Computers, Materials & Continua CMC, vol.65, no.1, pp.639-651, 2020

Identifying Game Processes Based on Private Working Sets

Jinfeng Li[1], Li Feng[1,*], Longqing Zhang[2], Hongning Dai[1], Lei Yang[1] and Liwei Tian[1]

Abstract: Fueled by the booming online games, there is an increasing demand for monitoring online games in various settings. One of the application scenarios is the monitor of computer games in school computer labs, for which an intelligent game recognition method is required. In this paper, a method to identify game processes in accordance with private working sets (i.e., the amount of memory occupied by a process but cannot be shared among other processes) is introduced. Results of the W test showed that the memory sizes occupied by the legitimate processes (e.g., the processes of common native windows applications) and game processes followed normal distribution. Using the *T*-test, a significant difference was identified between the legitimate processes and C/S-based computer games, in terms of the means and variances of their private working sets. Subsequently, we derived the density functions of the private working sets of the considered game processes and those of the legitimate processes. Given the private working set of a process and the derived probability density functions, the probability that the process is a legitimate process and the probability that the process is a game process can be determined. After comparing the two probabilities, we can easily determine whether the process is a game process or not. As revealed from the test results, the recognition accuracy of this method for C/S-based computer games was approximately 90%.

Keywords: Game process recognition, private working set, comparative analysis.

1 Introduction

Online games are becoming increasingly predominant in our leisure activities. Many people, especially those with poor self-control are suffering from addictions to online games. Accordingly, the possibility of monitoring online games has attracted increasing attention. Although computer labs in schools are primarily used for learning, some students with weak self-control often play games online in their labs. The development of intelligent game monitoring software will be effective in preventing students from playing games in the computer labs of their schools.

[1] Faculty of Information Technology, Macau University of Science and Technology, Taipa, 999078, Macau.

[2] Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa, 999078, Macau.

[*] Corresponding Author: Li Feng. Email: lfeng@must.edu.mo.

Received: 25 February 2020; Accepted: 29 May 2020.

The existing methods of game recognition are complex and exhibit relatively poor performance. In this paper, an effective method for identifying game processes by drawing a comparison of memory footprints is proposed. It was found that the private working sets of legitimate processes and those of game processes comply with normal distributions based on the results of the W test, whereas a significant difference was identified between the private working sets of legitimate processes and those of game processes. Subsequently, according to the sample data considered in this study, the probability density functions of the private working sets of game processes and those of legitimate processes were derived. Given the private working set of a process and the derived probability density function, the probability P_1 that the process is a legal process and the probability P_2 that the process refers to a game process can be determined. If $P_1 > P_2$, it is predicted as a legal process; otherwise, it is a game process. The proposed method is simple to apply, and yields high recognition accuracy on client/server-based (C/S-based) computer games

The rest of this paper is organized as follows. In Section 2, the related works are discussed. In Section 3, the proposed method of game process recognition is presented. In Section 4, the accuracy of the proposed method is evaluated. Finally, in Section 5 the conclusion of this study is described.

2 Related work

Numerous researchers have performed investigations on the identification of game and illegitimate processes. The main methods of game or illegitimate process identification can be listed as follows:

A. Method of blacklisting

First, a blacklist of illegitimate processes is created, following which the process information of the local machine is scanned. Subsequently, the records of the blacklist are compared with the process information of the local machine. If these are found to be identical, the corresponding process is determined to be illegitimate. This method is simple, yet time-consuming as all the blacklist information has to be updated [Gao and Guan (2007); Li and Li (2006); Yeming, Ori and Claudia (2017)].

B. Method using keystroke features of users

The characteristics of players pressing keys on the keyboard and clicking the mouse are different from those using non-game software [Zhang (2012); Li, Zhang, Yue et al. (2014); Shanmugapriya and Padmavathi (2011); Balagani, Phoha, Ray et al. (2011)]. For instance, a player frequently uses a specific set of keyboard keys; whereas, a user who does not play games uses these keys less frequently. However, in considerable cases, the characteristics of players pressing keys on the keyboard and clicking the mouse slightly differ from those of professional software users. For instance, professionals who use photoshop to perform operations on images often use certain keyboard shortcuts at high frequency and intensively.

C. Method of characteristic code detection

This method has been extensively applied in anti-virus software. Several experts and

scholars have researched the topic of characteristic code detection [Zhong, Li, Tang et al. (2010); Zou, Zhang, Zhang et al. (2014)]. Based on the collection of viruses or malware samples, the virus signature can be extracted from the malicious code. The hex code acts as an identifier of the virus or malware; and based on it, a signature database is built. A virus or malware scanner adopts a pattern matching algorithm (e.g., Brute Force (BF) and the Knuth-Morris-Pratt (KMP) algorithm) for signature matching [Wu, Fan, Wang et al. (2012)]. The disadvantage is that this method is costly as the signature database needs to be updated regularly.

D. Other methods

Some researchers delved into relevant issues using image processing, machine vision, and deep learning techniques [Luo, Qin, Xiang et al. (2019); Zhou, Qin, Xiang et al. (2020); Liu, Xiang, Qin et al. (2019); Wang, Qin, Xiang et al. (2019); Long, Pen and Zhu (2019)]. However, useable outcomes have rarely been achieved in this field. Many of these techniques are fueled by improvements in computing power, and the mentioned methods, which are closer to people's intuitive insights into game interfaces will also be worth studying, which should be our future research direction.

3 Data collection and processing

3.1 Data collection of legitimate processes

In this paper, legitimate applications refer to those applications that are allowed to be used in computer laboratories. The legitimate sample software considered in this study is listed in Tab. 1.

Table 1: List of legitimate sample applications

Soft Name and Version	Soft Name and Version
Visual Studio 2017	Baidu Disk
MS Powerpoint 2016	MS Excel 2016
SQL Server 2016	IBM SPSS Statistics 22
MS Word 2016	WeChat
Screen Video SpecialistV2015	TCP&UDP
MS Access 2016	Eclipse 4.5 (Mars)

The software of a target computer is launched, and the process information pertaining to the aforementioned software was scanned by the process scanner implemented by our team, and then saved to a database. The process information primarily comprises process names, average memory (private working set) sizes occupied by the processes, etc. For a better understanding, the term “private working set” is explained as follows. A working set refers to the physical memory occupied by a program (including the memory shared with other programs), and a private working set refers to the exclusive physical memory of that particular program. In this paper, the terms “private working set” and “memory” have been interchangeably used.

To be specific, the process information of sample applications was read and saved every 3 s using a monitoring tool (e.g., a process scanner module). The average of the top 40 values of the memory occupancy of each process was then taken. The average value is

used to represent memory size occupied by a process. It was determined that the memory footprints of the sampling processes range primarily between 40,000 KB and 250,000 KB, as shown in Fig. 1. Likewise, the information of other legitimate processes is also acquired. Most other legitimate processes were found to individually take up less than 40,000 KB of memory, as shown in Fig. 2.

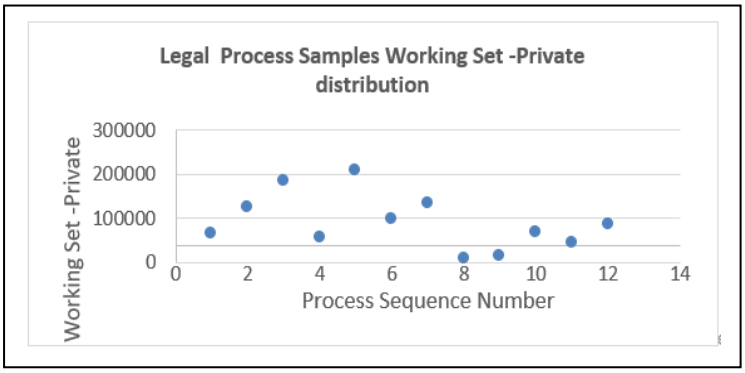

Figure 1: Distribution of private working sets of legitimate sample processes

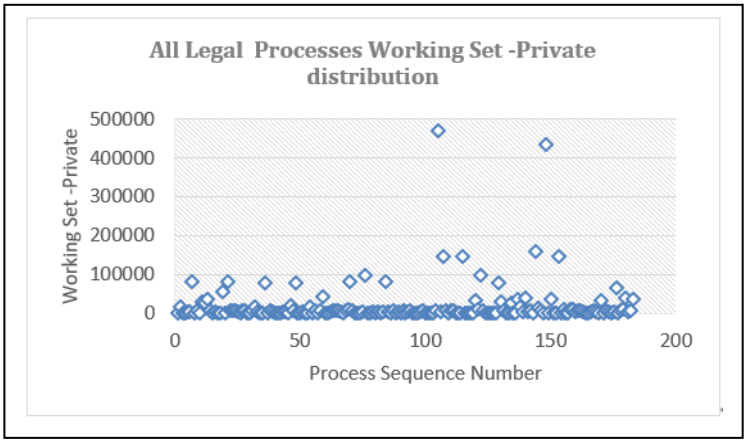

Figure 2: Distribution of private working sets of all legitimate processes in the computer

3.2 Game process data collection

Fifty popular online games were randomly selected as samples (as listed in Tab. 2). Based on the previous approach, a monitoring tool was used to read and store the process information of the samples. Moreover, the memory sizes occupied by the game processes were found to range primarily between 300000 KB and 1000000 KB, as depicted in Fig. 3.

Table 2: List of sample games

Name of game	Name of game	Name of game
Magic Chivalrous Biography (MX)	Blade & Soul	Torchlight
Legend of Mir II	League of Legends	Bejeweled
Expedition (YZ)	Warm Blood and Strong Protection	Crysis 2
HuaJiangShan	Need for Speed	StarCraft II
Dungeon & Fighter (DNF)	World of Warships	World of Warcraft
Call of Duty: Black Ops	Napoleon Total War	Football Manager 2011
Super Meat Boy	Machinarium	MonkeyIsland2: LeChucksRevenge-SpecialEdition
Dragon Age 2	Left 4 Dead 2	Need for Speed Hot Pursuit
Magicka	Red Dead Redemption 2	Amnesia
Alien Swarm	Fallout: New Vegas	Red Faction
Warhammer 40000 Dawn of War II	Fraternity	Battlefield: Bad Company 2
Batman: Arkham Asylum	Bulletstorm	MineCraft
Rift: Planes of Telara	Mass Effect 2	Minecraft
Teamfight Tactics	Total War: THREE KINGDOMS	Remnant: From the Ashes
SEKIRO: Shadows Die Twice	Control	Astral Chain
Risk of Rain 2	World War Z	Outlaws of the Old West
Yakuza Kiwami 2	One Piece: World Seeker	

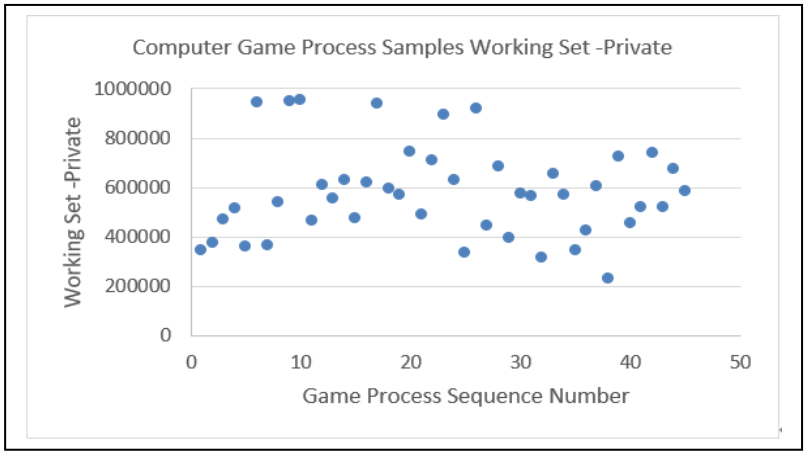

Figure 3: Private working set distributions of game processes

3.3 W test

The W test, which is a correlation-based algorithm, is also known as the Shapiro-Wilk test. The results yield a correlation coefficient; the closer it is to 1, the better will the data and normal distribution fit. It is generally considered that when the sample size n reaches $3 \le n \le 50$, the W test can be adopted to verify whether these samples comply with a normal distribution [Zhang and Dong (2015); He and Wang (2014)]. The formula for the W test can be expressed as follows:

$$W = \frac{\left(\sum_{i=1}^{n} \alpha_i x_i\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{1}$$

If $W < W_\alpha$, the normal hypothesis is discarded according to the significance level α; if $W > W_\alpha$, the normal hypothesis is accepted.

The values of the memory (i.e., the private working set) occupied by the valid processes are tested by SPSS. The results are presented in Tab. 3.

Table 3: Normality test of legitimate processes

	Kolmogorov-Smirnova			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Private working set	0.191	184	0.200*	0.937	184	0.485

As can be seen, the value of Sig is 0.485, which is greater than 0.05. Thus, it can be shown that the values of the memory sizes occupied by legitimate processes indicate a normal distribution.

This method is also used to test the values of memory occupancy of game processes and the results are listed in Tab. 4.

Table 4: Normality test of game processes

	Kolmogorov-Smirnova			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Private working set	0.265	50	0.146	0.767	50	0.106

As can be seen, the value of Sig is 0.106, which is greater than 0.05. Thus, the values of the memory sizes occupied by game processes indicate a normal distribution.

Fig. 4 shows the differences in the probability distributions of the private working sets occupied by legitimate processes and game processes in an intuitive manner.

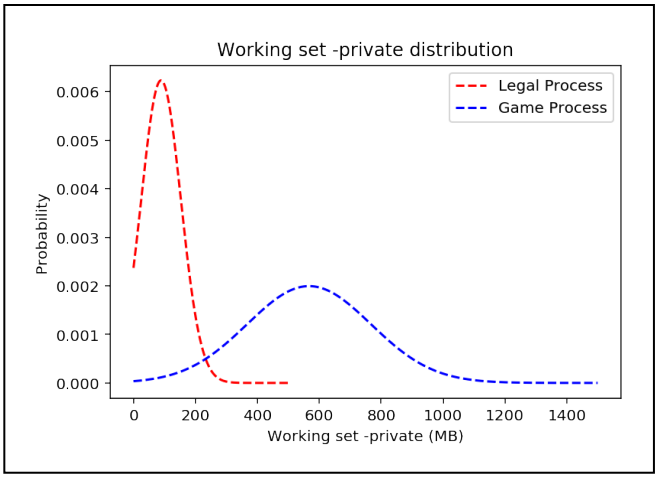

Figure 4: Probability distributions of private working sets occupied by legitimate processes and game processes

3.4 Comparative analysis

The *T*-test is performed to infer whether a significant difference can be identified between the mean values of the two collections (the values of the process memory occupancy of legitimate applications and those of game process memory occupancy). *T*-test, also called Student's *T*-test, is primarily performed for normal distributions with a small sample size (e.g., n<30) and an unknown population standard deviation σ. The *T*-test is split into a single-population test and a double-population test. The double-population *T*-test aims to test whether there is a significant difference between the averages of two samples and the corresponding populations. The double-population *T*-test can fall into two cases, i.e., independent sample *T*-test and paired sample *T*-test. The formula for calculating independent sample *T*-test statistics is as follows:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}} \tag{2}$$

$\overline{X_1}$: Average of the memory occupancy sizes of game samples;

$\overline{X_2}$: Average of the memory occupancy sizes of legitimate processes;

S_1^2 : Variance of the memory occupancy sizes of sample games;

S_2^2 : Variance of memory occupancy sizes of legitimate sample processes, which contain legitimate sample software as presented in Tab. 1 and other legitimate processes running on the target computer;

n_1 : Sample size of c/s computer games;

n_2 : Sample size of the legitimate processes.

Two groups of independent samples are tested by SPPS, and the results are listed in Tab. 5.

Table 5: Independent samples test

		Levene's Test for Equality of Variances		*t*-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Private working sets	Equal variances assumed	34.106	0.000	-19.481	98	0.000	-493940.13089	25355.44765	-544257.19155	-443623.07024
	Equal variances not assumed			-19.481	56.658	0.000	-493940.13089	25355.44765	-544720.19113	-443160.07065

The value of F of the Levene's Test is 34.106, and the value of Sig is 0, which is less than the significance level (e.g., 0.05). Therefore, a significant difference is identified in the variance. As can be observed from the results of the T-test, the value of Sig (2-tailed) is 0, which is lower than the significance level (e.g., 0.05), and a significant difference is identified in the mean of the two populations.

3.5 Probability density functions

The values of the memory sizes used by the two types of processes comply with the normal distributions, and therefore, the probability density function can be expressed as follows:

$$p(x|\theta) \sim N(\mu, \sigma^2) \tag{3}$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ can be computed using the maximum likelihood estimation. Moreover, the equations are written as follows:

$$\hat{\mu} = \frac{1}{N}\sum_{k=1}^{N} x_k \tag{4}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{k=1}^{N}(x_k - \hat{\mu})^2 \tag{5}$$

The probability density of the memory sizes of legitimate processes can be defined as follows:

$$f_1(x) = \frac{1}{\sigma_1 * \sqrt{2\pi}} e^{\left[-\frac{(x-\widehat{\mu_1})^2}{2*\widehat{\sigma_1}^2}\right]} \tag{6}$$

The probability density of the memory sizes of game processes can be defined as follows:

$$f_2(x) = \frac{1}{\sigma_2 * \sqrt{2\pi}} e^{\left[-\frac{(x-\widehat{\mu_2})^2}{2*\widehat{\sigma_2}^2}\right]} \tag{7}$$

3.6 Game recognition

The basic idea of a game recognition algorithm is to 1) get the memory size of a process, 2) calculate the probabilities of whether it is a game or a legitimate process using the memory size, 3) predict whether the corresponding process is a game process. The method to compute the probabilities is described below.

a) If the size of memory taken by a process $x_0 \leq \hat{\mu}_1$: the probability that the process is a legitimate process is expressed as

$$p_1(x < x_0) = \int_{-\infty}^{x_0} f_1(x)dx = \Phi(\frac{x - \widehat{\mu_1}}{\widehat{\sigma_1}}) \tag{8}$$

the probability that the process is a game process is

$$p_2(x < x_0) = \int_{-\infty}^{x_0} f_2(x)dx = \Phi(\frac{x - \widehat{\mu_2}}{\widehat{\sigma_2}}) \tag{9}$$

b) If the size of memory taken by a process $\hat{\mu}_1 < x_0 < \hat{\mu}_2$: the probability that the process is a legitimate process is written as

$$p_1(x > x_0) = \int_{x_0}^{+\infty} f_1(x)dx = 1 - \Phi(\frac{x - \widehat{\mu_1}}{\widehat{\sigma_1}}) \tag{10}$$

the probability that the process is a game process is

$$p_2(x < x_0) = \int_{-\infty}^{x_0} f_2(x)dx = \Phi(\frac{x - \widehat{\mu_2}}{\widehat{\sigma_2}}) \tag{11}$$

c) If the size of memory occupied by a process $x_0 \geq \hat{\mu}_2$: the probability that the process is a legitimate process is expressed as

$$p_1(x > x_0) = \int_{x_0}^{+\infty} f_1(x)dx = 1 - \Phi(\frac{x - \widehat{\mu_1}}{\widehat{\sigma_1}}) \tag{12}$$

the probability that the process is a game process is

$$p_2(x > x_0) = \int_{x_0}^{+\infty} f_2(x)dx = 1 - \Phi(\frac{x - \widehat{\mu_2}}{\widehat{\sigma_2}}) \tag{13}$$

Given the mentioned analysis, the process of computer game process identification is elucidated as follows:

First, the process memory information of the sample applications, sample games, and all the other processes in the target computer is read.

Subsequently, Eq. (4) can be adopted to get $\hat{\mu}_1$ and $\hat{\mu}_2$, and then Eq. (5) can be employed to get $\hat{\sigma}_1$ and $\hat{\sigma}_2$.

Lastly, the probabilities of game and legitimate process are calculated through the memory size occupied with the described method. Furthermore, whether the process is a

legitimate process, or a game process is demonstrated according to the results of probability calculation.

For instance, the memory size occupied by a process is assumed as x_0.

a) If $x_0 \leq \hat{\mu}_1$: we calculate x_0 into Eqs. (8) and (9), respectively, and then compare the calculation results; if $p_1(x < x_0) > p_2(x < x_0)$, this process is judged as a legitimate process; if $p_1(x < x_0) < p_2(x < x_0)$, the process is considered a game process.

b) If $\hat{\mu}_1 < x_0 < \hat{\mu}_2$: we substitute x_0 into Eqs. (10) and (11), respectively, and then compare the calculation results; if $p_1(x > x_0) > p_2(x < x_0)$, this process is verified as a legitimate process; if $p_1(x > x_0) < p_2(x < x_0)$, the process is considered a game process.

c) If $x_0 \geq \hat{\mu}_2$: we substitute x_0 into Eqs. (12) and (13), respectively, and then compare the calculation results; if $p_1(x > x_0) > p_2(x > x_0)$, this process is verified as a legitimate process; if $p_1(x > x_0) < p_2(x > x_0)$, the process is considered a game process.

From Eqs. (8)-(13), the probability is calculated by converting the general normal distribution into a standard normal distribution. Subsequently, the result is found in the standard normal distribution table.

4 Performance evaluation

20 legitimate applications and 20 computer games were randomly selected to test the efficacy of the proposed method. The results of identification of legitimate application processes are listed in Tab. 6. and those of game processes are listed in Tab. 7.

Table 6: Test results of legitimate application process recognition

Soft Name and Version	Is killed	Soft Name and Version	Is killed
Word 2010	False	Autocad 2004	False
PowerPoint 2013	False	Sql server2008	False
Wechat_devtools 1.02.1804120_x64	False	Mysql 5.7.17	False
PyCharm 2018 3.2	False	Visio 2016	False
Outlook 2016	False	Jupyter Notebook 5.5.0	False
Photoshop CS6	False	Adobe Flash Professional cc 2015	False
Dreamweaver 2017	False	Xmind 3.7.2.201705211940	False
Visual studio 2019	False	Webstorm 2017.1	True
Eclipse Neon 3	True	3dmax 2014	True
Sublime Text 3, Build 3126	False	Android Studio 3.2	False

Table 7: Test results of game process recognition

Name of game	Is killed	Name of game	Is killed
AO International Tennis	True	Sword Hero II	True
Romance of the Three Kingdoms XIII	True	GuJian 3	True
Fantasy Westward Journey	False	Total War: THREE KINGDOMS	True
The World of Legend	True	Counter-Strike Online	True
Assassin's Creed Odyssey	True	XTLBB Online	True
XY 2	True	XuanYuan Sword:The Gate of Firmament	True
JX Online V3	True	Call of Duty Online	True
BIOHAZARD RE:2	True	The Magic Blade	True
DOTA2	True	HearthStone: Heroes of Warcraft	True
Overwatch	True	Player Unknown's Battle Grounds	True

Definition 1: false positives rate=(number of legitimate processes misreported as game processes)/(total number of legitimate processes).

According to Tab. 6, as suggested from the test results, the false positives rate is 15%.

Definition 2: game process recognition rate=(number of game processes identified)/(total number of game processes).

According to Tab. 7, as revealed from the test results, the game process recognition rate is 95%.

Definition 3: accuracy of game process recognition=(1-false positives rate+game process recognition rate)/2.

The accuracy of game process recognition was calculated to be 90%.

This method outperforms and is simpler to use than the methods mentioned in Section 2.

5 Conclusion and future work

We proposed a game process recognition method based on the private working sets of computer software. The proposed method is simple and yields high recognition accuracy (approximately 90%) for computer games based on C/S architecture. However, since the process memory occupancy is tightly related to computer hardware and operating systems, the probability density function and its parameter values proposed in this paper cannot be directly applied to other computer environments. In other words, this method exhibits poor portability. Therefore, in future work, machine learning technologies will be adopted to improve the accuracy and portability of the proposed method.

Acknowledgment: The authors gratefully acknowledge Prof. Qinglin Zhao at Macau University of Science and Technology for his helpful advice. The authors also thank the very thorough reviewers.

Funding Statement: This work is funded in part by the National Nature Science Foundation of China (File Nos. 61872451 and 61872452) and in part by the Science and Technology Development Fund, Macau SAR (File Nos. 0098/2018/A3 and 0076/2019/A2). Li Feng is the corresponding author.

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

References

Balagani, K. S.; Phoha, V. V.; Ray, A.; Phoha, S. (2011): On the discriminability of keystroke feature vectors used in fixed text keystroke authentication. *Pattern Recognition Letters*, vol. 32, no. 7, pp. 1070-1080.

Gao, D. Q.; Guan, Q. (2007): Design of an illegal processes monitor program. *Modern Computer*, vol. 24, no. 11, pp. 90-93.

He, Q.; Wang, Z. K. (2014): Application of normality test in teaching and research. *Higher Science Education*, vol. 22, no. 4, pp. 18-21.

Li, C.; Zhang, G. X.; Yue, B. L.; He, D. L. (2014): Keystroke dynamics based user authentication using conditional random fields. *Application Research of Computers*, vol. 31, no. 7, pp. 2112-2115.

Li, Y. H.; Li, X. Z. (2006): Design and implementation of a game monitoring program. *Information Technology*, vol. 30, no. 11, pp. 24-26.

Liu, Q.; Xiang, X. Y.; Qin, J. H.; Tan, Y.; Tan, J. S. et al. (2019): Coverless steganography based on image retrieval of DenseNet features and DWT sequence mapping. *Knowledge-Based Systems*, https://doi.org/10.1016/j.knosys.2019.105375.

Long, M.; Pen, F.; Zhu, Y. (2019): Identifying natural images and computer generated graphics based on binary similarity measures of PRNU. *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 489-506.

Luo, Y. J.; Qin, J. H.; Xiang, X. Y.; Tan, Y.; Liu, Q. et al. (2019): Coverless real-time image information hiding based on image block matching and Dense Convolutional Network. *Journal of Real-Time Image Processing*.

Shanmugapriya, D.; Padmavathi, G. (2011): An efficient feature selection technique for user authentication using keystroke dynamics. *International Journal of Computer Science and Network Security*, vol. 11, no. 10, pp. 191-195.

Shi, Y.; Stitelman, O.; Perlich, C. (2017): Blacklisting the blacklist in online advertising: improving delivery by bidding for what you can win. the ADKDD'17. *ACM*.

Wang, J.; Qin, J. H.; Xiang, X. Y.; Tan, Y.; Pan, N. (2019): CAPTCHA recognition based on deep convolutional neural network. *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 5851-5861.

Wu, W. M.; Fan, W. F.; Wang, Z. Y.; Li, X. F.; Huang, J. W. (2012): Pe file auto free-antivirus strategy based on characteristic code. *Computer Engineering*, vol. 38, no. 12, pp. 118-121.

Zhang, B. Z. (2012): A game prohibiting based on real time keystroke detection

algorithm. *Information Security*, vol. 3, no. 9, pp. 32-34.

Zhang, L. K.; Dong, J. Q. (2015): Sample sequential normality test in ballistic consistency assessment. *Journal of Artillery Launch and Control*, vol. 37, no. 2, pp. 68-72.

Zhong, M. Q.; Li, H. Z.; Tang, Z. G.; Zhang, J. (2010): Auto-detection system of suspicious file based on virtual machine technology. *Journal of Computer Applications*, vol. 27, no. 12, pp. 3357-3362.

Zhou, Z.; Qin, J. H.; Xiang, X. Y.; Tan, Y.; Liu, Q. et al. (2020): News text topic clustering optimized method based on TF-IDF Algorithm on spark. *Computers, Materials & Continua*, vol. 62, no. 1, pp. 217-231.

Zou, W. F.; Zhang, Y. Y.; Zhang, S. X.; Yang, C. Y. (2014): Research on anti-trojan malware mechanism based on characteristic behavior. *Telecommunications Science*, vol. 59, no. 11, pp. 105-109.