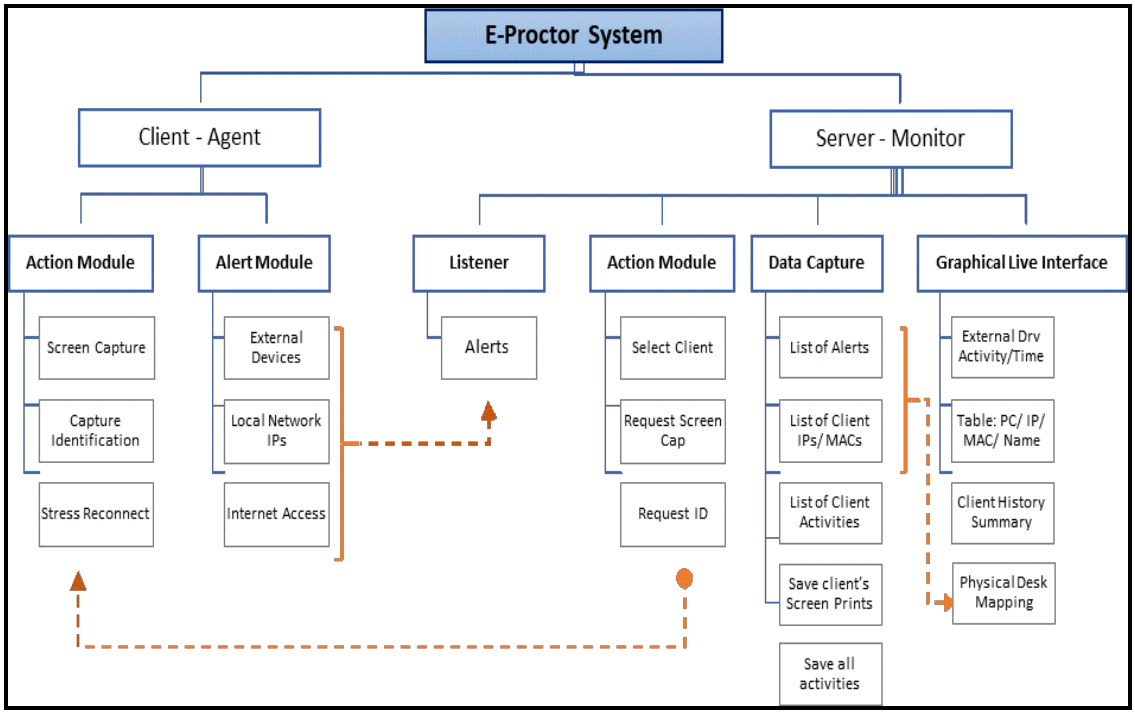

Figure 3. Major components of E-Proctor system

The details of the transparent monitoring process (see Figure 4) are as follows:

- The agent on the client side is activated by the operating system. The client then sends a *ClntReady* message to the server.
- The server then responds by sending *ClntReady* message back to the client. This confirms to the client that it is now connected to the server. The exchange of *ClntReady* between the client and server happens once every 2 seconds.
- If an alert occurs on the client side, then any outgoing *ClntReady* message will be replaced by a suitable alert message. For instance, if a memory stick is inserted into a USB port on the client machine then *AlertMemo* message will be sent to the monitoring server (Scenario A).
- The server then confirms the receipt of the alert message by sending *AlertMemo* back to the client. On receiving the acknowledgment from the server, the client prepares a suitable summary of activities taking place on the client machine and sends it to the server (in this case, *MFD* data). The server logs this information for later use. The nature of information logged on the server is depicted in Figure 6. On completing the logging process, the server sends a *ClntReady* message back to the client to indicate to the client to continue with its normal monitoring process.

If the server wishes to communicate a request to a selected client, the following message exchange sequence takes place (Scenario B):

- The default incoming message *ClntReady* is replaced with *ScreenCap* message.
- On receiving the *ScreenCap* message from the server, the client suspends its normal activities (i.e., sends *ClntReady*) and starts to act in accordance with the servers request. For example, the *ScreenCap* message is a request to capture the desktop screen and send the image file (jpg file) to the server.
- The server receives the jpg-file and stores it for later use.

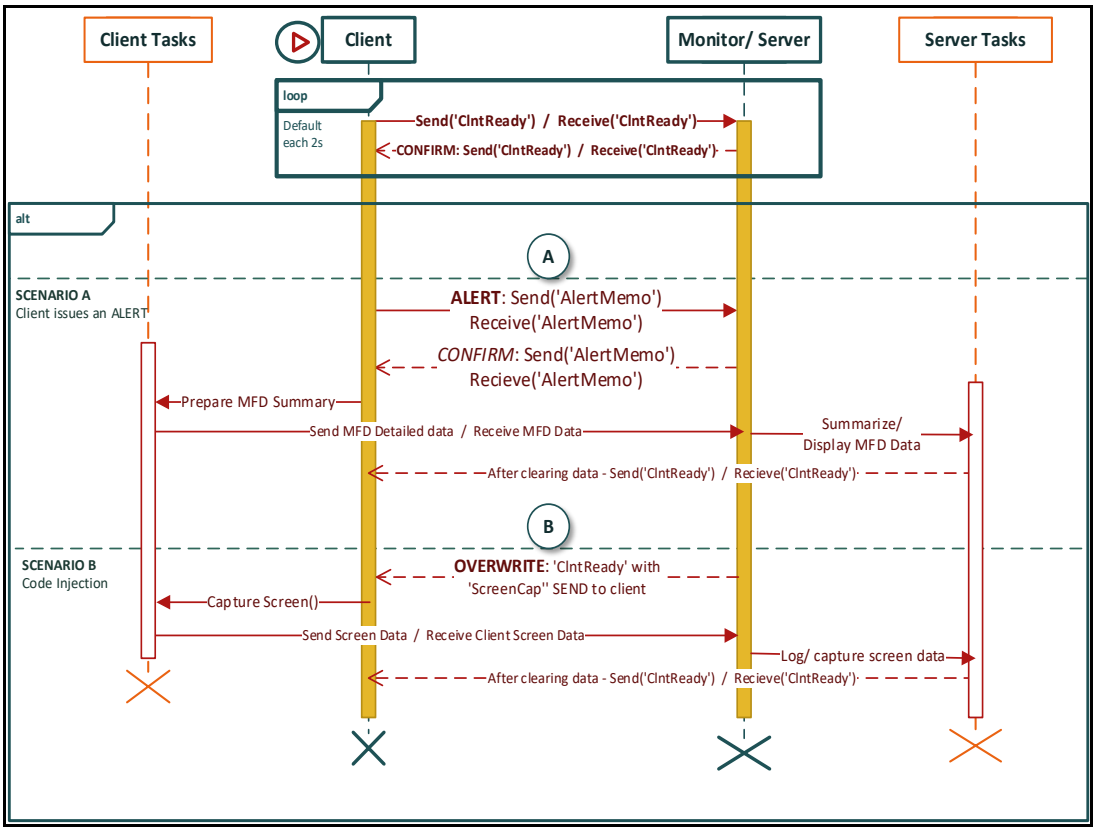

Figure 4. Sequence diagram of communication between client and the monitoring station

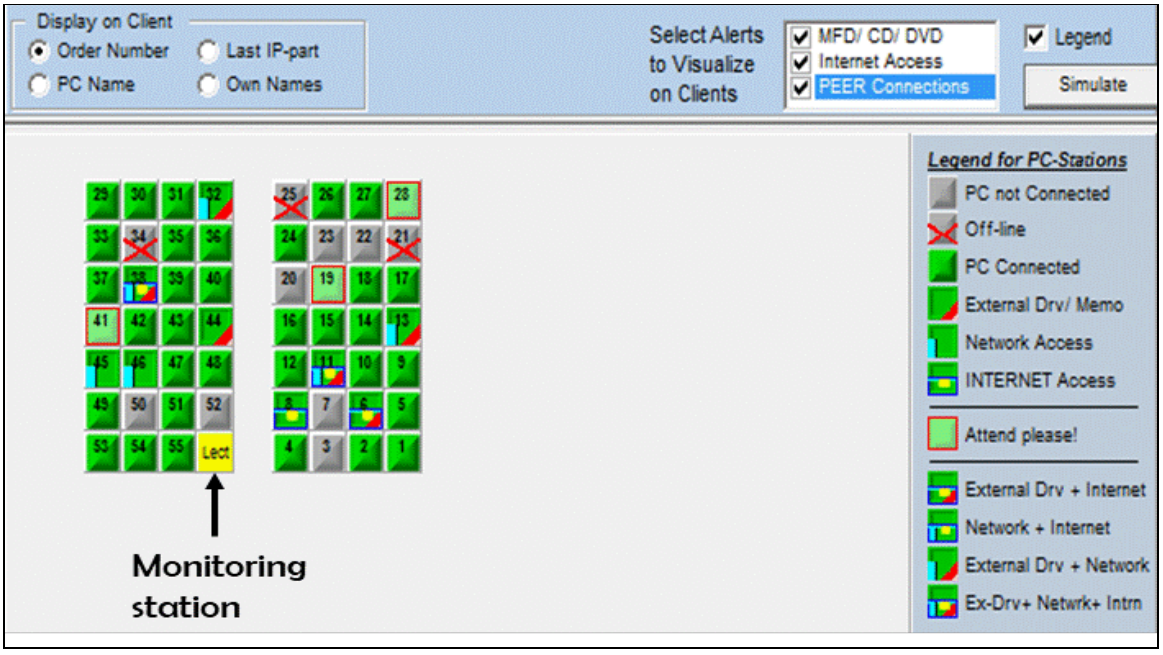

Figure 5. A 2D layout of the workstations in the assessment venue and the nature of activities taking place on them

To facilitate in the ease of identifying the person trying to access unauthorized material, a two dimensional (2D) image mapping the physical setup is essential. This 2D image of the layout is dis-

played on the monitoring station (see Figure 5). In our application, the statuses for the following operations are displayed on the monitoring station:

- Client not connected at all (e.g., station 3).
- PC station log off or disconnection during a session (e.g., station 21).
- Active connection established – Station in ready state with user logged on (e.g., station 1).
- External data device connected, for example, a memory stick (e.g., station 44).
- Access to peer station established (e.g., station 45).
- Access to internet address established (e.g., station 8).

The display on the monitoring station alerts the person in charge of maintaining the integrity of the assessment to initiate actions according to the institutional guidelines. To discourage plagiarism by using punitive measures, the evidence of illegal activities needs to be logged for later use. In addition to the information shown on the monitoring station, more details such as the time and the duration of illegal activity taking place need to be captured. The details of the information captured in our application are shown in Figure 6.

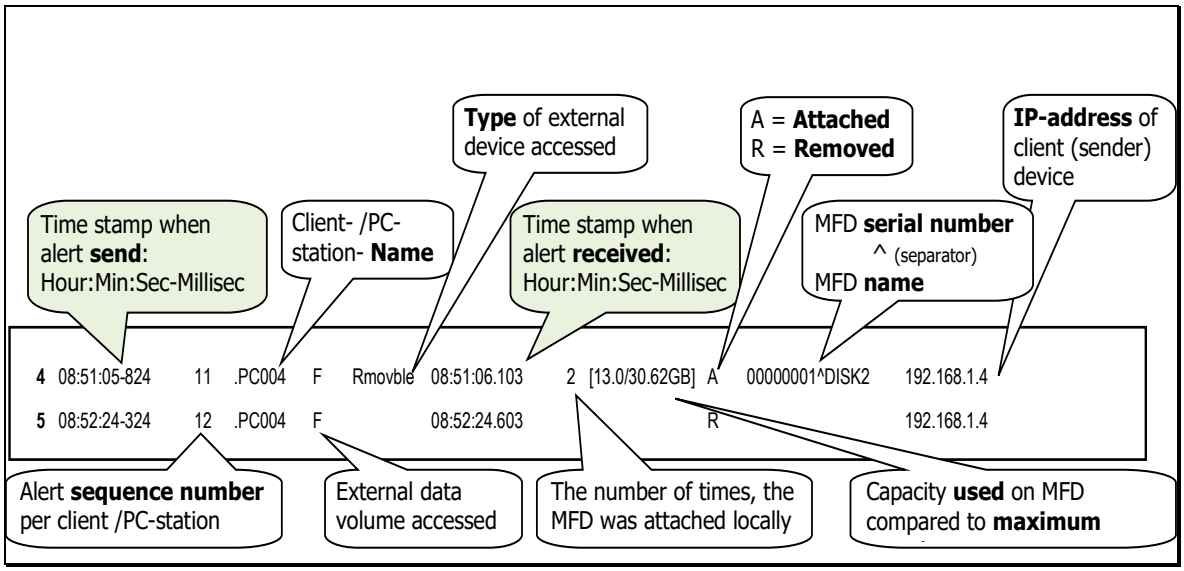

Figure 6. Information logged for later use

A brief explanation of the information captured (refer to Figure 4) from an arbitrary workstation, station 4 in this case, follows. Client workstation PC004 with IP address of 192.168.1.4 tries to access a removable external device at 08:51:05-824. This information is displayed on the monitoring station about a quarter of a second later (08:51:06-103 minus 08:51:05-824 = 279 milliseconds). The external device was connected to F: drive. It can be observed from Figure 6, that the removable device was accessed twice between 08:51:06-103 and 08:51:05-824 and the device contains 13.0 GB of information but could hold 30.62 GB. In addition, the "A" and "R" represents when the removal device was attached to the client workstation and when it was removed. For positive identification of the removal device, its make and serial number are recorded. This information can be of great assistance if disciplinary actions are taken against the perpetrators.

A comparison between the features supported by the commonly used tools in support of computer-based assessment and E-Proctor is presented in Table 1.

Table 1. A comparison of commonly used tool features for its support in computer based assessment

Software Product ➔ **Scale used:** X = Absent/Non-existent support √/2 = Limited support √ = Adequate supported **Attributes for Comparison**	Securexam Remote Proctor™	NetOp Vision Pro/ NetOp School*	LanSchool*	E-Proctor (prototype)
Monitoring of unauthorized assess (For external device, internet connection, local Peer)	X	X	X	√
Transparent logging or illegal activities	√/2	X	X	√
Ease of identifying workstation accessing illegal resource	√/2	X	√/2	√

RESULTS AND DISCUSSIONS

From a software point of view, our tool consists of two major components – the client component and the server component. The client software was installed on 55 workstations and the server was installed on one machine (the monitoring station). However, for testing the alert transfer rate from the client workstation to the monitoring station when inserting and removing a memory stick from the USB port, 26 randomly selected client workstations were used (Figure 7).

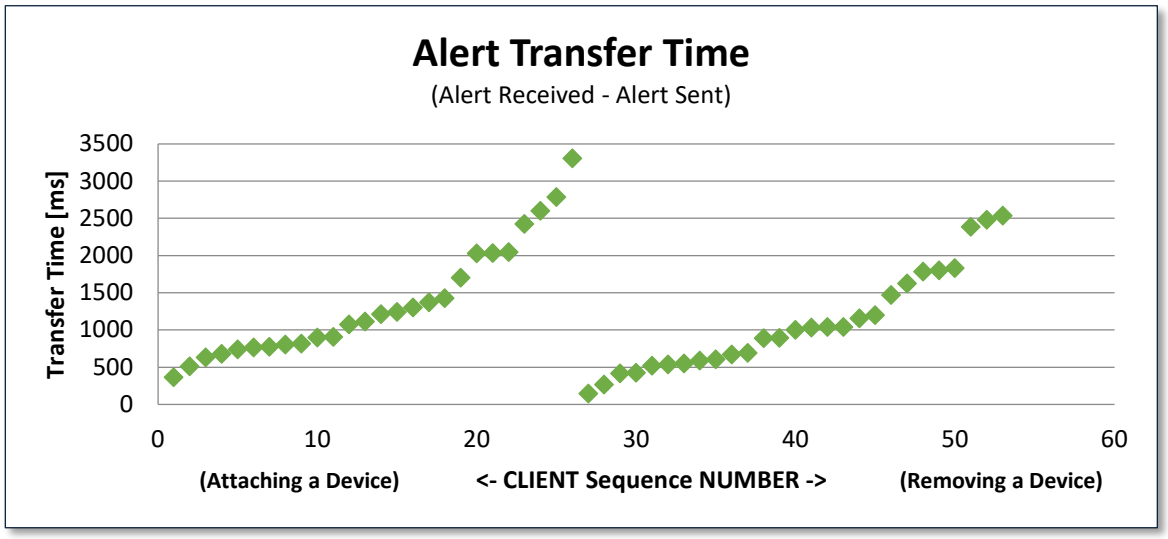

Figure 7. Alert transfer time from workstations to the monitoring station

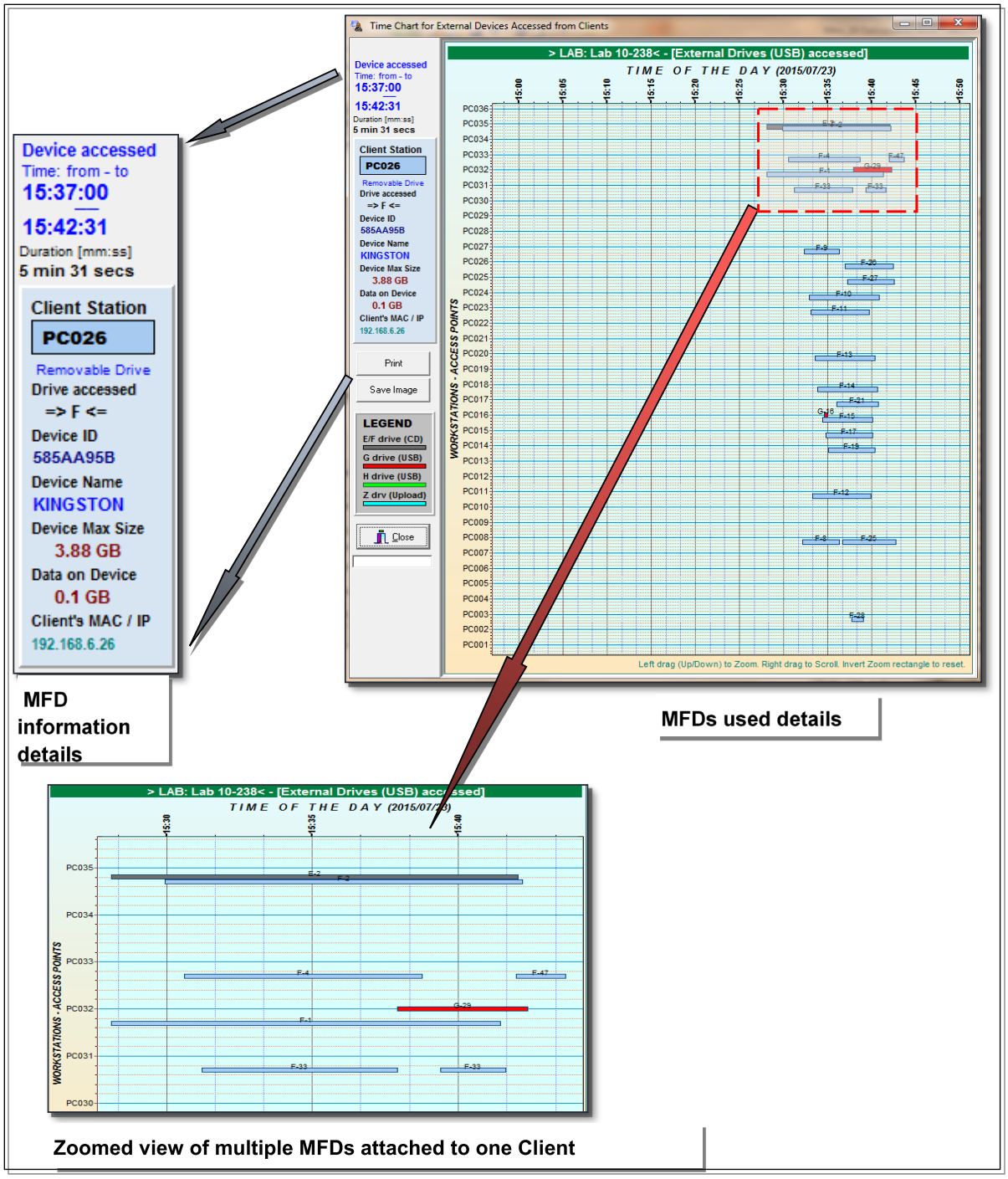

Figure 8. Details of report generated

The mean time for the alert transfer to the monitoring station, when a memory device is attached to the USB port, is 1226.72 milliseconds and it is less than half (594.1 milliseconds) when removing a device. The time difference is mainly due to the additional information (e.g., size of the memory device) that needs to be sent to the monitoring station when a device is attached.

Besides providing a mechanism for detecting access to unauthorized resources during an assessment, our tool also generates a detailed graphical report that may be used to trace the trespassers faster and more efficiently. Information related to the report is presented in Figure 8.

The report, in essence, captures information that links a particular wrongdoer to a certain unlawful action. Details such as the time and the duration of a particular unauthorized access are shown. Additionally, the report also contains information related to the client workstation (machine name and the IP address), the nature of the device that was accessed and the number of times an illegal activity took place as shown in Figure 8.

SUMMARY, CONCLUSIONS AND FUTURE WORK

In this current age, where nearly everything is connected to everything else, it has become increasingly easy to copy and paste source codes from other sources and pass it as one's own. Such actions, especially in an educational environment, have the potential of serious consequences. In an effort to enhance academic integrity, institutions rely extensively on policies and plagiarism detection tools. These mechanisms are either proactive or reactive and do not support real-time plagiarism detection in computer-based assessment settings.

In this article, we develop a tool that tries to supplement the proactive and the reactive approaches. Our tool, E-Proctor, was designed keeping the principles of transparent monitoring of unauthorized accesses, uniquely identifying each computer on the network, sending (near) real-time alerts to the monitoring station, logging the details of transgressions taking place, and initiating actions on the client machine in accordance with the institutional rules.

The tool was installed in a computer laboratory with 55 workstations and one monitoring station. The mean time taken for an alert to be sent to the monitoring station when memory sticks were inserted or removed for USB ports was found to be about 1.22 sec and 0.59 sec. In such a short time period very little, if any, plagiarism can take place. Hence, we consider this as a real-time detection.

A limitation of the proposed E-Proctor tool is it captures the IP address of a resource on the Internet rather than the URL (Uniform Resource Locator) of that resource. For example, if one of the client machines with IP address of 192.168.10.43 went onto the Internet and connected to an URL mapped to an IP address of 156.123.45.67 then the E-Proctor tool captures the IP address and not the URL. This may become a problem in trying to prove that a particular illegal resource was accesses as many resources can be hosted from a particular IP address.

As future work, we propose a comparison between our tool and other such tools for its performance and its features. In addition, we want to extend this work to include testing for scalability of the tool to larger settings.

REFERENCES

Agrawal, M., & Sharma, D. K. (2016). *A state of art on source code plagiarism detection*. Paper presented at the 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016. https://doi.org/10.1109/NGCT.2016.7877421

Anderson, C., & Gades, P. (2017). *Proctoring exams in an online environment*. Innovate! Teaching with Technology Conference 2107. University of Minnesota Morris. Minnesota, USA.

Astin, A. W. (2012). *Assessment for excellence: The philosophy and practice of assessment and evaluation in higher education*: Rowman & Littlefield Publishers.

Balwanz, D., & Ngcwangu, S. (2016). Seven problems with the 'scarce skills' discourse in South Africa. *South African Journal of Higher Education, 30*(2), 31-52. https://doi.org/10.20853/30-2-608

Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*: Academic Press.

Devlin, M., & Gray, K. (2007). In their own words: A qualitative study of the reasons Australian university students plagiarize. *High Education Research & Development, 26*(2), 181-198. https://doi.org/10.1080/07294360701310805

Dey, S. K., & Sobhan, M. A. (2006). *Impact of unethical practices of plagiarism on learning, teaching and research in higher education: Some combating strategies.* Paper presented at the 7th International Conference on Information Technology Based Higher Education and Training, 2006. ITHET'06. https://doi.org/10.1109/ITHET.2006.339791

Gibbs, G., & Simpson, C. (2005). Conditions under which assessment supports students' learning. *Learning and Teaching in Higher Education, 1*, 3-31.

Hage, J., Rademaker, P., & van Vugt, N. (2010). A comparison of plagiarism detection tools. *Technical Report UU-CS-2010-015,* Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands.

Hodgkinson, T., Curtis, H., MacAlister, D., & Farrell, G. (2016). Student academic dishonesty: The potential for situational prevention. *Journal of Criminal Justice Education, 27*(1), 1-18. https://doi.org/10.1080/10511253.2015.1064982

Joy, M., & Luck, M. (1999). Plagiarism in programming assignments. *IEEE Transactions on Education, 42*(2), 129-133. https://doi.org/10.1109/13.762946

Kitahara, R. T., & Westfall, F. (2007). Promoting academic integrity in online distance learning courses. *MERLOT Journal of Online Learning and Teaching, 3*(3), 265-276.

LanSchool. (2016). LanSchool V8.0. Retrieved from https://www.lenovosoftware.com/LanSchool_v8_Features%20Flyer1.pdf

Lukashenko, R., Graudina, V., & Grundspenkis, J. (2007). *Computer-based plagiarism detection methods and tools: an overview.* Paper presented at the 2007 International Conference on Computer Systems and Technologies. https://doi.org/10.1145/1330598.1330642

Martins, V. T., Fonte, D., Henriques, P. R., & da Cruz, D. (2014). *Plagiarism detection: A tool survey and comparison.* Paper presented at the OASIcs-OpenAccess Series in Informatics.

NetOp Vision. (2015). *NetOp Vision User's guide.* Retrieved from http://www.codework-systems.com/downloads/vision/manuals/netopvisionusersguide_en_73.pdf

Oberreuter, G., & VeláSquez, J. D. (2013). Text mining applied to plagiarism detection: The use of words for detecting deviations in the writing style. *Expert Systems with Applications, 40*(9), 3756-3763. https://doi.org/10.1016/j.eswa.2012.12.082

Simon, Cook, B., Sheard, J., Carbone, A., & Johnson, C. (2013). *Academic integrity: Differences between computing assessments and essays.* Paper presented at the 13th Koli Calling International Conference on Computing Education Research, Koli, Finland. https://doi.org/10.1145/2526968.2526971

Wilcox, D. L., Cameron, G. T., & Reber, B. H. (2015). *Public relations: Strategies and tactics*: Pearson New York, NY.

BIOGRAPHIES

Heimo Jeske obtained his BSc degree in Computer Science from the University of Namibia. He is currently a lecturer in the Department of Computer Science at Tshwane University of Technology, South Africa. His research interests include Software Engineering, and he is passionate about teaching programming languages. He is pursuing his MTech degree at the same university.

Manoj Lall is a senior lecturer in the department of Computer science at Tshwane University of Technology, Pretoria, South Africa. He obtained a doctoral degree in Computer Science from the University of South Africa in 2013. His research interests include Formal methods, Machine learning, and Agent based modelling. He has published over thirty international refereed conference and journal papers.

Okuthe P. Kogeda obtained a doctorate degree in Computer Science from University of the Western Cape in Cape Town, South Africa in 2009. He is currently a Senior Lecturer, Chair of Departmental Research & Innovation Committee, and Head of Postgraduate Section in the Computer Science Department at Tshwane University of Technology, South Africa. He was a Senior Lecturer in the Computer Science Department at University of Fort Hare in Eastern Cape, South Africa from 2009 to 2011. He was a Lecturer in the Computer Science Department at University of the Western Cape in Cape Town, South Africa from 2004 to 2009. He was a Lecturer at University of Nairobi in Nairobi, Kenya from 1999 to 2000. He is a member of IITPSA, IAENG and IEEE. He is NRF rated researcher since 2015. He has successfully supervised over 30 postgraduate students. He has published over sixty internationally refereed conference and journal papers, author of five Chapters in books and author of three edited books.