Cybersecurity

RESEARCH Open Access

TKCA: a timely keystroke-based continuous user authentication with short keystroke sequence in uncontrolled settings

Lulu Yang[1,2], Chen Li[1,2], Ruibang You[1,2], Bibo Tu[1,2*] and Linghui Li[3]

Abstract

Keystroke-based behavioral biometrics have been proven effective for continuous user authentication. Current state-of-the-art algorithms have achieved outstanding results in long text or short text collected by doing some tasks. It remains a considerable challenge to authenticate users continuously and accurately with short keystroke inputs collected in uncontrolled settings. In this work, we propose a Timely Keystroke-based method for Continuous user Authentication, named TKCA. It integrates the key name and two kinds of timing features through an embedding mechanism. And it captures the relationship between context keystrokes by the Bidirectional Long Short-Term Memory (Bi-LSTM) network. We conduct a series of experiments to validate it on a public dataset - the Clarkson II dataset collected in a completely uncontrolled and natural setting. Experiment results show that the proposed TKCA achieves state-of-the-art performance with 8.28% of EER when using only 30 keystrokes and 2.78% of EER when using 190 keystrokes.

Keywords: Keystroke dynamics, Continuous user authentication, Embedding, LSTM, Bi-LSTM

Introduction

The traditional computer desktop or cloud desktop authentication method uses a point of entry for users to log in with a username and password or PIN. Unauthorized access could occur when a legitimate user forgets to log out and steps away from the terminal for lunch or an emergency or when an attacker has stolen his password or PIN. The attacker may gain access to a fully operational system with a privileged account and access it like the user. Subsequent attack mounted on the desktop is hard to discover, which poses a significant security risk. Continuous user authentication is a way to tackle this issue. It actively and continuously authenticates users without their awareness. Behavioral biometrics is popular in the continuous user authentication field by using users' characteristics to authenticate their identity information. Continuous user authentication based on behavioral biometrics includes free-text keystroke dynamics (Monrose and Rubin 1997; Dowland and Furnell 2004; Gunetti and Picardi 2005; Janakiraman and Sim 2007; Sim and Janakiraman 2007; Montalvão Filho and Freire 2006; Davoudi and Kabir 2009; 2010; Harun et al. 2010; Stewart et al. 2011; Al Solami et al. 2011; Messerman et al. 2011; Rahman et al. 2011; Ferreira and Santos 2012; Bours 2012; Monaco et al. 2013; Deutschmann et al. 2013; Ahmed and Traore 2013; Kang and Cho 2015; Çeker and Upadhyaya 2016; Mondal and Bours 2017; Huang et al. 2017; Ayotte et al. 2019; Alshehri et al. 2017; 2018; Xiaofeng et al. 2019), mouse dynamics (Pusara and Brodley 2004; Ahmed and Traore 2007; Nakkabi et al. 2010; Zheng et al. 2011; Feher et al. 2012; Lin et al. 2012; Mondal and Bours 2013), touch screen inputs (Frank et al. 2012; Cai et al. 2013; Feng et al. 2014; Buschek et al. 2015), eye movements (Kinnunen et al. 2010; Eberz et al. 2015; 2016), gait (Ailisto et al. 2005; Rong et al. 2007; Derawi et al. 2010), etc. Although continuous user

*Correspondence: tubibo@iie.ac.cn
[1] Institute of Information Engineering, Chinese Academy of Sciences, 100093 Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, 100049 Beijing, China
Full list of author information is available at the end of the article

authentication can not replace traditional authentication schemes, it makes up for their shortcomings.

Free text keystroke dynamics is proper for continuous user authentication because it is non-invasive, relatively cheap to realize, no additional hardware requirement, and available after the login stage (Dowland and Furnell 2004). Considerable progress has been made in free text Keystroke-based Continuous user Authentication (KCA) (Monrose and Rubin 1997; Dowland and Furnell 2004; Gunetti and Picardi 2005; Janakiraman and Sim 2007; Sim and Janakiraman 2007; Montalvão Filho and Freire 2006; Davoudi and Kabir 2009; 2010; Harun et al. 2010; Stewart et al. 2011; Al Solami et al. 2011; Messerman et al. 2011; Rahman et al. 2011; Ferreira and Santos 2012; Bours 2012; Monaco et al. 2013; Deutschmann et al. 2013; Ahmed and Traore 2013; Kang and Cho 2015; Çeker and Upadhyaya 2016; Mondal and Bours 2017; Huang et al. 2017; Ayotte et al. 2019; Alshehri et al. 2017; 2018; Xiaofeng et al. 2019). The process of a KCA algorithm is usually to collect a user's keystrokes and time stamps, extract features, and use statistical or machine learning methods to determine whether the current user is legitimate. Notable performances have been achieved with long (Gunetti and Picardi 2005; Çeker and Upadhyaya 2016) or short text (Alshehri et al. 2017; 2018; Xiaofeng et al. 2019) collected by doing tasks (i.e., answering questions or transcriptions) and long data (Huang et al. 2017; Ayotte et al. 2019) collected in uncontrolled settings.

However, it remains a challenge to use short keystroke inputs collected in uncontrolled settings for accurate KCA. Since proved in Huang et al. (2017), that KCA performance degrades significantly when applied to data collected in uncontrolled environments. Moreover, users with similar traits produce similar keystroke data (Lau and Maxion 2014). And the same user's typing behavior is varied by mood, keyboard layout, application, time resolution, and other factors. They are making it challenging to distinguish users or identify the same user. Otherwise, in real-world scenarios, users usually type only a few characters to search or chat online and spend the rest of the time browsing the web, documents, pictures, videos, etc. We find that the number of keystrokes less than 50 in once typing accounts for 51.95% of all keystrokes in the Clarkson II dataset (Murphy et al. 2017). It is necessary to overcome these difficulties of using short keystroke data for accurate KCA in real uncontrolled scenarios.

In this work, we propose a Timely Keystroke-based Continuous user Authentication, named TKCA, to detect attackers using short keystroke sequences in uncontrolled scenarios quickly and accurately. We use the Bi-directional Long Short-Term Memory (Bi-LSTM) network to capture the relationship between context keystrokes. As stated in Sim and Janakiraman (2007), a keystroke sequence's typing pattern may change when it is part of a longer word. For example, the digraph "in" may have different timing information in typing the word "mini" and the word "thing" because a user's typing behavior depends not only on the exact key but also on the context keys. To make the most use of the keystroke data's information, we use all available keystrokes instead of selecting commonly used digraphs, trigraphs while filtering out others or just using timing features. We propose integrating the key name and two kinds of timing features by embedding mechanism. As far as we know, we are the first engaging embedding mechanism in KCA to convert key names into digital vectors and amply timing features.

To evaluate the performance of TKCA and compare it with previous works, we conduct a series of experiments on the Clarkson II dataset. The TKCA algorithm achieves an EER (Equal Error Rate) of 8.28% when only using 30 keystrokes. When the number of keystrokes increases to 90, the EER reaches 4.30%. With more keystrokes, the performance continues to improve steadily. With 190 keystrokes, the EER drops to 2.78%. The evaluation results show that the TKCA achieves state-of-the-art performance, and it is desirable for accurate KCA with short keystroke sequences in uncontrolled settings.

The primary contributions in this work are as follows:

(1) We design a keystroke model based on embedding mechanism and Bi-LSTM. We propose integrating the key name and two kinds of timing features (the hold time and the digraph flight time) by the embedding mechanism that converts key names into digital vectors and amply timing features. Moreover, we use Bi-LSTM to learn the dependence between context keystrokes. They are proved to be effective in improving accuracy in the experimental section.

(2) We propose TKCA based on the keystroke model and majority vote mechanism to quickly detect attackers using short unconstrained keystroke sequences. For each legitimate user, we train a unique binary classifier based on the keystroke model. The TKCA uses an authorized user's classifier to classify keystroke sequences. Multiple classification results are fused by a majority vote to determine whether the current user is legitimate.

(3) We evaluate the proposed TKCA and compare it with previous KCA algorithms on the Clarkson II dataset, collected in a completely uncontrolled and natural setting. Experimental results show the EER of TKCA 8.28% when only using 30 keystrokes, which dramatically improves the KCA field's performance.

The rest of this paper is organized as follows: "Related work" section provides the related work. The keystroke model and the TKCA algorithm is presented in "Methodology" section. In "Experiments and evaluation" section, we evaluate TKCA and compare it with other KCA algorithms on the Clarkson II dataset comprehensively. We discuss the strengths and limitations of the proposed TKCA method and future work in "Discussion and future work" section. Finally, we make a conclusion in "Conclusion" section.

Related work

Keystroke dynamics is an efficient and inexpensive behavioral biometrics that can be used to authenticate users in the background while the user is actively working. Various previous works on user authentication using keystroke dynamics focus on Keystroke-based Static user Authentication (KSA), which extracts typing patterns from predefined texts. Applications such as username, password, and PIN authentication (Joyce and Gupta 1990; Monrose et al. 2002; Killourhy and Maxion 2009; Syed et al. 2016) apply KSA to assist in authenticating users. However, KSA is not suitable for scenarios where continuous user authentication is required. Hence, the idea of using keystroke dynamics for free text, Keystroke-based Continuous user Authentication (KCA), has been proposed. In the literature, researchers use different types of keystroke data for KCA studies according to the collection device. Many KCA studies use keystroke data collected on a traditional keyboard for continuous user authentication when users work at a computer terminal. Besides, some KCA studies focused on mobile systems gather keystroke data on soft keyboards, such as touch screens (Feng et al. 2013; Wu et al. 2015). Since this work focuses on using keystroke data to detect attackers in computer/cloud terminal scenarios, we will only introduce KCA studies using traditional hardware keyboards. KCA studies using traditional keyboards have come a long way in the last two decades (Dowland and Furnell 2004; Gunetti and Picardi 2005; Janakiraman and Sim 2007; Sim and Janakiraman 2007; Monrose and Rubin 1997; Montalvão Filho and Freire 2006; Davoudi and Kabir 2009; 2010; Harun et al. 2010; Stewart et al. 2011; Al Solami et al. 2011; Messerman et al. 2011; Rahman et al. 2011; Ferreira and Santos 2012; Bours 2012; Monaco et al. 2013; Deutschmann et al. 2013; Ahmed and Traore 2013; Kang and Cho 2015; Çeker and Upadhyaya 2016; Mondal and Bours 2017; Huang et al. 2017; Ayotte et al. 2019; Alshehri et al. 2017; 2018; Xiaofeng et al. 2019) as shown in Table 1. In the following, we will introduce the existing KCA studies in terms of features, methods, evaluation metrics, and datasets.

Features

Almost all existing KCA studies use keystroke timing features for classification. As shown in Fig. 1, timing features include hold time, latency time, n-graphs flight time, etc. The naming of timing features varies from study to study. For example, hold time is also called dwell time, duration, or held time. They all represent the duration between pressing and releasing the same key. In this work, we refer to this time as hold time. Hold time has been used in Janakiraman and Sim (2007); Sim and Janakiraman (2007); Monrose and Rubin (1997); Stewart et al. (2011); Ferreira and Santos (2012); Bours (2012); Monaco et al. (2013); Deutschmann et al. (2013); Ahmed and Traore (2013); Çeker and Upadhyaya (2016); Mondal and Bours (2017); Alshehri et al. (2017); Alshehri et al. (2018); Xiaofeng et al. (2019). Latency time is the time from the previous key released to the current key pressed, which has been used in Monrose and Rubin (1997); Stewart et al. (2011); Rahman et al. (2011); Ferreira and Santos (2012); Bours (2012); Monaco et al. (2013); Ahmed and Traore (2013); Mondal and Bours (2017); Xiaofeng et al. (2019). N-graph flight time is the time from the first key pressed to the last key pressed (Moskovitch et al. 2009), and the frequently used values of n are 2 (digraph), 3 (trigraph), 4. Many KCA studies use n-graph flight time as a feature, such as (Dowland and Furnell 2004; Gunetti and Picardi 2005; Janakiraman and Sim 2007; Sim and Janakiraman 2007; Montalvão Filho and Freire 2006; Davoudi and Kabir 2009; 2010; Al Solami et al. 2011; Messerman et al. 2011; Ferreira and Santos 2012; Ahmed and Traore 2013; Kang and Cho 2015; Çeker and Upadhyaya 2016; Mondal and Bours 2017; Huang et al. 2017; Ayotte et al. 2019; Alshehri et al. 2017; 2018; Xiaofeng et al. 2019). Besides, there are some other timing features, for example, n-graph total time used in Dowland and Furnell (2004); Mondal and Bours (2017), up-up time used in Mondal and Bours (2017), and percent usage of certain keys used in Stewart et al. (2011). In addition to timing features, the key itself can also be used as a feature. For example, (Xiaofeng et al. 2019) uses the keycode as a feature.

Methods

Various techniques have been used in KCA algorithms. Gunetti and Picardi (2005); Messerman et al. (2011); Rahman et al. (2011); Kang and Cho (2015) use 'R' Distance and 'A' Distance. 'R' Distance is determined by the normalized disorder between the two ordered vectors of average n-graph latencies. 'A' Distance is determined by the difference in average n-graph latencies. Euclidean distance, Manhattan distance, or Bhattacharyya Distance have been used in Dowland and Furnell (2004); Janakiraman and Sim (2007); Sim and Janakiraman (2007); Monrose and Rubin (1997); Davoudi and Kabir (2009); Davoudi and Kabir (2010); Harun et al. (2010); Ferreira and Santos (2012); Bours (2012); Kang and Cho (2015) to calculate the mean and standard deviation of some timing features. Moreover, to identify users, a number of studies use machine learning methods including k-Nearest Neighbour (k-NN) or Nearest Neighbour (Monrose and Rubin 1997; Stewart et al. 2011; Monaco et al. 2013; Kang and Cho 2015), Markov Chain (Montalvão Filho and Freire 2006), Support Vector Machines (SVM) (Çeker and Upadhyaya 2016; Mondal and Bours 2017), Kernel Density

Table 1 Available KCA algorithms based on free-text keystroke dynamics

Study	Feature	Method	Dataset	Performance	Length
(Monrose and Rubin 1997)	hold time, latency time	k-NN, Euclidean distance	collect	ACC: 23%	31
(Dowland and Furnell 2004)	n-graph flight time (n=2, 3),	mean, standard deviation	collect	FAR: 4.9%,	
	keyword-based keystroke latency			FRR: 0%,	
				ANIA: 6,390,	
				ANGA: 68,755	
(Gunetti and Picardi 2005)	n-graph flight time (n=2, 3, 4)	'R' and 'A' Distance	collect	FAR: 0.005%	700-900
				FRR: 4.833%	
(Montalvão Filho and Freire 2006)	digraph flight time	Markov Chain	collect	EER: 12.7%	about 550
(Janakiraman and Sim 2007),	held time, inter-key time	Bhattacharyya Distance,	collect	ACC: 74-100%	1,500-100,000
(Sim and Janakiraman 2007)		Naive Bayes			
(Davoudi and Kabir 2009; 2010)	digraph flight time	HDE, 'R' and 'A' Distance	(Gunetti and Picardi 2005)	FAR: 0.07%, FRR: 15.2%	700-900
(Harun et al. 2010)		ANN, some Distance	(Montalvão Filho and Freire 2006)	EER: 22.9%	about 550
(Stewart et al. 2011)	hold time, latency time,	k-NN	collect	EER: 0.55%	about 6,000
	percent usage of keys				
(Al Solami et al. 2011)	most typed digraph	Clustering	(Gunetti and Picardi 2005)	ACC: 100%	700-900
(Messerman et al. 2011)	n-graph flight time (n=2, 3, 4)	'R' and 'A' Distance	collect	eFAR: 2.61%, iFAR: 2.02%, FRR: 1.84%	150-450
(Rahman et al. 2011)	latency time	'R' and 'A' Distance	collect	EER: 10-15%	854-1,836
(Ferreira and Santos 2012)	hold time, latency time,	mean, standard deviation	collect	EER of 1.4%	250
	n-graph flight time (n=2, 3, 4)				
(Bours 2012)	hold time, latency time	mean, standard deviation	collect	ANIA: 182	
(Monaco et al. 2013)	hold time, latency time	mean, standard deviation	collect	EER: 3.7%	755
(Deutschmann et al. 2013)		Bayesian network	collect	ANIA: 760-950	
(Ahmed and Traore 2013)	hold time, latency time,	ANN	collect	EER: 2.13%	500
	n-graph flight time (n=2, 3)				
(Kang and Cho 2015)	digraph flight time	12 different techniques	collect	EER: 5.64%	1,000
(Çeker and Upadhyaya 2016)	hold time, digraph flight time	One-class SVM	(Vural et al. 2014)	EER: 0	1,319-3,454
(Mondal and Bours 2017)	hold time, total time, latency time,	ANN, CPANN, SVM	collect	ANIA: 167 or larger	

Table 1 Available KCA algorithms based on free-text keystroke dynamics (*Continued*)

Study	Feature	Method	Dataset	Performance	Length
	flight time, up-up time for digraph				
(Alshehri et al. 2017; 2018)	hold time, digraph flight time	DTW	collect, (Vural et al. 2014)	ACC: 98.39%, 97.32%	100
(Huang et al. 2017)	digraph flight time	KDE	(Gunetti and Picardi 2005), (Vural et al. 2014), (Sun et al. 2016), (Murphy et al. 2017)	EER: 3.48%, 3.36%, 1.95%, 7.59%	1,000
(Ayotte et al. 2019)	digraph flight time	KDE, Energy Distance, Kolmogorov-Smirnov	(Murphy et al. 2017)	EER: 35.1%, 15.3%, 6.3%, 3.6%	100, 200, 500, 1,000
(Xiaofeng et al. 2019)	keycode, hold time, latency time, digraph flight time	CNN, RNN	(Sun et al. 2016)	EER: 3.04	30

Estimation (KDE) (Davoudi and Kabir 2009; Huang et al. 2017; Ayotte et al. 2019), Dynamic Time Warping (DTW) (Alshehri et al. 2017; 2018), Artificial Neural Network (ANN) (Harun et al. 2010; Ahmed and Traore 2013; Mondal and Bours 2017), Histogram-based Density Estimation (Davoudi and Kabir 2009; 2010), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) (Xiaofeng et al. 2019), etc.

Metrics

Besides, the evaluation metrics of KCA studies are not all the same. FAR, FRR, EER, and Accuracy are the four most commonly used evaluation metrics. FAR (False Accept Rate) is the ratio of impostor attacks that are falsely accepted as genuine users and has been used in Dowland and Furnell (2004); Gunetti and Picardi (2005); Montalvão Filho and Freire (2006); Davoudi and Kabir (2009); Davoudi and Kabir (2010); Harun et al. (2010); Stewart et al. (2011); Messerman et al. (2011); Rahman et al. (2011); Ferreira and Santos (2012); Monaco et al. (2013); Ahmed and Traore (2013); Kang and Cho (2015); Çeker and Upadhyaya (2016); Huang et al. (2017); Ayotte et al. (2019); Xiaofeng et al. (2019). Messerman et al. (2011) uses eFAR and iFAR to differentiate between unknown external and known internal attackers. FRR (False Reject Rate) is the ratio of genuine tests that are falsely rejected as impostors and has been used in Dowland and Furnell (2004); Gunetti and Picardi (2005); Montalvão Filho and Freire (2006); Davoudi and Kabir (2009); Davoudi and Kabir (2010); Harun et al. (2010); Stewart et al. (2011); Messerman et al. (2011); Rahman et al. (2011); Ferreira and Santos (2012); Monaco et al. (2013); Ahmed and Traore (2013); Kang and Cho (2015); Çeker and Upadhyaya (2016); Huang et al. (2017); Ayotte et al. (2019); Xiaofeng et al. (2019). EER (Equal Error Rate) is the point on a DET (Detection Error Tradeoff) curve where FAR and FRR are equal and has been used in Montalvão Filho and Freire (2006); Harun et al. (2010); Stewart et al. (2011); Rahman et al. (2011); Ferreira and Santos (2012); Monaco et al. (2013); Ahmed and Traore (2013); Kang and Cho (2015); Çeker and Upadhyaya (2016); Huang et al. (2017); Ayotte et al. (2019); Xiaofeng et al. (2019). EER balance FAR and FRR to avoid one of them being too large. ACC (Accuracy) is the ratio of truly rejected impostors and accepted

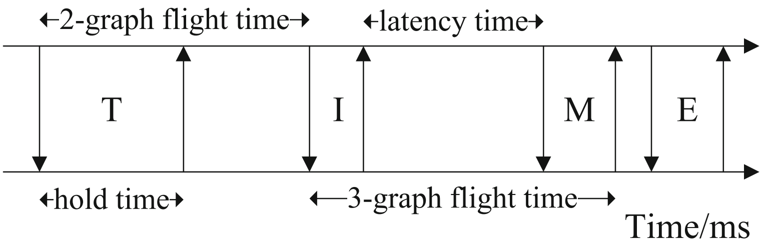

Fig. 1 A keystroke sequence. Each upward and downward pointing arrow indicates a keystroke event

genuine users and has been used in Janakiraman and Sim (2007); Sim and Janakiraman (2007); Monrose and Rubin (1997); Al Solami et al. (2011); Alshehri et al. (2017); Alshehri et al. (2018). Except for these four metrics, (Mondal and Bours 2017) proposes to use ANIA and ANGA to evaluate KCA algorithms. ANIA (the Average Number of Imposter Actions) shows how much an imposter can do before being locked out. And, ANGA (the Average Number of Genuine Actions) shows how much a genuine user can do before being locked out of the system wrongfully.

Datasets

As for datasets, most KCA studies use the dataset collected by themselves, making performance comparisons difficult. Huang et al. (2017) evaluates two existing KCA algorithms and their KDE-based KCA algorithm on four publicly available free-text keystroke datasets. The free text collection scenarios for the first three datasets: Torino dataset (Gunetti and Picardi 2005), Clarkson I dataset (Vural et al. 2014), Buffalo dataset (Sun et al. 2016), are similar in that users can answer some questions or do some tasks according to their situation. They have been used in Gunetti and Picardi (2005); Davoudi and Kabir (2009); Davoudi and Kabir (2010); Al Solami et al. (2011); Huang et al. (2017); Çeker and Upadhyaya (2016); Huang et al. (2017); Alshehri et al. (2017); Alshehri et al. (2018); Xiaofeng et al. (2019), respectively. The Clarkson II dataset (Murphy et al. 2017) is collected when the participants work in a completely uncontrolled, natural setting without any task. It has been used in Huang et al. (2017); Ayotte et al. (2019).

Though notable performances have been achieved with long or short text collected by doing tasks and long data collected in uncontrolled settings, (Huang et al. 2017) finds that KCA performance degrades significantly when applied to data collected in uncontrolled environments. It remains a challenge to use short keystroke inputs collected in nature settings for accurate KCA.

Methodology

In this section, we give the details of the proposed TKCA. We first introduce two definitions. Then, we present the keystroke model. Next, we brief how TKCA quickly discovers attackers by using short keystroke sequences and a majority vote.

Definitions

3-dimensional keystroke action. A 3-dimensional keystroke action is a tuple of the form $< key, ht, df >$, where key is the key name, ht is the hold time, and df is the digraph flight time for a keystroke. The key name is used as a feature because users' typing behavior relates to the specific key and context keys. Hold time and 2-graph flight time are selected because they are the most efficient (Gunetti and Picardi 2005) and always positive (consideration for embedding, latency time not selected).

keystroke sequence. A keystroke sequence is a series of consecutive 3-dimensional keystroke actions. Such that a keystroke sequence S can be formulated as a multivariate series of the form $\{< key_1, ht_1, df_1 >, < key_2, ht_2, df_2 >, ... < key_\omega, ht_\omega, df_\omega >\}$ with length of ω.

Keystroke model

In this work, we design a keystroke model to capture typing behavior in a keystroke sequence and convert it into an identity label. Figure 2 shows the keystroke model's network architecture, which contains five parts: input, embedding, LSTM, attention, and output.

Input The input is a keystroke sequence S in which each s_i is a 3-dimensional keystroke action, as described in "Definitions" section.

Embedding For learning, we need to find a way to convert key names to numeric values. We tried numbering keys in one or two dimensions according to their distribution on a keyboard. However, we discard this idea due to the different keyboard layouts and poor performance. Afterward, we find the embedding can solve this issue entirely. Embedding is a way to transform discrete variables into consecutive vector presentations. We use a key embedding to transform keys into digital vectors. And we use timing embeddings to convert hold times and digraph flight times into vectors for feature amplifying. As shown in Fig. 2, $Matrix_1$ is the key embedding, while $Matrix_2$ and $Matrix_3$ are the two time embeddings.

LSTM Because the user's typing behavior depends not only on the exact key but also on the context key he types. In the keystroke model, we use a bidirectional LSTM layer to extract the relationship between context keystrokes. As a variant of Recurrent Neural Networks (RNN), LSTM (Hochreiter and Schmidhuber 1997) has been proven effective for sequence tasks (LeCun et al. 2015) because it has "memory" units. It reads one input at a time, and remember some information/context through the hidden layer activations that get passed from one time-step to the next. This allows a unidirectional LSTM to take information from the past to process later inputs. A bidirectional LSTM (Bi-LSTM) can take context from both the past and the future. Because a user's typing behavior can be influenced by context keystrokes, in this work, we use Bi-LSTM to capture bidirectional keystroke dependencies in a sequence.

Attention The attention layer learns and combines the importance of each keystroke in a sequence. Otherwise, we use a dropout with a value of 0.5 in each LSTM layer and the attention layer to prevent over-fitting. And we use a fully connected sub-layer for further depth feature extraction.

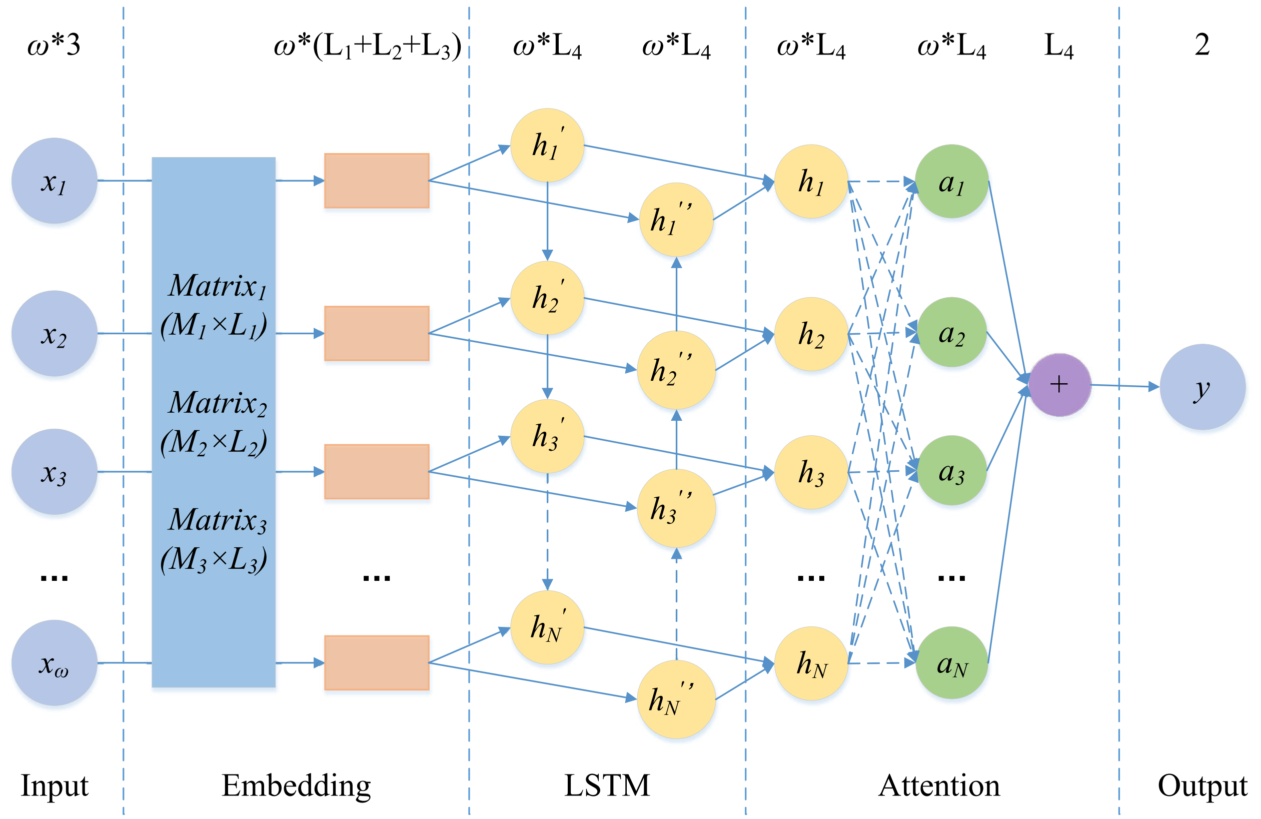

Fig. 2 Network architecture of the keystroke model. Where ω is the length of the keystroke sequence, M_1 is the total number of different keys in the dataset, M_2 is the number of *ht*, and M_3 is the number of *df*. L_1, L_2, and L_3 are the size of each vector in $Matrix_1$, $Matrix_2$, and $Matrix_3$. and L_4 is the size of hidden layers

Output We employ the softmax function to generate the similarity then use the argmax function to output an identity label. There are two labels: '0' and '1'. In this work, the label '0' indicates that the current user behaves similarly to the legitimate user. In contrast, '1' represents that the current user behaves differently from the legitimate user.

All parameters, including the three embedding matrixes in the model, are generated randomly and tuned through back-propagation when we train classifiers.

TKCA algorithm

For each legitimate user, we train a unique binary classifier based on the keystroke model. The TKCA uses the well-trained classifier to classify keystroke sequences typed by the current user and uses a majority vote fusing multiple classification results (labels) to verify if the current user is legitimate. For example, we use $2n + 1$ keystroke sequences once. A $\frac{n+1}{2n+1}$ majority vote means that the label with greater than n votes will be the final predicted label.

The pseudo-code for the TKCA algorithm is presented in Algorithm 1. The inputs are (i) the sequence length: ω, (ii) the limit HT of ht, (iii) the limit DT of df, and (iv) a $\frac{n+1}{2n+1}$ majority vote. The algorithm is triggered every once in a while or when a key is pressed. Once the current user types a key, the number corresponding to the key name *key*, the hold time *ht*, and the digraph flight time *df* of the keystroke are simultaneously recorded. A keystroke sequence S is formulated when the number of keystrokes reaches ω. Because there may be a long pause between keystrokes and some functional keys may be pressed long, such as "Ctrl" and "Shift", the *ht* and *df* are sometimes large and need to be limited in a certain time range. *ht* and *df* are checked to ensure that they are within limits HT and DT. Then, TKCA input S into the classifier. Classification result *label* will be used to renew the value of *sml*. And the total number of sequences *total* increases 1. When *total* reaches $2n + 1$, if *sml* is larger than n, the current user will be considered as the genuine user. *total* and *sml* will be reset to 0 and initiating a new round of continuous user authentication. Otherwise, the current user will be regarded as an intruder. Necessary measures are taken to prevent the intruder from using the computer terminal or cloud terminal, such as locking the screen, logging out the current user, generating an alarm log, or notifying the legitimate user via email or

Algorithm 1 TKCA algorithm.

```
Input: (i) the sequence length: $\omega$, (ii) the limit $HT$ of $ht$, (iii) the limit $DT$ of $df$, (iv) a $\frac{n+1}{2n+1}$ majority vote;

total = 0; sml = 0; cur_user = 0
init an empty queue $Q$
while $cur\_user == 0$ do
    enqueue the current 3-dimensional keystroke action $< key, ht, df >$ to $Q$
    if length($Q$) $>= \omega$ then
        formulate a keystroke sequence $S$ and clear $Q$
        for each $ht$ and $df$ in $S$ do
            check($ht$, $HT$), and check($df$, $DT$)
        classify $S$ and get $label$
        $sml$ += 1 - $label$
        $total$ += 1
        if $total == 2n + 1$ then
            if $sml > n$ then
                current user is legal
                $total$ = 0, $sml$ = 0
            else
                $cur\_user = 1$
                current user is illegal
return
```

message. The parameters in Algorithm 1 will be evaluated in "Experiments and evaluation" section.

Experiments and evaluation

To evaluate the proposed method, we conduct comprehensive experiments on the Clarkson II dataset. Experimental results show that TKCA achieves comparable performance. In the following section, we first introduce the dataset and implementation details. Then, we perform experiments to evaluate classifiers' performance and the TKCA algorithm and compare it with previous works.

Dataset

The Clarkson II dataset is collected in an entirely uncontrolled and natural setting (Murphy et al. 2017). A Windows-based logger loaded on users' computers passively records all keystrokes from natural behaviors, without any particular task. This dataset contains keystroke data from 103 participants. Keystroke events are timestamped of ticks (100-nanosecond intervals), while the system clock tick has a resolution of approximately 10-16 milliseconds. It means that the last 4 bits should not be taken into account.

Some users have multiple records of the same keystroke event with the same timestamp. We delete the redundant duplicate keystroke records and preserve all valid keystroke records. Even though a previous work (Huang et al. 2016) on this dataset shows that the performance can be improved by cleaning up "gibberish" keystrokes from the data. The amount of valid keystroke events varies greatly. Each user's average contribution is 98K, with a minimum value of 20 and a maximum value of 581K. The data records for each user in the Clarkson II dataset are ordering keystroke events formed by timestamp, key event (key down or key up), and key name. Before learning the typing behaviors in these data, we need to preprocess the data. We extract the key name, the hold time, and the digraph flight time of each keystroke to making up a 3-dimensional keystroke action.

Implementation details

Experimental setup We implement our proposed model based on the Pytorch framework (an open-source deep learning framework) and run on a computer configured with an NVIDIA Tesla P4 GPU, 32G RAM, 600G hard disk, and 12 CPU processors.

Training and testing plan As mentioned, the data amount of users in the Clarkson II dataset varies exceedingly. There are 88 users with more than 10K keystroke data. Before the experimental evaluation, we divide each user's keystroke data into two equal parts: one part only for training and another only for testing. For each user, we train a unique binary classifier based on the keystroke model. That is, there are 88 classifiers to be trained. And we randomly split the remaining 87 users into two categories: 43 known internal users and 44 unknown external users. Each classifier corresponds to a legitimate user, 43 internal users, and 44 external users. Internal users are the classifier knows their tying behaviors. Inversely, external users are the ones unknown to the classifier. To train each classifier, we use the keystroke sequences in the current user's training part as positive examples with the label '0' and randomly select the same number of keystroke sequences from internal users' training parts as negative examples with the label '1'.

To test each classifier, we use all test data of the legitimate user and randomly select an equal amount from 43 internal users' test data and another equal amount from 44 external users' test data, respectively. For every test keystroke sequence, we record whether this is a True Positive (TP), False Negative (FN), False Positive (FP), or True Negative (TN). In this manner, False Rejection Rate (FRR), False Acceptance Rate of internal users (iFAR), False Acceptance Rate of external users (eFAR), False Acceptance Rate (FAR), and Accuracy (ACC) for each classifier can be calculated using Eqs. (1)–(5), respectively. The mean value of performance of all classifiers is taken in the next comparative analysis. To obtain stable performance, we conduct each experiment five times and use the average results.

Table 2 The ACC (%) of classifiers with different values of *c*, *HT*, and *DT*

	***DT*=600**				***DT*=900**				***DT*=1200**			
c/HT	***180***	***210***	***240***	***270***	***180***	***210***	***240***	***270***	***180***	***210***	***240***	***270***
10	85.61	85.53	85.61	85.50	85.43	85.40	85.27	85.23	85.26	85.38	85.24	85.31
12	85.73	85.53	85.73	85.54	85.44	85.40	85.52	85.41	85.61	85.47	85.25	85.34
14	85.62	85.53	85.48	85.46	85.50	85.43	85.12	85.32	85.36	85.40	85.31	85.23
16	85.35	85.17	85.25	85.17	85.13	85.03	84.89	85.12	85.08	84.99	84.86	84.97

$$FRR = FN/(TP + FN) \tag{1}$$

$$iFAR = iFP/(iFP + iTN) \tag{2}$$

$$eFAR = eFP/(eFP + eTN) \tag{3}$$

$$FAR = (iFAR + eFAR)/2 \tag{4}$$

$$ACC = 1 - ((FRR + FAR)/2) \tag{5}$$

The performance of classifiers

As explained above, we will train a unique binary classifier based on the keystroke model for each legitimate user. The performance of classifiers can be influenced by:

1. The clock resolution c, the time limit HT and DT.
2. Different features.
3. Each part of the keystroke model.
4. The length of a keystroke sequence ω.

We set the keystroke sequence length ω to a value of 10 when testing other parameters. Besides, in the following experiments, the hyper-parameters M_1, L_1, L_2, L_3, and L_4 in the keystroke model are set to 133, 32, 24, 24, and 96, respectively. The value of M_1 is determined by counting the number of different keys and adding 1 (representing unknown). The values of L_1, L_2, L_3, and L_4 are derived from the grid search experiments. The hyper-parameters M_2 and M_3 are equal to HT/c, and DT/c. We do not adjust the best parameters for each classifier but global values based on all classifiers' average performance because we use keystroke data of other users to train the classifier for each legitimate user. Next, we detail the impact of the factors mentioned above on the performance of classifiers.

(I) The clock resolution c, the time limit HT and DT:

Since the number of vectors M_1 and M_2 (the size of $Matrix_2$ and $Matrix_3$ in the embedding layer) is related to the time limit HT, DT, and clock resolution c. The effects of these three parameters on the classifiers are tested together. The system clock resolution is about 10-16 milliseconds stated in Murphy et al. (2017). We calculate the median and average of all hold times for all users. They are 108 milliseconds and 151 milliseconds. And the median and average of all digraph flight times (less than 10 s) for all users are 184 milliseconds and 460 milliseconds, respectively. We select the values of c, HT, and DT according to the above information. Table 2 illustrates the impact of these three parameters on classifiers performance. The best accuracy is 85.73% when we set c to 12 milliseconds, DT to 600 milliseconds, and HT to 180 milliseconds or 240 milliseconds. Since the average hold time is 151 milliseconds, which is very close to 180 milliseconds, we choose to set HT to 240 milliseconds. If there is no individual declaration, we will arrange the clock resolution c to 12 milliseconds, the time limit HT to 240 milliseconds, and DT to 600 milliseconds in the following experiments.

(II) Different features:

In this paper, we assume that the combination of the key name (*key*), the hold time (*ht*), and the digraph flight time (*df*) can get better performance results than the combination of two of them. We test whether this hypothesis is correct by using different feature combinations. Table 3 shows the effect of each feature's absence on classifiers' performance. When using *ht* and *df* while *key* is absent or using *key* and *df* while *ht* is absent, the classifiers perform worse with an accuracy of 79.03% or 79.25%. When *df* is not used, the accuracy 82.79% eases the situation slightly. As supposed, the classifiers perform best with an accuracy of 85.73% when using all three features. The comparison illustrates the importance of using *key*, *ht*, and *df* as features in KCA studies.

(III) Each part of the keystroke model:

In this work, we design a keystroke model, as shown in Fig. 2. We compare the impact on classifiers performance when a part of the keystroke model is absent or when Bi-LSTM is replaced with a different deep learning model.

As shown in Table 4, the classifiers perform worse when there is no embedding layer or Bi-LSTM in the keystroke model than when there is no attention mechanism. And

Table 3 The effects of different feature combinations on the performance of classifiers

Features	**FRR (%)**	**iFAR (%)**	**eFAR (%)**	**ACC (%)**
with all	12.48	14.58	17.53	85.73
without key	18.77	22.17	24.16	79.03
without ht	17.76	22.50	24.98	79.25
without df	15.09	17.73	20.93	82.79

Table 4 The effect of different parts of the keystroke model on classifiers performance

Model parts	FRR (%)	iFAR (%)	eFAR (%)	ACC (%)
with all	12.48	14.58	17.53	85.73
without embedding	16.32	21.53	23.99	80.46
without LSTM	17.07	20.07	22.17	80.90
without attention	12.62	14.88	17.78	85.52

Table 6 The effects of the length of keystroke sequences ω on classifiers performance

ω	FRR (%)	iFAR (%)	eFAR (%)	ACC (%)
10	12.34	15.05	17.15	85.78
15	9.92	12.86	16.12	87.79
20	9.09	11.02	14.63	89.04
25	8.54	10.37	13.48	89.75

not having any of them results in a performance reduction of the classifiers. In particular, the classifiers' accuracy decreases by an average of 5.27% without using the embedding mechanism. And the accuracy of the classifiers reduces by an average of 4.83% without using the Bi-LSTM.

To verify the effectiveness of the Bi-directional LSTM, we compare the keystroke model based on Bi-LSTM with ANN, CNN, LSTM, Bi-RNN, and Bi-GRU (GRU is a variant of LSTM). Table 5 shows that the keystroke model's performance with Bi-LSTM is better than others. The accuracy of the keystroke model with Bi-GRU is close to but not superior to that based on Bi-LSTM.

(IV) The length of keystroke sequences: The effect of keystroke sequence length on classifiers performance is shown in Table 6. The accuracy of classifiers will increase with the increase of keystroke sequence length overall, and the speed of improvement decreases. We will evaluate the relationship between the number of keystrokes and the performance of TKCA in the following.

The performance of the TKCA algorithm

This part evaluates the TKCA algorithm under different lengths of keystroke sequences and majority vote mechanisms. We use all test data of the legitimate user, internal and external users. Here, we use the Equal Error Rate (EER) as the metric of the evaluation. EER is the point on a DET curve where FAR and FRR are equal. We also use the average of iFAR and eFAR to approximate the final FAR. But we will not use Eq. (5) to calculate ACC because the number of positive test samples is much smaller than the number of negative test samples. To get a set of FAR and FRR pairs, we adjust the threshold value of each classifier's sigmoid function instead of setting the threshold to a specific value of 0.5 as in the previous experiments.

Table 5 The effects of different neural networks on classifiers performance

Model	FRR (%)	iFAR (%)	eFAR (%)	ACC (%)
ANN	17.03	19.98	21.62	81.08
CNN	15.26	17.81	19.91	82.94
LSTM	12.95	15.47	18.43	85.04
Bi-RNN	14.39	17.00	19.64	83.64
Bi-GRU	13.02	15.07	17.89	85.24
Bi-LSTM	12.48	14.58	17.53	85.73

Figure 3 shows the DET curve of the TKCA algorithm using only one keystroke sequence at different values of the keystroke sequence length ω without majority vote. When the keystroke sequence length ω is 10, 15, 20, or 25, the EER is 13.51%, 11.34%, 10.09%, or 9.35%, respectively.

Different majority votes are evaluated on the four keystroke sequence lengths mentioned above. When the number of keystrokes in a sample is the same, the number of keystroke sequences is inverse to the keystroke sequence length ω. For example, there are 990 keystrokes in a sample. When ω is 10, the number of keystroke sequences is 99. We can use a $\frac{50}{99}$ majority vote. And when the ω is 25, the number of keystroke sequences is 39. We can use a $\frac{20}{39}$ majority vote. This work evaluates the sample size ranges from 10 to 990[1]. As shown in Fig. 4, the more keystrokes contain in a sample, the smaller the EER. The EER has a slight advantage with the ω of 10. The EER drops from 13.51% to 0.85% when keystrokes in a sample range from 10 to 990.

The comparison with previous KCA works

We compare the performance of TKCA with previous KCA algorithms on the Clarkson II dataset in Table 7. When a sample containing 1,000 keystrokes, the EER for Gunetti & Picardi's algorithm (Gunetti and Picardi 2005), Buffalo's SVM algorithm (Çeker and Upadhyaya 2016), and KDE based algorithm (Huang et al. 2017) tested on the Clarkson II dataset in Huang et al. (2017) are 10.36%, 15.67%, and 7.59%. Ayotte et al. (2019) improves KCA studies' performance by combining the results of KDE, Kolmogorov-Smirnov (KS), and Energy distance through majority rules and get an EER of 15.3% with 200 digraphs. It is 3.6% with 1,000 digraphs. Liverpool's DTW algorithm (Alshehri et al. 2017; 2018) gets a good result when test on the Clarkson I dataset with 100 keystrokes. But the best EER of it tested on the Clarkson II dataset is 45.25% with 125 keystrokes. Study (Xiaofeng et al. 2019) is outstanding on the Buffalo dataset with 30 keystrokes. However, the

[1] That is when ω=10, majority votes vary from 1/1 to 50/99. And when ω=15, majority votes differ from 1/1 to 33/65. When ω=20, they are range from 1/1 to 25/49. At last, when ω=25, they deviate from 1/1 to 20/39.

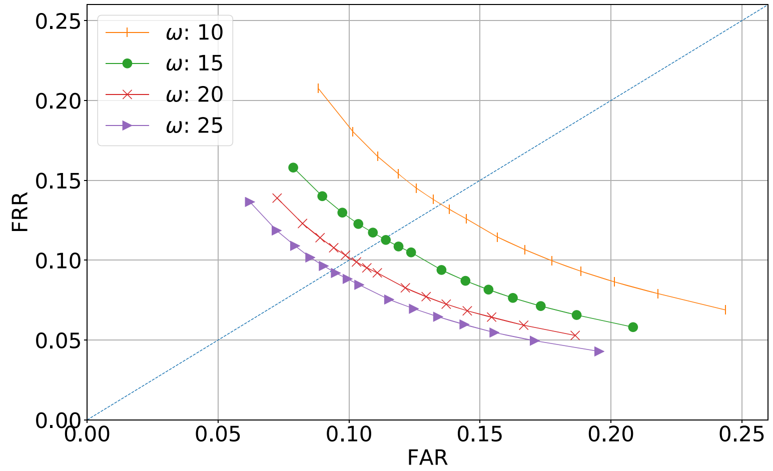

Fig. 3 DET curves for different values of the keystroke sequence length ω

best EER of it tested on the Clarkson II dataset is 20.46% with 70 keystrokes. Compared with these algorithms, our proposed TKCA algorithm achieves state-of-the-art performance with the EER of 8.28% when only using 30 keystrokes and 2.78% when using 190 keystrokes on the Clarkson II dataset.

Discussion and future work

TKCA can use short keystroke sequences for unconstrained input to identify attackers timely. For each legitimate user, TKCA trains a binary classifier and uses it to compare the typing behavior between the current user and the valid user. To make accurate decisions, TKCA combines several classification results by a majority vote. Once TKCA determines that the current user is not a logged-in user, necessary measures are taken to prevent the intruder from using the computer or cloud terminal. In "Experiments and evaluation" section, we have evaluated the TKCA algorithm's accuracy through a series of experiments.

About continuous authentication

At login time, multi-factor authentication (Wang and Wang 2016; Jiang et al. 2020; Qiu et al. 2020) can enhance authentication like passwords and PINs to address password or PIN leakage. After the login phase, unauthorized access could occur when a legitimate user forgets to log out and steps away from the terminal for lunch

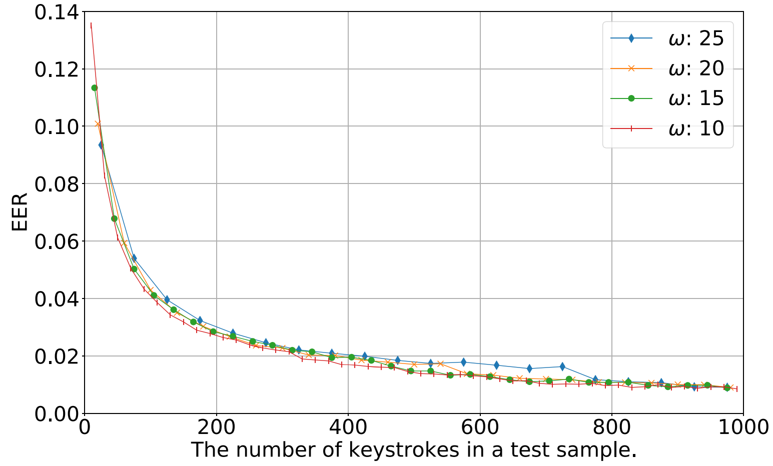

Fig. 4 The relationship between EER and the number of keystrokes in a sample

Table 7 The comparison of TKCA with previous KCA algorithms

Study	Method	Realized by	Sample Size	EER
(Gunetti and Picardi 2005)	'R' Distance	(Huang et al. 2017)	1,000	10.36%
	'A' Distance			
(Çeker and Upadhyaya 2016)	One-class SVM	(Huang et al. 2017)	1,000	15.67%
(Huang et al. 2017)	KDE	(Huang et al. 2017)	1,000	7.59%
(Alshehri et al. 2017)	DTW	This work	125	45.25%
(Ayotte et al. 2019)	KDE,	(Ayotte et al. 2019)	100,	35.1%,
	KS,		200,	15.3%,
	Energy		500,	6.3%,
			1,000	3.6%
(Xiaofeng et al. 2019)	CNN, RNN	This work	70	20.46%
TKCA	Embedding,	This work	30,	8.28%,
	Bi-LSTM,		70,	5.05%,
	Attention		90,	4.30%,
			190,	2.78%,
			490,	1.46%,
			990	0.85%

or an emergency or when the attacker has bypassed the login entry. The authentication during the login phase is not sufficiently applicable to detect this unauthorized access. Continuous user authentication based on behavioral biometrics can continuously authenticate users by collecting their physical or behavior information and analyzing it transparently to tackle this issue. It cannot replace traditional authentication or multi-factor authentication schemes but makes up for their shortcomings.

Continuous user authentication based on behavioral biometrics includes free-text keystroke dynamics (Monrose and Rubin 1997; Dowland and Furnell 2004; Gunetti and Picardi 2005; Janakiraman and Sim 2007; Sim and Janakiraman 2007; Montalvão Filho and Freire 2006; Davoudi and Kabir 2009; 2010; Harun et al. 2010; Stewart et al. 2011; Al Solami et al. 2011; Messerman et al. 2011; Rahman et al. 2011; Ferreira and Santos 2012; Bours 2012; Monaco et al. 2013; Deutschmann et al. 2013; Ahmed and Traore 2013; Kang and Cho 2015; Çeker and Upadhyaya 2016; Mondal and Bours 2017; Huang et al. 2017; Ayotte et al. 2019; Alshehri et al. 2017; 2018; Xiaofeng et al. 2019), mouse dynamics (Pusara and Brodley 2004; Ahmed and Traore 2007; Nakkabi et al. 2010; Zheng et al. 2011; Feher et al. 2012; Lin et al. 2012; Mondal and Bours 2013), touch screen inputs (Frank et al. 2012; Cai et al. 2013; Feng et al. 2014; Buschek et al. 2015), eye movements (Kinnunen et al. 2010; Eberz et al. 2015; 2016), gait pattern (Ailisto et al. 2005; Rong et al. 2007; Derawi et al. 2010), etc. We compare different biometrics continuous authentication from aspects of the environment (the settings of hardware, operating systems, and applications), assignment tasks, filtering data, stability, the requirement for additional hardware, implementation cost, and application scenarios in Table 8. Since every biometric factor of continuous authentication should be non-invasive and available after login, they are not shown in Table 8. To simulate the real usage scenarios, it needs to collect users' data in an uncontrolled environment without any tasks. Though filtering data as it is processed can improve accuracy, it also gives attackers chances. Pulse response, eye movement, and voice are relatively stable. But they require additional hardware, and the implementation cost is expensive. In traditional desktop or cloud desktop, free-text keystroke dynamics are proper for continuous user authentication. The proposed TKCA achieves a timely keystroke-based continuous user authentication in real scenarios without filtering data. In future work, we will combine free-text keystroke dynamics with other appropriate behavioral factors (i.e., mouse dynamics) in continuous authentication in desktop or cloud desktop.

The security strength

Since the TKCA is a continuous authentication based on free-text in uncontrolled settings, the acceptable keystroke sequences for a legitimate user are not only a fixed one or some. Some entropy metrics for user authentication (like passwords (Wang et al. 2017) or PINs (Wang et al. 2017)) are not suitable for evaluating the security strength of TKCA. In this work, we can use other users' test data for similarly evaluating the expected number of guesses to attack each legitimate user's classifier. When the TKCA does not use majority vote, and the keystroke sequence length ω is 10, 15,20, and 25, the expected number of attacks tested on the Clarkson II dataset is 5.75, 6.31, 6.90, and 7.15, respectively. However, the existing behavior-based continuous authentication schemes lack entropy evaluations. Beside s, the security strength and the accuracy of the classifiers have a positive correlation. Instead of using entropy, we evaluate the accuracy of classifiers and EER of the TKCA and compare the results with previous KCA studies in "Experiments and evaluation" section. Study (Eberz et al. 2017) find that the Equal Error Rate (EER), as well as derived metrics, are reported by the vast majority of continuous authentication papers.

The deployment concerns

Similar to other behavioral biometrics, typing behaviors may change over time and lead to a decrease in classifier

Table 8 The comparison of different biometrics-based continuous authentication

Biometrics	Study	Environment	Assign tasks	Filter data	Stability	Additional hardware	Implementation cost	Application scenarios
Keystroke	(Gunetti and Picardi 2005)	Uncontrolled	Yes	No	Medium	No	Cheap	Desktop Computers
dynamics	(Çeker and Upadhyaya 2016)	Uncontrolled	Yes	Yes				
	(Huang et al. 2017)	Uncontrolled	No	No				
	(Alshehri et al. 2017)	Uncontrolled	Yes	No				
	(Ayotte et al. 2019)	Uncontrolled	No	No				
	(Xiaofeng et al. 2019)	Uncontrolled	Yes	No				
	This work etc.	Uncontrolled	No	No				
Mouse	(Pusara and Brodley 2004)	Controlled	Yes	Yes	Medium	No	Cheap	Desktop Computers
dynamics	(Ahmed and Traore 2007)	Uncontrolled	No	Yes				
	(Nakkabi et al. 2010)	Uncontrolled	No	No				
	(Zheng et al. 2011)	Controlled	Yes	No				
	(Feher et al. 2012)	Uncontrolled	No	Yes				
	(Lin et al. 2012)	Uncontrolled	No	Yes				
	(Mondal and Bours 2013)	Uncontrolled	No	No				
	etc.							
Pulse	(Rasmussen et al. 2014)	Controlled	No	No	High	Yes	Expensive	Desktop Computers
response	etc.							
Eye	(Kinnunen et al. 2010)	Controlled	Yes	No	High	Yes	Expensive	Desktop Computers
movement	(Eberz et al. 2015)	Controlled	Yes	No				
	(Eberz et al. 2016)	Controlled	Yes	No				
	etc.							
Touch	(Frank et al. 2012)	Controlled	Yes	No	Medium	No	Cheap	Portable devices
screen	(Cai et al. 2013)	Controlled	Yes	No				
inputs	(Feng et al. 2014)	Uncontrolled	No	Yes				
	(Buschek et al. 2015)	Controlled	Yes	No				
	etc.							
Gait	(Ailisto et al. 2005)	Uncontrolled	No	No	Medium	No	Cheap	Portable devices
pattern	(Rong et al. 2007)	Uncontrolled	No	No				
	(Derawi et al. 2010)	Controlled	Yes	Yes				
	etc.							
Voice	(Feng et al. 2017)	Controlled	Yes	Yes	High	Yes	Expensive	Portable and IoT devices
	etc.							

accuracy. We can collect a valid user's latest keystroke data and feed it to the previously well-trained classifier. Specifically, we can save the keystroke sequence whose similarity to a legitimate user's behavior exceeds a threshold while TKCA is verifying a current user. After some time, when the number of keystroke sequences reaches a particular value, the keystroke data is fed to the classifier to learn the legitimate user's recent typing behaviors. Therefore, the TKCA can earn the changes in the legal user's typing behaviors and adjust the parameters in the classifier for this user usually. In this way, the TKCA can keep the classifier's accuracy as a user's typing behavior changes.

We get a superior performance by setting the keystroke sequence length to a fixed value in this work. In real-world deployment scenarios, the number of keystrokes entered varies for different users or at other times. We can use two criteria for partitioning the keystroke sequence according to the requirements in future work. A keystroke sequence is generated once the number of keystrokes is greater than a maximum sequence length or the pause time is greater than a maximum pause time. Otherwise, we can use the mechanisms in trusted computing to ensure the security of TKCA itself.

When collecting users' keystroke data, some privacy information may be involved. In this work, TKCA collects keystroke sequences by extracting 3-dimensional keystroke actions. TKCA will not save or use key names directly but assign every key a number to represent it. It can mitigate the possibility of privacy exposure somewhat. However, protecting users' privacy included in keystroke data in the whole life cycle while implementing timely KCA is a worthy concern.

In future work, we will focus on deploying the TKCA in a desktop or cloud desktop system and how to protect users' privacy included in keystroke data while achieving timely KCA.

Conclusion

This paper presents TKCA, a timely KCA method for continuous user authentication, which prevents attackers fast and accurately in uncontrolled environments. We have integrated the key name, the hold time, and the digraph flight time to improve classification accuracy. We have proposed a keystroke model using embedding, Bi-directional LSTM, and the attention mechanism to learn the depth keystroke features and analyze users' typing behavior. We have trained a unique binary classifier for every legitimate user based on the keystroke model. To further improve the accuracy, we have used a majority vote mechanism in TKCA. The comprehensive experiments demonstrate that our proposed method achieves outstanding performance on the Clarkson II dataset collected in a completely uncontrolled and natural setting. While for previous KCA studies, the EER is 20.46% or higher when using 70 keystrokes to valid a user. The TKCA gets an EER of 8.28% when only using 30 keystrokes and 2.78% when using 190 keystrokes to validate a user without filtering out any keystroke. It achieves the goal of using short keystroke sequences in uncontrolled and natural settings for timely and accurate keystroke-based continuous user authentication.

Abbreviations

TKCA: Timely keystroke-based continuous user authentication

Acknowledgements

The authors would like to thank the editors and reviewers for the constructive comments to improve the paper.

Authors' contributions

Lulu Yang: Investigation, Conceptualization, Methodology, Software, Data curation, Writing - Original draft preparation. Chen Li: Software, Validation. Ruibang You: Software, Validation. Bibo Tu: Supervision, Writing - Reviewing and Editing, Funding acquisition. Linghui Li: Suggestion of Methodology, Writing - Reviewing and revising. The author(s) read and approved the final manuscript.

Funding

This work is supported by the National Key R&D Program of China (Grant No.2016YFB0801002).

Availability of data and materials

The Clarkson II dataset can be accessed after signing an agreement with the authors of Murphy et al. (2017).

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Institute of Information Engineering, Chinese Academy of Sciences, 100093 Beijing, China. [2]School of Cyber Security, University of Chinese Academy of Sciences, 100049 Beijing, China. [3]Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, 100876 Beijing, China.

Received: 1 December 2020 Accepted: 1 February 2021
Published online: 03 May 2021

References

Ahmed AAE, Traore I (2007) A new biometric technology based on mouse dynamics. IEEE Trans Dependable Secure Comput 4(3):165–179

Ahmed AA, Traore I (2013) Biometric recognition based on free-text keystroke dynamics. IEEE Trans Cybern 44(4):458–472

Ailisto HJ, Lindholm M, Mantyjarvi J, Vildjiounaite E, Makela S-M (2005) Identifying people from gait pattern with accelerometers. In: Biometric Technology for Human Identification II. International Society for Optics and Photonics. Vol. 5779. pp 7–14

Al Solami E, Boyd C, Clark A, Ahmed I (2011) User-representative feature selection for keystroke dynamics. In: 2011 5th International Conference on Network and System Security. IEEE. pp 229–233

Alshehri A, Coenen F, Bollegala D (2017) Accurate continuous and non-intrusive user authentication with multivariate keystroke streaming. In: Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. SCITEPRESS-Science and Technology Publications. pp 61–70

Alshehri A, Coenen F, Bollegala D (2018) Iterative keystroke continuous authentication: A time series based approach. KI-Künstliche Intelligenz 32(4):231–243

Ayotte B, Huang J, Banavar MK, Hou D, Schuckers S (2019) Fast continuous user authentication using distance metric fusion of free-text keystroke data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp 0–0

Bours P (2012) Continuous keystroke dynamics: A different perspective towards biometric evaluation. Inf Secur Tech Rep 17(1-2):36–43

Buschek D, De Luca A, Alt F (2015) Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp 1393–1402

Cai Z, Shen C, Wang M, Song Y, Wang J (2013) Mobile authentication through touch-behavior features. In: Chinese Conference on Biometric Recognition. Springer. pp 386–393

Çeker H, Upadhyaya S (2016) User authentication with keystroke dynamics in long-text data. In: 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS). IEEE. pp 1–6

Davoudi H, Kabir E (2009) A new distance measure for free text keystroke authentication. In: 2009 14th International CSI Computer Conference. IEEE. pp 570–575

Davoudi H, Kabir E (2010) Modification of the relative distance for free text keystroke authentication. In: 2010 5th International Symposium on Telecommunications. IEEE. pp 547–551

Derawi MO, Nickel C, Bours P, Busch C (2010) Unobtrusive user-authentication on mobile phones using biometric gait recognition. In: 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IEEE Computer Society. pp 306–311

Deutschmann I, Nordström P, Nilsson L (2013) Continuous authentication using behavioral biometrics. IT Prof 15(4):12–15

Dowland PS, Furnell SM (2004) A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In: IFIP International Information Security Conference. Kluwer / Springer. pp 275–289

Eberz S, Rasmussen K, Lenders V, Martinovic I (2015) Preventing lunchtime attacks: Fighting insider threats with eye movement biometrics. In: 22nd Annual Network and Distibuted System Security Symposium, NDSS 2015, San Diego, California, USA, February 8–11, 2015. Internet Society

Eberz S, Rasmussen KB, Lenders V, Martinovic I (2016) Looks like eve: Exposing insider threats using eye movement biometrics. ACM Trans Priv Secur 19(1):1–31

Eberz S, Rasmussen KB, Lenders V, Martinovic I (2017) Evaluating behavioral biometrics for continuous authentication: Challenges and metrics. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp 386–399

Feher C, Elovici Y, Moskovitch R, Rokach L, Schclar A (2012) User identity verification via mouse dynamics. Inf Sci 201:19–36

Feng H, Fawaz K, Shin KG (2017) Continuous authentication for voice assistants. In: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking. pp 343–355

Feng T, Yang J, Yan Z, Tapia EM, Shi W (2014) Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In: Proceedings of the 15th Workshop on Mobile Computing Systems and Applications. pp 1–6

Feng T, Zhao X, Carbunar B, Shi W (2013) Continuous mobile authentication using virtual key typing biometrics. In: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE Computer Society. pp 1547–1552

Ferreira J, Santos H (2012) Keystroke dynamics for continuous access control enforcement. In: 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. IEEE Computer Society. pp 216–223

Frank M, Biedert R, Ma E, Martinovic I, Song D (2012) Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Trans Inf Forensics Secur 8(1):136–148

Gunetti D, Picardi C (2005) Keystroke analysis of free text. ACM Trans Inf Syst Secur 8(3):312–347

Harun N, Woo WL, Dlay S (2010) Performance of keystroke biometrics authentication system using artificial neural network (ann) and distance classifier method. In: International Conference on Computer and Communication Engineering (ICCCE'10). IEEE. pp 1–6

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Huang J, Hou D, Schuckers S, Law T, Sherwin A (2017) Benchmarking keystroke authentication algorithms. In: 2017 IEEE Workshop on Information Forensics and Security (WIFS). IEEE. pp 1–6

Huang J, Hou D, Schuckers S, Upadhyaya S (2016) Effects of text filtering on authentication performance of keystroke biometrics. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE. pp 1–6

Janakiraman R, Sim T (2007) Keystroke dynamics in a general setting. In: International Conference on Biometrics. Springer. pp 584–593

Jiang Q, Zhang N, Ni J, Ma J, Ma X, Choo K-KR (2020) Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles. IEEE Trans Veh Technol 69(9):9390–9401

Joyce R, Gupta G (1990) Identity authentication based on keystroke latencies. Commun ACM 33(2):168–176

Kang P, Cho S (2015) Keystroke dynamics-based user authentication using long and free text strings from various input devices. Inf Sci 308:72–93

Killourhy KS, Maxion RA (2009) Comparing anomaly-detection algorithms for keystroke dynamics. In: 2009 IEEE/IFIP International Conference on Dependable Systems & Networks. IEEE Computer Society. pp 125–134

Kinnunen T, Sedlak F, Bednarik R (2010) Towards task-independent person authentication using eye movement signals. In: Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications. pp 187–190

Lau S, Maxion R (2014) Clusters and markers for keystroke typing rhythms. In: The {LASER} Workshop: Learning from Authoritative Security Experiment Results ({LASER} 2014). pp 1–10

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Lin C-C, Chang C-C, Liang D (2012) A new non-intrusive authentication approach for data protection based on mouse dynamics. In: 2012 International Symposium on Biometrics and Security Technologies. IEEE Computer Society. pp 9–14

Messerman A, Mustafić T, Camtepe SA, Albayrak S (2011) Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In: 2011 International Joint Conference on Biometrics (IJCB). IEEE Computer Society. pp 1–8

Monaco JV, Bakelman N, Cha S-H, Tappert CC (2013) Recent advances in the development of a long-text-input keystroke biometric authentication system for arbitrary text input. In: 2013 European Intelligence and Security Informatics Conference. IEEE. pp 60–66

Mondal S, Bours P (2013) Continuous authentication using mouse dynamics. In: 2013 International Conference of the BIOSIG Special Interest Group (BIOSIG). GI. pp 1–12

Mondal S, Bours P (2017) A study on continuous authentication using a combination of keystroke and mouse biometrics. Neurocomputing 230:1–22

Monrose F, Reiter MK, Wetzel S (2002) Password hardening based on keystroke dynamics. Int J Inf Secur 1(2):69–83

Monrose F, Rubin A (1997) Authentication via keystroke dynamics. In: Proceedings of the 4th ACM Conference on Computer and Communications Security. pp 48–56

Montalvão Filho JR, Freire EO (2006) On the equalization of keystroke timing histograms. Pattern Recognit Lett 27(13):1440–1446

Moskovitch R, Feher C, Messerman A, Kirschnick N, Mustafic T, Camtepe A, Lohlein B, Heister U, Moller S, Rokach L, et al (2009) Identity theft, computers and behavioral biometrics. In: 2009 IEEE International Conference on Intelligence and Security Informatics. IEEE. pp 155–160

Murphy C, Huang J, Hou D, Schuckers S (2017) Shared dataset on natural human-computer interaction to support continuous authentication research. In: 2017 IEEE International Joint Conference on Biometrics (IJCB). IEEE. pp 525–530

Nakkabi Y, Traoré I, Ahmed AAE (2010) Improving mouse dynamics biometric performance using variance reduction via extractors with separate features. IEEE Trans Syst Man Cybern Syst Hum 40(6):1345–1353

Pusara M, Brodley CE (2004) User re-authentication via mouse movements. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security. pp 1–8

Qiu S, Wang D, Xu G, Kumari S (2020) Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices. IEEE Trans Dependable Secure Comput. IEEE

Rahman KA, Balagani KS, Phoha VV (2011) Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes. In: CVPR 2011 Workshops. IEEE. pp 31–38

Rasmussen KB, Roeschlin M, Martinovic I, Tsudik G (2014) Authentication using pulse- response biometrics. In: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23–26, 2014. Internet Society

Rong L, Zhiguo D, Jianzhong Z, Ming L (2007) Identification of individual walking patterns using gait acceleration. In: 2007 1st International Conference on Bioinformatics and Biomedical Engineering. IEEE. pp 543–546

Sim T, Janakiraman R (2007) Are digraphs good for free-text keystroke dynamics? In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society. pp 1–6

Stewart JC, Monaco JV, Cha S-H, Tappert CC (2011) An investigation of keystroke and stylometry traits for authenticating online test takers. In: 2011 International Joint Conference on Biometrics (IJCB). IEEE Computer Society. pp 1–7

Sun Y, Ceker H, Upadhyaya S (2016) Shared keystroke dataset for continuous authentication. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE. pp 1–6

Syed Z, Banerjee S, Cukic B (2016) Normalizing variations in feature vector structure in keystroke dynamics authentication systems. Software Qual J 24(1):137–157

Vural E, Huang J, Hou D, Schuckers S (2014) Shared research dataset to support development of keystroke authentication. In: IEEE International Joint Conference on Biometrics. IEEE. pp 1–8

Wang D, Cheng H, Wang P, Huang X, Jian G (2017) Zipf's law in passwords. IEEE Trans Inf Forensics Secur 12(11):2776–2791

Wang D, Gu Q, Huang X, Wang P (2017) Understanding human-chosen pins: characteristics, distribution and security. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp 372–385

Wang D, Wang P (2016) Two birds with one stone: Two-factor authentication with security beyond conventional bound. IEEE Trans Dependable Secure Comput 15(4):708–722

Wu J-S, Lin W-C, Lin C-T, Wei T-E (2015) Smartphone continuous authentication based on keystroke and gesture profiling. In: 2015 International Carnahan Conference on Security Technology (ICCST). IEEE. pp 191–197

Xiaofeng L, Shengfei Z, Shengwei Y (2019) Continuous authentication by free-text keystroke based on CNN plus RNN. Procedia Comput Sci 147:314–318

Zheng N, Paloski A, Wang H (2011) An efficient user verification system via mouse movements. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. pp 139–150

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.