Recurrent neural network-based user authentication for freely typed keystroke data

Junhong Kim [1] **Pilsung Kang** [1]

Abstract

Keystroke dynamics-based user authentication (KDA) based on long and freely typed text is an enhanced user authentication method that can not only identify the validity of current users during login but also continuously monitors the consistency of typing behavior after the login process. Previous long and freely typed text-based KDA methods had difficulty incorporating the key sequence information and handling variable lengths of keystrokes, which in turn resulted in lower authentication performance compared to KDA methods based on short and fixed-length text. To overcome these limitations, we propose a recurrent neural network (RNN)-based KDA model. As the RNN model can process an arbitrary length of input and target sequences, our proposed model takes two consecutive keys as the input sequence and actual typing time for the corresponding key sequence as the target sequence. Based on experimental results involving 120 participants, our proposed RNN–KDA model yielded the best authentication performance for all training and test length combinations in terms of equal error rate (EER). It achieved a 5%–6% EER using only 10 test keystrokes while the EERs of other benchmark methods were above 20%. In addition, its performance steadily and more rapidly improves compared to the benchmark methods when the length of training keystrokes increases.

Keywords: *Keystroke dynamics-based user authentication, free text, recurrent neural network, long short-term memory*

1. Introduction

We live in a hyper-connected world where not only are human beings connected to each other, but where things are also connected to human beings or even to other things (e.g., the internet of things (IoT)) (Vermesan & Friess, 2015). With this rapidly increasing network connectivity, the development of mobile devices provides great convenience by recording and analyzing users' behaviors; however, the scope of security system also increases (Farooq et al., 2015; Zhang et al., 2014) and security-related concerns are consistently growing, especially for protecting personal information and corporate data from unauthorized users (Lou & Ren, 2009; Hiltgen et al., 2006; Pisani & Lorena, 2015). Consequently, companies are investing heavily in hardware and software for security systems, but many network systems remain vulnerable to security breaches, leading to personal/corporate information leakage/loss. As a fundamental building block for security systems, user authentication grants an authorized user access to a request only when a set of required information is matched. The most widely adopted authentication system is to compare the identification (ID) and the corresponding password during login (Yan et al., 2004; Kang & Cho, 2009; Biddle et al., 2011).

Distinctive characteristics of password-based authentication methods are ease of development, operation, and maintenance. However, if a particular third party obtains a valid password, the security system becomes useless (Yan et al., 2004; Kang & Cho, 2009; Kang, 2015). To solve this problem, user authentication systems using biometric features as a secondary authentication method to password strings have been explored (Clarke & Furnell, 2005; Ma et al., 2003; Prabhakar et al., 2003; Peacock et al., 2004). Biometric features such as fingerprints, iris, voice, and blood vessels do not require users to carry separate security devices, e.g., a one-time password. In addition, these biometric-based user authentication methods have yielded satisfactory authentication performance in many studies (Ma et al., 2003; Yager & Dun-

[1] School of Industrial Management Engineering, Korea University, Seoul, South Korea. Correspondence to: Pilsung Kang <pilsung_kang@korea.ac.kr>.

stone, 2010; Connolly et al., 2012; Voth, 2003; Shu & Zhang, 1998). However, many biometric feature-based authentication systems require additional hardware to read the biometric information, so the implementation scope is limited to certain devices in which recognition hardware is installed. To overcome this limitation, keystroke dynamics-based user authentication (KDA), which does not require any recognition hardware and can be fully implemented by software programs, has been highlighted in (Pisani & Lorena, 2015; Stefan et al., 2012; Ngugi et al., 2011; Gunetti & Picardi, 2005; Uzun & Bicakci, 2012; Alpar, 2017; Alsultan et al., 2017; Banerjee & Woodard, 2012; Teh et al., 2013).

KDA is a method for user authentication based on quantitative data collected from each user's typing behavior. As each person has his/her own typing style or habit, this unconscious behavioral pattern can be used to identify or recognize a valid user very accurately (Joyce & Gupta, 1990; Kang et al., 2008). Accordingly, KDA can be a practically useful alternative method to solve the limitation of password-based authentication among the various biometric feature-based user authentication methods because of its hardware independence (Bleha et al., 1990; Karnan et al., 2011; Monrose et al., 2002; Bhatt & Santhanam, 2013). Recently, not only has keystroke dynamics been applied to user authentication, but also to predict someone's emotion (Nahin et al., 2014; Kołakowska, 2013) or to identify the gender of the typist (Fairhurst & Da Costa-Abreu, 2011).

Early studies of KDA mainly focused on the keystroke dynamics data collected when users type their IDs and passwords during the login process. Under this circumstance, the password strings are known and the length of the password is fixed (Bleha et al., 1990; Fairhurst & Da Costa-Abreu, 2011; Chang, 2012; Hosseinzadeh & Krishnan, 2008). Although the authentication performance based on keystroke dynamics collected from known and fixed-length text is close to that of passive biometric feature-based authentication methods, it is not possible to determine whether the current user is the valid user or not once access is granted. A simple solution to this problem is to make the system ask users to type their passwords periodically, which inconveniences users (at the very least). To solve this problem, KDA based on freely typed text instead of known and fixed-length text has been studied (Kang, 2015; Gunetti & Picardi, 2005; Montalvão Filho & Freire, 2006; Teh et al., 2011; Kang & Cho, 2015; Monrose & Rubin, 1997; Gunetti & Ruffo, 1999; Dowland et al., 2001; Villani et al., 2006; Curtin et al., 2006; Janakiraman & Sim, 2007; Buch et al., 2008; Hu et al., 2008; Hempstalk et al., 2008; Davoudi & Kabir, 2009; Samura & Nishimura, 2009; Davoudi & Kabir, 2010b; Park et al., 2010; Messerman et al., 2011; Singh & Arya, 2011; Chantan et al., 2012; Bakelman et al., 2012; Bours, 2012; Kim et al., 2018). However, authentication performances of freely typed text-based KDA methods are below those of KDA based on known and fixed-length text.

Performance degeneration of freely typed text-based KDA occurs mainly because past studies did not fully utilize the information available from the keystroke data, and some methods used over-abstracted information. For example, the Kolomogorov-Sminrov (K–S) statistic or Cramér-von Mises criterion compare two empirical distributions of a particular keystroke timing, e.g., down-down time (the time between pressing two consecutive keys), between two different keystroke datasets. With these methods, key sequence information is completely ignored, which can be a significant clue to identify personal typing behavior (Park et al., 2010). The R measure and A measure (Gunetti & Picardi, 2005), which are other well-known freely typed text-based KDA methods, use key sequence information, but they compare the relative order of typing speed (R measure) or the proportion of common digraphs with similar typing speed (A measure), which are also summarized information of typing time. For machine learning-based KDA methods, keystroke data must be transformed into a fixed-size numerical vector (Kang & Cho, 2015; Kim et al., 2018). Although many machine learning-based methods attempt to preserve the information of original keystroke data as much as possible, information loss is inevitable during the transformation step. Even worse, most machine learning-based KDA methods require an impostor dataset during the model training to find the optimal hyper-parameters for the authentication algorithms (Kim et al., 2018), which is unrealistic in practice, where the authentication model must be built based only on a valid user's keystroke data.

To improve authentication performance of freely typed text-based KDA, we propose a recurrent neural network (RNN)-based authentication model. As two types of information are available during typing, i.e., key sequence information (which key is pressed and released in a certain sequence) and typing time (when a key is pressed and when it is released), we develop an RNN model to predict the actual elapsed typing times for every digraph in a given key sequence. In this RNN model, two consecutively typed keys are used as the input, and a valid user's actual typing times are used as the target. Therefore, a separate RNN model is trained for each user. To preserve long range dependency, which may be caused by the typing habit of a user, we adopted a long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) model in the RNN structure. Once the RNN model is sufficiently trained, a new set of key sequences is provided to predict the typing time and the difference between the predicted time and actual typing time is used as a novelty score. If the RNN model is properly trained, it learns a distinctive typing behavior of the corresponding user, so that the prediction error for the corresponding user will be low but for other users will be high. By formulating the RNN model to predict the user's typing time and using the loss from the RNN model as the novelty score, we do not require impostor's data during training, which is the main drawback of most current machine learning-based authentication models.

We believe that our proposed RNN–KDA model based on freely typed text has the following advantages over existing statistical or machine learning-based methods. First, our proposed model will be able to identify impostor users more accurately than existing methods. Although existing methods require a relatively large amount of test keystroke data to correctly authenticate users, our model can identify the user correctly with a relatively small amount of keystroke data, which helps to reduce damage from criminal attempts such as messenger-phishing. Second, the authentication model is strengthened by using a valid user's keystroke data collected after model training by efficiently updating the RNN–KDA model. Although other machine learning-based authentication algorithms must be completely retrained when a new dataset becomes available, the RNN–KDA model can be updated from the current configuration by adjusting the network weights to reflect the newly added data.

The remainder of this paper is organized as follows. Chapter 2 discusses previous studies mainly focusing on freely typed text-based KDA and demonstrates the necessity for the proposed RNN–KDA. Chapter 3 discusses data collection and preprocessing methods, followed by introducing commonly used freely typed text-based user authentication models. Chapter 4 describes the proposed RNN-based KDA method. Chapter 5 presents the experimental settings and Chapter 6 discusses the experimental results. Finally, Chapter 7 concludes the study and discusses future research directions.

2. Related Work

Over the past few decades, a number of freely typed text-based KDA methods have been studied. In this section, we briefly introduce some significant studies and discuss their limitations and the necessity for our study.

To the best of our knowledge, the first research on user authentication based on freely typed text was conducted by Monrose & Rubin (1997). They collected short and fixed texts and long and freely typed texts from 42 participants over seven weeks. They performed KDA by computing the Euclidean distance of the up-down time (time elapsed between releasing a key and pressing the following key) and down-up time (time elapsed between pressing a key and releasing it) between the common digraphs. Experimental results showed that 90% authentication accuracy was achieved for short and fixed texts, but only 23% authentication accuracy was achieved for long and freely typed text, which implies that user authentication for long and freely typed text is much more difficult than for short and fixed text.

Since then, long and freely typed text-based KDA has been studied more actively. Dowland et al. (2001) used the mean and standard deviation of the up-down time obtained from the common digraphs typed more than a certain number to compute the authentication score. This simple authentication score yielded approximately 50% authentication accuracy. Davoudi & Kabir (2010a;b) used only the down-down time of digraphs. To compute the authentication score, they adopted a modified relative distance and weighted relative distance. The two distance metrics resulted in 0.08%/0.07% false acceptance rate (FAR) and 18.8%/15.2% false rejection rate (FRR), respectively.

Gunetti & Picardi (2005) used N-graph based down-down times, e.g., tri-graphs, in addition to commonly used digraph-based down-down times. As authentication scores, they proposed the relative measure (R measure) and the absolute measure (A measure). Experimental results showed 0.005% FAR and 5% FRR for 205 participants. Park et al. (2010) attempted to utilize the keyboard layout information by dividing the keyboard area into four distinctive regions: keys typed by the left hand, keys typed by the right hand, space bar, and backspace bar. Based on this location information, a total of 16 digraph-based down-down times were used to compute the authentication score, which resulted in a 8.92% equal error rate (EER).

Kang & Cho (2015) attempted to extend the applicability of long and free text-based user authentication from a traditional input device, i.e., a PC keyboard, to other input devices, such as soft keyboards (typed using a stylus pen) and touch keyboard (typed using either one hand or two hands). Based on the keystroke dataset collected from 35 participants, they tested eight authentication score computation methods: mean and variance equality test, K–S test, C–M criterion, digraph matrix distance, R measure, A measure, RA measure, and R+A measure. For PC keyboards, distribution-based metrics, i.e., K–S statistic and C–M criterion, performed well when the number of keystrokes was relatively few, but R and A based measures achieved lower than 5% EER only when a sufficient (higher) number of keystrokes were available. Although the authentication performance for keyboards is practically acceptable, it is degenerated for other input devices.

Recently, machine learning-based KDA methods have been proposed for better authentication performance than the aforementioned statistical or heuristic methods. As only the valid user's keystroke data is available during the model training, a one-class classification strategy is commonly adopted. Hempstalk et al. (2008) used Gaussian density and expectation–maximization classifiers based on up-down time for the digraph, duration for each key, typing speed, error rate, and press/release ordering. Ten participants were involved in the experiment and their models exhibited 11.3% false acceptance rate (FAR) and 20.4% false rejection rate (FRR). Kang & Cho (2015) divided the keyboard layout to extract a set of fixed length features from long and free texts. All keys are grouped into three categories (keys typed by the left hand, right hand, and the spacebar), based on the average down-down time of the eight digraph combinations being computed. Experimental results involving 35 participants resulted in an average EER of 5.63% when a sufficient number of keystrokes are available. Kim et al. (2018) proposed a user-adaptive feature extraction method by considering an individual user's different typing speeds. They grouped the unigraphs and digraphs into eight categories based on the user's relative typing speed. One unigraph time and four digraph times (down-down time, down-up time, up-down time, and up-up time) were used, resulting in a total of 40 features extracted for training the one-class classification algorithms. They tested five one-class classification algorithms, i.e., Gaussian density estimation, Parzen window density estimation, one-class support vector machine, k-nearest neighbor, and K-Means clustering, on the keystrokes dataset collected from 150 participants. Their model achieved 3.38% EER (with 500 training keystrokes) and 3.36% EER (with 1,000 keystrokes) when 100 test keystrokes were used to compute the authentication score.

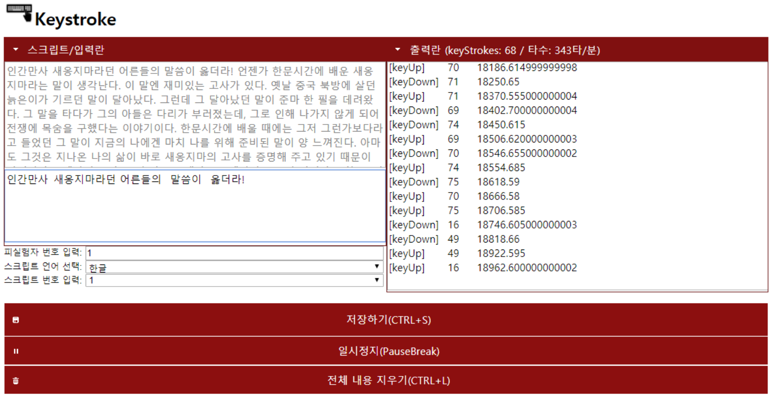
Although machine learning-based KDA models have achieved favorable authentication performance, one critical restriction of these models is that they cannot cope with different keystroke lengths as long as a fixed number of features are extracted from a collection of keystrokes. For example, regardless of training or test keystroke lengths, whether 100, 500, or 1,000, these keystrokes are converted to an 8-dimensional vector (Kang & Cho, 2015) or a 40-dimensional vector (Kim et al., 2018). Consequently, these methods do not train patterns embedded between individual digraphs, but rather train a summarized or abstracted pattern between them. In addition, some machine learning-based KDA methods require both valid user and impostor keystroke data to optimize the hyper-parameters, e.g., k in k-nearest neighbor.

The main differences between this study and previous studies for long and free text-based KDA are as follows. First, our method does not transform a set of keystroke data into a fixed set of feature vectors, but simply uses raw keystroke data (a sequence of keys typed and the corresponding elapsed time) as an input of the KDA model. By doing so, our model can deal with variable lengths of keystrokes and discover the user's individual digraph-level typing pattern. Second, our model can be built based solely on valid users' keystroke data, so the proposed model can be implemented in real systems without any practical obstacles.

3. Data collection and preprocessing

To collect keystroke data, we developed a web-based keystroke data collector as shown in Figure 1. The script (in Korean) assigned to each participant is displayed in the top left, whereas actually typed texts are shown in the left middle. In the top right, a key identifier and time (in milliseconds) when the participant presses a key are sequentially displayed. A total of 120 participants were involved in our study. Some basic statistics of the collected keystroke data are shown in Table 1. The participants provided 17,860 keystrokes on average and with a minimum of 13,670 keystrokes. These variations in keystrokes length are mainly caused by the length of the scripts assigned to different users, and some users used the delete key and the backspace key more frequently than other users, resulting in longer keystrokes.

Based on the collected raw keystroke data, we extracted five types of keystroke timing values: duration, down-down (DD)-time, up-down (UD)-time, up-up (UU)-time, and down-up (DU)-time. Duration is the time between pressing and releasing the same key. DD-time is the time between pressing a key and pressing the next key, whereas UU-time is the time between releasing a key and releasing the next key. UD-time is



Keystroke

스크립트/입력란

인간만사 새옹지마라던 어른들의 말씀이 옳더라! 언젠가 한문시간에 배운 새옹지마라는 말이 생각난다. 이 말엔 재미있는 고사가 있다. 옛날 중국 북방에 살던 늙은이가 기르던 말이 달아났다. 그런데 그 달아났던 말이 준마 한 필을 데려왔다. 그 말을 타다가 그의 아들은 다리가 부러졌는데, 그로 인해 나가지 않게 되어 전쟁에 목숨을 구했다는 이야기이다. 한문시간에 배울 때에는 그저 그런가보다라고 들었던 그 말이 지금의 나에겐 마치 나를 위해 준비된 말이 양 느껴진다. 아마도 그것은 지나온 나의 삶이 바로 새옹지마의 고사를 증명해 주고 있기 때문이

인간만사 새옹지마라던 어른들의 말씀이 옳더라!

피실험자 번호 입력: 1

스크립트 언어 선택: 한글

스크립트 번호 입력: 1

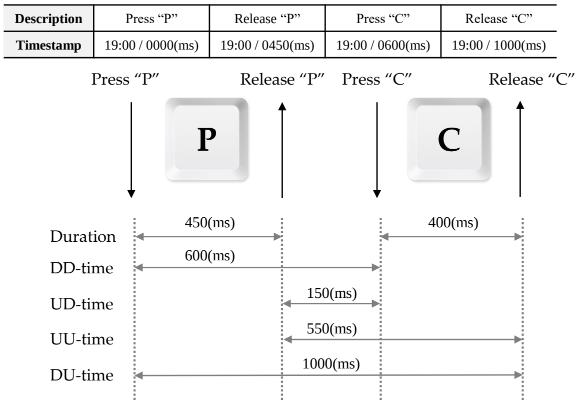
출력란 (keyStrokes: 68 / 타수: 343타/분)

```
[keyUp]     70    18186.614999999998
[keyDown]   71    18250.65
[keyUp]     71    18370.555000000004
[keyDown]   69    18402.700000000004
[keyDown]   74    18450.615
[keyUp]     69    18506.620000000003
[keyDown]   70    18546.655000000002
[keyUp]     74    18554.685
[keyDown]   75    18618.59
[keyUp]     70    18666.58
[keyUp]     75    18706.585
[keyDown]   16    18746.605000000003
[keyDown]   49    18818.66
[keyUp]     49    18922.595
[keyUp]     16    18962.600000000002
```

Figure 1. Web-based PC keystroke data collector

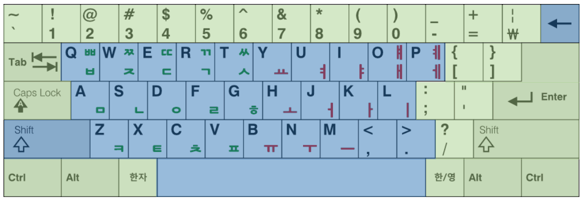
Table 1. Summary of statistics from collected keystroke data

No. of Participation	Min	1st Qtr	Median	Mean	3rd Qtr	Max
120	13,670	17,270	17,740	17,860	18,390	20,760



Description	Press "P"	Release "P"	Press "C"	Release "C"
Timestamp	19:00 / 0000(ms)	19:00 / 0450(ms)	19:00 / 0600(ms)	19:00 / 1000(ms)

Figure 2. Time capture moments and five basic keystroke times



~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 _ - + = ¦ ₩
Tab Q ㅃ ㅂ W ㅉ ㅈ E ㄸ ㄷ R ㄲ ㄱ T ㅆ ㅅ Y ㅛ U ㅕ I ㅑ O ㅒ ㅐ P ㅖ ㅔ { [}]
Caps Lock A ㅁ S ㄴ D ㅇ F ㄹ G ㅎ H ㅗ J ㅓ K ㅏ L ㅣ : ; " ' Enter
Shift Z ㅋ X ㅌ C ㅊ V ㅍ B ㅠ N ㅜ M ㅡ < , > . ? / Shift
Ctrl Alt 한자 한/영 Alt Ctrl

Figure 3. Keysets on the PC keyboard used for this study

the time between releasing a key and pressing the next key, whereas DU-time is the time between pressing a key and releasing the next key. Figure 2 illustrates an example of computing these five timing values.

In this study, we used 32 key identifiers to compute the above five key timing values: 31 individual keys and special single-key sets as shown in Figure 3. The 31 keys in blue color are individual keys consisting of consonants, vowels, shift, period, and backspace. The remaining keys in green color are regarded as single keys. Hence, pressing a key on the keyboard is transformed into a 32-dimensional one-hot encoded vector (only the key actually typed has a value of 1 and other keys have values of 0). If the DD-Time of a key exceeds one second (1,000 ms), we assume that the participant pauses the typing and we eliminate these keystrokes from his/her keystroke dataset.

4. RNN-based KDA Model

RNN is a specialized neural network used to process sequential data, which has shown excellent performance in various sequential data applications such as text, voice, and time-series signals (Graves & Jaitly, 2014; Graves et al., 2013; Zhang & LeCun, 2017; Liu et al., 2016). Keystroke data is also generated sequentially, so RNN is well suited for the KDA system. One of the main difficulties in training RNN is known as gradient vanishing or the exploding problem; a gradient cannot be appropriately propagated when the length of a sequence increases beyond a certain point (Karpathy et al., 2015). Long short-term memory (LSTM) resolves this problem by adaptively adjusting the information to be forgotten and memorized by using the cell state inside the RNN module (Hochreiter & Schmidhuber, 1997). By doing so, LSTM can deal with long range dependency and has worked well in various real applications (Hochreiter et al., 2001). Hence, we adopt the LSTM structure to build the KDA model. We used the basic LSTM cell in our study because several studies have reported that it exhibits no significant performance differences compared to other RNN cell variants (Jozefowicz et al., 2015; Greff et al., 2017).

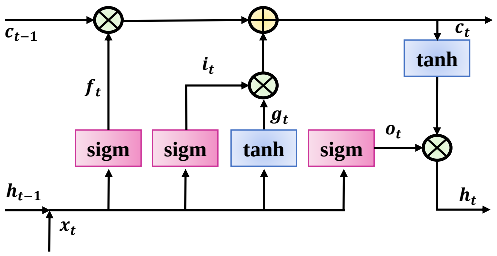


Figure 4. LSTM cell structure

The structure of an LSTM cell is shown in Figure 4 and equations for the LSTM mechanism are provided in Eq. (1)–(6).

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2}$$
$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{3}$$
$$g_t = tanh(W_g * [h_{t-1}, x_t] + b_g) \tag{4}$$
$$c_t = (c_{t-1} \otimes f_t) \oplus (g_t \otimes i_t) \tag{5}$$
$$h_t = tanh(c_t) \otimes o_t \tag{6}$$

The line across the top is the path that the past cell state c_{t-1} moves to and is transformed to the current cell state c_t. The cell state plays an important role because it adaptively stores the information from previous sequences (internal memory). The line across the bottom is the path along which the hidden state of the previous LSTM cell h_{t-1} and the current input x_t move. h_{t-1} and x_t are used to compute the forget gate f, input gate i, and output gate o, which are collectively used to compute the current cell state c_t and the current hidden state h_t. The forget gate f_t defines the amount of elements that can pass through the previous state h_{t-1}. The input gate i_t defines the state value computed for x_t at the current input, and the output gate o_t defines by how much the internal state is propagated to the next sequence. The internal hidden state g_t is computed based on the current input x_t and the previously hidden state h_{t-1}. The final hidden state h_t is the element-wise product of the current cell state c_t and the output state o_t.

The RNN structure for long and free text-based KDA is illustrated in Figure 5 and the specific network structure parameters are listed in Table 2. We used the concatenated one-hot vectors of two key identifiers as the input of the RNN model and used four digraph-based elapsed typing times (DD-time, UD-time, UU-time, and DU-time) as the targets of the RNN model. For the example in Figure 5, the first input sequence is a 64-dimensional vector concatenating two 32-dimensional one-hot vectors (“k” and “e”) and the corresponding outputs are four keystroke times, i.e DD-time, UD-time, UU-time, and DU-time, for the digraph “ke”. The proposed RNN–KDA model was trained for each user based on his/her own keystroke data only. The purpose of this model is to correctly predict the four keystroke times of the targeted user when two consecutive key sequences are continuously provided. If the model is trained well, it should sufficiently learn a valid user’s typing behavior so that it can predict the typing times accurately for any two given key sequences. As it is trained to learn the typing style of a specific user, the predicted key typing times for other users are not as accurate as those of the targeted user. Therefore, we can use the difference between the actual typing times and the predicted typing times determined by the RNN–KDA model as the novelty score.

We used two hidden layers for the RNN–KDA model, whose individual nodes are the LSTMs, and used two other fully-connected hidden layers between the second RNN layer and the output layer. The Huber loss (Huber et al., 1964), as given in Eq. (7), with $\delta = 1$ was used for the network loss function:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{7}$$

Finally, the NS(novelty score) is computed by Eq. (8):

$$\begin{aligned} \text{NS} = \frac{1}{4T} \sum_{t=1}^{T} \{ & L_\delta(DD_t, \widehat{DD}_t) + L_\delta(DU_t, \widehat{DU}_t) \\ & + L_\delta(UU_t, \widehat{UU}_t) + L_\delta(UD_t, \widehat{UD}_t) \}, \end{aligned} \tag{8}$$

where DD_t and $\widehat{DD}_t$ are the actual DD-time and predicted DD-time for the t-th digraph, respectively. Other symbols are defined in a similar manner.

Detailed training strategies are as follows. During training, we used batch normalization (Ioffe & Szegedy, 2015) and used the rectified linear unit (Krizhevsky et al., 2012) activation function for fully connected layers. The size of mini-batches was set to 16, and the RMSProp (Hinton et al., 2012) was used to optimize the network weights. The initial learning rate was set to 0.1, and the loss was stored every 10 iterations. When the median of the most recent 20 stored losses was greater than the median of the most recent five stored losses, or 150 iterations were per-

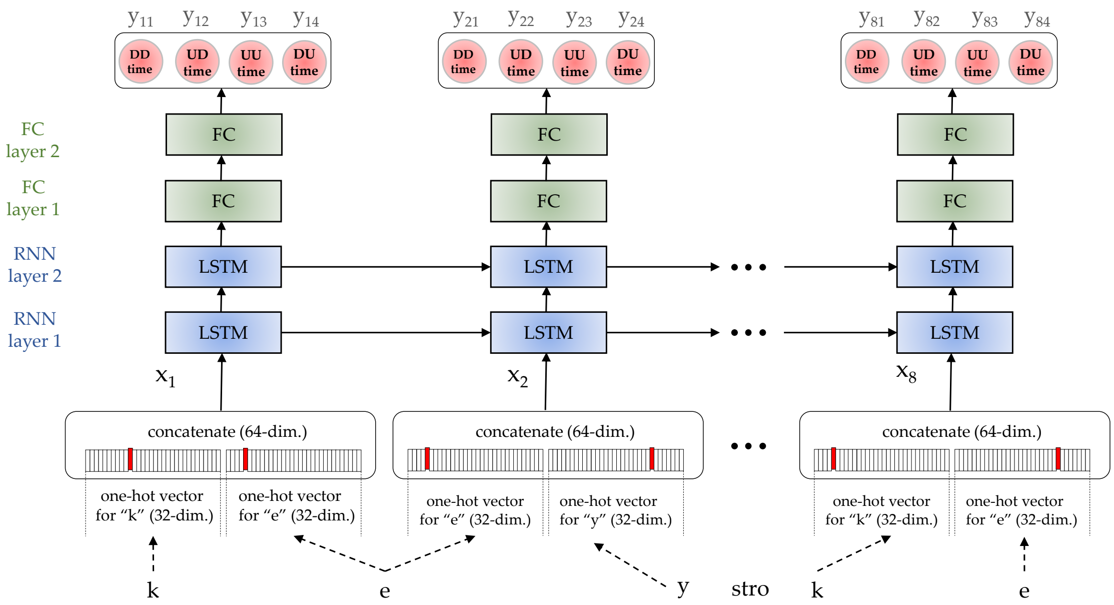


Figure 5. Structure of the proposed RNN-based KDA model with an example of a "keystroke" string

Table 2. Proposed algorithm parameter details

No. of LSTM cells in RNN layer 1	No. of LSTM cells in RNN layer 2	No. of hidden nodes in FC layer 1	No. of hidden nodes in FC layer 2	Batch size
128	128	128	128	16

formed with the same learning rate, the learning rate was halved. This decreasing learning rate process was repeated eight times.

5. Experiment Settings

5.1. Benchmark Methods

5.1.1. Kolmogorov–Smirnov Statistic and Cramér–von Mises Criterion

The simplest KDA method based on long and freely typed text is to compare the keystroke timing distributions between two users; if the two distributions are close enough, then the two sets are considered to be typed by the same user, otherwise they are considered to be typed by different users. Although many statistical tests assume a normal distribution, actual keystroke data rarely satisfy this assumption. Hence, the K–S statistic and C–M criterion (Kang & Cho, 2015), which do not assume any parametric distributions, were used to compare two empirical distributions of keystroke times between two users. Let K_{trn} and K_{tst} denote the keystroke datasets for training and test, respectively. N_{trn} and N_{tst} denote the number of keystrokes in the training and test datasets, respectively. Then, the cumulative distribution for the training and test datasets, i.e. C_{trn} and C_{tst}, are defined as follows:

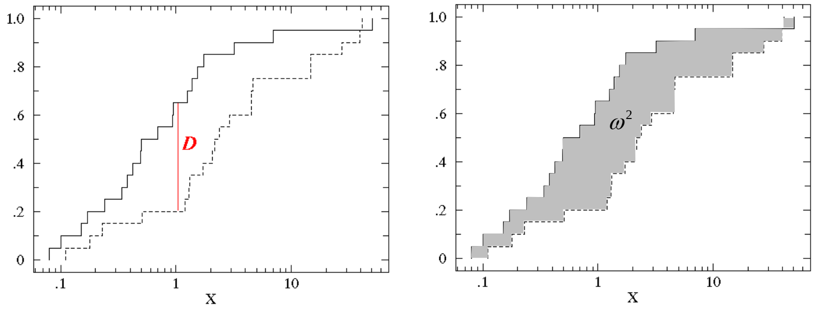
$$C_{trn}^{di}(\alpha) = \frac{1}{N_{trn}^{di}} \sum_{i=1}^{N_{trn}^{di}} I(X_i^{di} \leq \alpha), \quad di \subset \{DD, DU, UD, UU\}, \tag{9}$$

$$C_{tst}^{di}(\alpha) = \frac{1}{N_{tst}^{di}} \sum_{i=1}^{N_{tst}^{di}} I(X_i^{di} \leq \alpha), \quad di \subset \{DD, DU, UD, UU\}, \tag{10}$$

where X_i is the individual keystroke time of the i^{th} digraph. The I function is an indicator function which returns 1 if the condition inside the parenthesis is satisfied, and returns 0 otherwise. The K-S statistic is computed by Eq. (11):

$$D_{K_{trn},K_{tst}}^{di} = \sup \left| C_{trn}^{di} - C_{tst}^{di} \right|, \quad di \subset \{DD, DU, UD, UU\}. \tag{11}$$

Figure 6(a) is a graphical representation of the K-S statistic, which can be interpreted as the maximum difference of two cumulative density functions, C_{trn} and C_{tst}. Hence, the more similar the typing behaviors between two users are, the smaller the K-S values. The



(a) Kolmogorov–Smirnov statistic (b) Cramér–von Mises criterion

Figure 6. Example of Kolmogorov–Smirnov statistic and Cramér–von Mises criterion (reprinted from (Kang & Cho, 2015))

C-M criterion is computed using Eq. (12):

$$\begin{aligned} \text{C–M}^{di}_{K_{trn},K_{tst}} &= \omega^2_{di} \\ &= \int_{-\infty}^{\infty} [C^{di}_{trn}(\alpha) - C^{di}_{tst}(\alpha)]^2 d\alpha, \\ di &\subset \{DD, DU, UD, UU\}\,. \end{aligned} \tag{12}$$

Figure 6(b) is an illustrative example of the C-M criterion with the same two empirical distributions as in Figure 6(a). The main difference between the K-S statistic and C-M criterion is that the K-S statistic uses the maximum difference between the two empirical distributions to determine the degree of difference between two distributions, whereas the C-M criterion uses the cumulative integration of the differences between two distributions. Four K-S statistic values and four C-M criterion values can be computed according to keystroke times, so we used their geometric mean as the final result of applying the K-S statistic and C-M criterion:

$$\begin{aligned} \text{K–S}_{K_{trn},K_{tst}} &= \sqrt[4]{D^{DD}_{K_{trn},K_{tst}} \times D^{DU}_{K_{trn},K_{tst}}} \\ &\times \sqrt[4]{D^{UD}_{K_{trn},K_{tst}} \times D^{UU}_{K_{trn},K_{tst}}}, \end{aligned} \tag{13}$$

$$\text{C–M}_{K_{trn},K_{tst}} = \sqrt[4]{\omega^2_{DD} \times \omega^2_{DU} \times \omega^2_{UD} \times \omega^2_{UU}} \tag{14}$$

5.1.2. R measure, A measure, R+A measure, and RA measure

R and A measures and their variations (Gunetti & Picardi, 2005) compute the similarity of typing speeds for common digraphs between two keystroke datasets to determine whether these datasets are produced by the same user. The main difference between the R measure and A measure is that while the R measure mainly focuses on comparing the rank of typing speeds by computing the 'degree of disorder', the A measure focuses more on comparing the absolute typing speed by computing the 'ratio of average typing time'.

Figure 7 shows an illustrative example of computing the R and A measures. The keystroke set E1 is collected while typing the word 'authentication' and E2 is collected while typing the word 'theoretical' as shown in Figure 7(a). The numbers between characters are the elapsed time in terms of milliseconds from when the beginning and the final keys are pressed, which can be understood as the cumulative DD-time. There are four common digraphs between E1 and E2: 'ie', 'he', 'th', and 'ca'. To compute the R measure, the average DD-times of common digraphs are sorted in ascending order as shown in Figure 7(b). Then, the degree of disorder, i.e. sum of the differences between the ranks of common digraphs, is computed. In this example, the computation yields $2+0+2+3+1=8$. When the size of common digraphs is n, the maximum degree of disorder is computed as follows:

$$\text{Max. degree of disorder} = \begin{cases} \frac{1}{2}n^2 & \text{if } n \text{ is even,} \\ \frac{1}{2}(n^2-1) & \text{if } n \text{ is odd.} \end{cases} \tag{15}$$

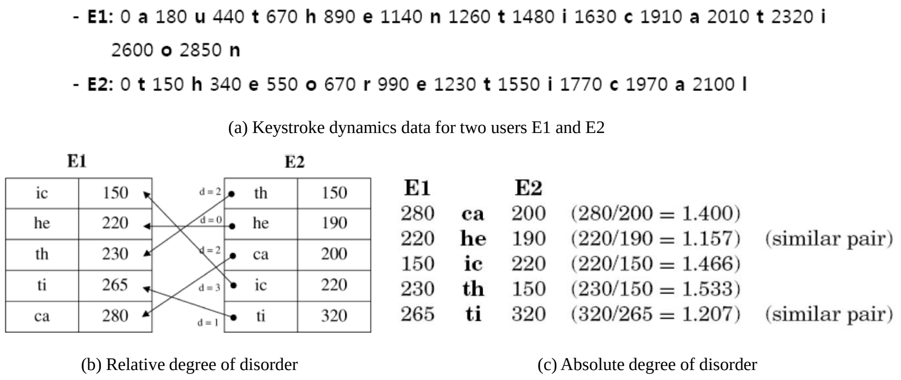
Finally, the R measure is computed by dividing the degree of disorder as follows.

$$\text{R measure} = 1 - \frac{\text{Sum of degree of disorder}}{\text{Max. degree of disorder}} \tag{16}$$

The process of computing the A measure is as follows. First, the ratio between two DD-times is computed for the i^{th} digraph using Eq. (17):

$$\text{Digraph ratio}_i = \frac{\max(\text{DD-time}^{E1}_i, \text{DD-time}^{E2}_i)}{\min(\text{DD-time}^{E1}_i, \text{DD-time}^{E2}_i)} \tag{17}$$

For example, the Digraph ratio for the first common digraph, 'ca', in Figure 7(c) is $\frac{280}{200} = 1.400$. Then, the proportion of digraphs whose digraph ratio is lower than a predefined thresh-



- **E1:** 0 **a** 180 **u** 440 **t** 670 **h** 890 **e** 1140 **n** 1260 **t** 1480 **i** 1630 **c** 1910 **a** 2010 **t** 2320 **i** 2600 **o** 2850 **n**
- **E2:** 0 **t** 150 **h** 340 **e** 550 **o** 670 **r** 990 **e** 1230 **t** 1550 **i** 1770 **c** 1970 **a** 2100 **l**

(a) Keystroke dynamics data for two users E1 and E2

E1	
ic	150
he	220
th	230
ti	265
ca	280

E2	
th	150
he	190
ca	200
ic	220
ti	320

(b) Relative degree of disorder

E1		E2		
280	**ca**	200	(280/200 = 1.400)	
220	**he**	190	(220/190 = 1.157)	(similar pair)
150	**ic**	220	(220/150 = 1.466)	
230	**th**	150	(230/150 = 1.533)	
265	**ti**	320	(320/265 = 1.207)	(similar pair)

(c) Absolute degree of disorder

Figure 7. Example of R measure and A measure (reprinted from(Kang & Cho, 2015))

old θ to the total number of common digraphs is used as the A measure, as shown in Eq. (18):

$$\text{A measure} = \frac{1}{n}\sum_{i=1}^{n} I(\text{Digraph ratio}_i \leq \theta). \tag{18}$$

In the example in Figure 7, θ is set to 1.4 so that the number of common digraphs with digraph ratios lower than 1.4 is two, which results in an A measure of 0.4 (2/5).

Next, the R + A measure combines the R and A measures by adding them, while the RA measure combines the R and A measures by multiplying them, per Eq. (19) and Eq. (20), respectively.

$$\text{R+A measure} = \text{R measure} + \text{A measure}, \tag{19}$$

$$\text{RA measure} = \text{R measure} \times \text{A measure}. \tag{20}$$

Although the R, A, R+A, and RA measures were discussed based on DD-time, they can be applied to other digraph-based key times such as DU, UD, and UU. In our experiment, R, A, R+A, and RA measures were computed for the four digraph-based key times and we aggregated them by taking their geometric mean as in the K-S statistic and C-M criterion.

5.2. Data Partitioning

One of the most important requirements for the KDA system is the authentication speed. Assuming the same authentication accuracy or error rate, the fewer the test keystrokes, the better the KDA model. In other words, the KDA model should determine whether the current keystroke data is provided by a valid user as early as possible. In practice, it is easy to continuously collect a valid user's keystroke data once the system has been installed. This implies that it is possible to use a large set of keystroke data to train the authentication model, but the test process should be conducted as quickly as possible, based only on a few keystrokes. Based on these practical requirements, we used a total of 18 combinations of training and keystroke set sizes to investigate the authentication performance by combining six different training set sizes (N_{trn}), i.e. 500, 1,000, 3,000, 5,000, 7,000, and 10,000, with three test set sizes (N_{tst}), i.e., 10, 50, and 100.

A total of 120 participants were enrolled in our study. Based on the keystroke dataset provided by each user, we constructed the training dataset and test dataset for the proposed RNN-KDA model and benchmark model as shown in Figure 8. First, the entire dataset of each participant is divided into two sets: a training dataset from the first to the N_{trn}-th keystroke and a test dataset from the $(N + 1)$-th to the final keystroke. For K–S, C–M, R, A, R+A, and RA measures, the training dataset of the valid user is used as the reference dataset. For the RNN–KDA model, however, because we set the batch size to 16, we randomly sampled 16 sets of 100 consecutive keystrokes in each batch. Hence, it is possible for two sets of training keystroke sets to overlap. For the performance evaluation, a total of 357 keystroke subsets with length of N_{tst} are sampled from the test data of the valid user. As in the training dataset for RNN-KDA model, each subset represents consecutive digraphs and it is possible for two subsets to overlap. As the impostor's keystroke dataset, three keystroke subsets with size of N_{tst} are randomly sampled from all users except the valid user. The only difference between the benchmark methods and the proposed RNN–KDA model is the manner in which the training dataset of the valid user is constructed. The K–S, C–M, R, A, RA, and R + A used from the first to the N keystroke in the valid user's keystroke dataset as the training dataset,

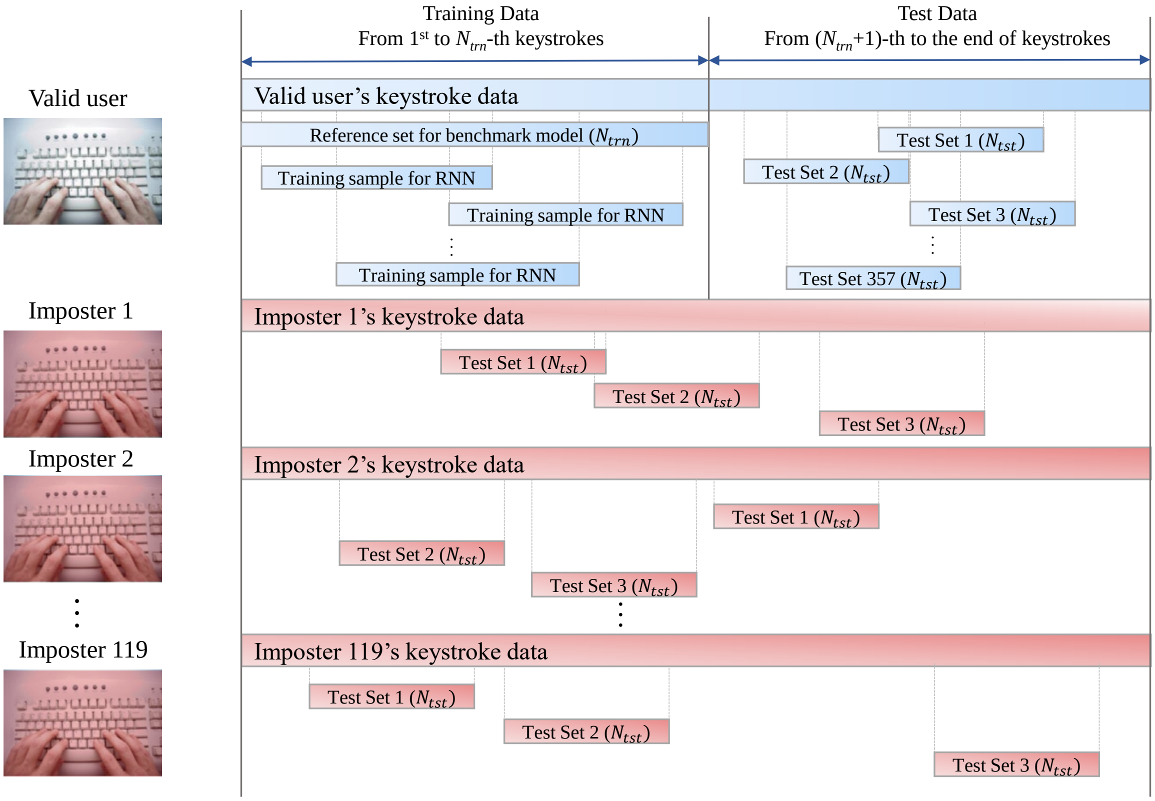


Figure 8. Sampling keystroke data from a valid user and from impostors

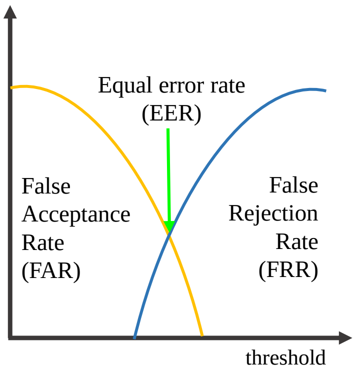


Figure 9. FAR and FRR changes based on threshold changes and EER definition

whereas the RNN–KDA used sampled keystroke sets from the valid user's training keystroke dataset.

5.3. Performance Measure

EER is used as a performance measure in our study. Two types of errors occur in security systems: FRR and FAR. FRR is the proportion of a valid user's falsely rejected trials while FAR is the proportion of an impostor's falsely accepted trials. There is a trade-off between FRR and FAR as shown in Figure 9. When the threshold of the authentication is tightly set, a low FAR can be achieved at the cost of FRR. On the other hand, a high FAR with a low FRR can be achieved when the threshold is loosely set. Hence, to evaluate the fundamental authentication ability, a threshold independent performance measure is required. EER is used as an alternative and calculates the error rate at the point where the FRR and FAR are the same. In this study, EER was computed based on the novelty scores obtained by 714 test keystroke sets (357 from a valid user and 357 from impostors).

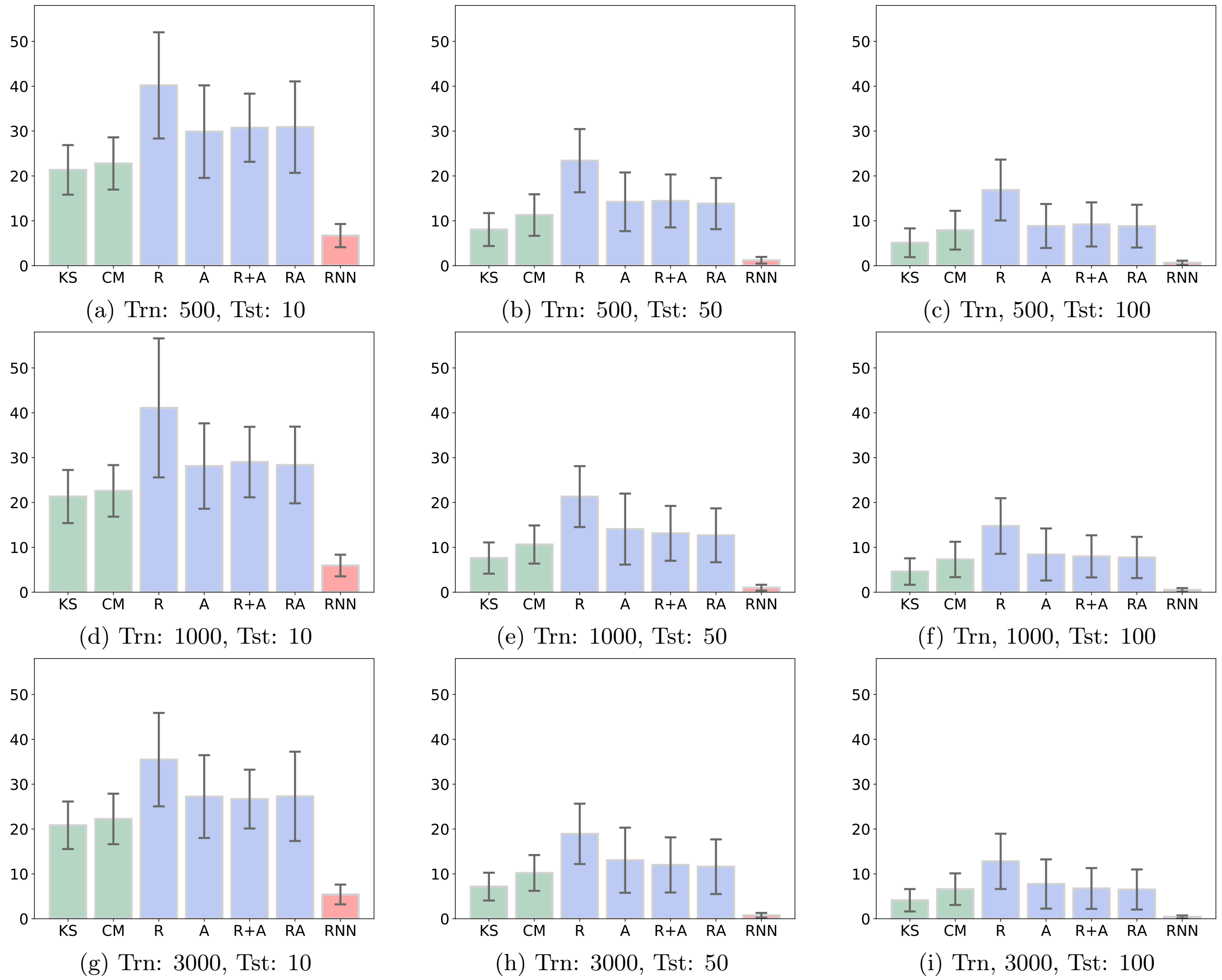
6. Experimental Results

Table 3 lists the average and standard deviation of EER for each authentication method using different combinations of training and test keystrokes. As the K–S statistics yielded the best performance in all cases among the benchmark methods, a statistical hypothesis test was conducted between the proposed RNN–KDA and K–S statistics. Based on the experimental results, we make the following observations. First, the

Table 3. Average EERs for K–S, C–M, R, A, R+A, and RNN (numbers in parentheses are the standard deviation; asterisks indicate that the average EER of RNN–KDA is lower than that of K–S statistic at a significance level of 0.001)

No. Trn.	No. Tst.	K–S	C–M	R	A	R+A	RA	RNN	Improvement
500	10	21.35 (5.53)	22.78 (5.82)	40.19 (11.84)	29.88 (10.32)	30.75 (7.59)	30.88 (10.20)	**6.70*** **(2.59)**	68.62%
	50	8.05 (3.67)	11.28 (4.63)	23.41 (7.05)	14.23 (6.54)	14.43 (5.90)	13.84 (5.70)	**1.20*** **(0.74)**	85.09%
	100	5.10 (3.21)	7.90 (4.32)	16.86 (6.78)	8.84 (4.90)	9.19 (4.92)	8.80 (4.77)	**0.63*** **(0.48)**	87.64%
1,000	10	21.33 (5.92)	22.59 (5.75)	41.09 (15.49)	28.13 (9.52)	29.01 (7.86)	28.36 (8.54)	**5.95*** **(2.41)**	72.10%
	50	7.62 (3.48)	10.63 (4.25)	21.31 (6.79)	14.07 (7.91)	13.12 (6.12)	12.69 (6.00)	**1.00*** **(0.66)**	86.87%
	100	4.60 (2.95)	7.30 (3.95)	14.76 (6.20)	8.41 (5.80)	7.99 (4.70)	7.74 (4.60)	**0.51*** **(0.41)**	88.91%
3,000	10	20.84 (5.30)	22.25 (5.63)	35.47 (10.42)	27.24 (9.23)	26.68 (6.55)	27.29 (9.95)	**5.41*** **(2.20)**	74.04%
	50	7.16 (3.10)	10.21 (3.98)	18.92 (6.73)	13.05 (7.26)	12.00 (6.15)	11.60 (6.09)	**0.77*** **(0.53)**	89.25%
	100	4.12 (2.49)	6.59 (3.53)	12.79 (6.16)	7.75 (5.49)	6.75 (4.55)	6.52 (4.47)	**0.42*** **(0.31)**	89.81%
5,000	10	20.87 (5.10)	22.30 (5.56)	36.65 (11.35)	27.58 (11.53)	25.78 (6.67)	25.71 (8.13)	**5.33*** **(2.23)**	74.46%
	50	7.18 (3.17)	10.21 (3.97)	17.83 (6.55)	13.18 (8.13)	11.20 (6.00)	11.02 (6.06)	**0.76*** **(0.65)**	89.42%
	100	4.05 (2.53)	6.54 (3.46)	11.84 (5.81)	7.73 (6.08)	6.36 (4.78)	6.19 (4.74)	**0.42*** **(0.40)**	89.53%
7,000	10	20.89 (4.95)	22.25 (5.39)	35.34 (10.91)	27.55 (11.23)	25.94 (6.56)	25.32 (7.90)	**5.31*** **(2.00)**	74.58%
	50	7.12 (3.15)	10.17 (3.96)	17.45 (6.51)	12.93 (7.67)	11.04 (6.07)	10.78 (6.08)	**0.67*** **(0.54)**	90.59%
	100	4.00 (2.52)	6.41 (3.31)	11.46 (5.72)	7.87 (6.40)	6.24 (4.67)	6.09 (4.60)	**0.38*** **(0.43)**	90.50%
10,000	10	21.03 (5.04)	22.48 (5.53)	36.73 (12.48)	26.50 (9.80)	26.02 (7.27)	27.31 (11.54)	**5.17*** **(1.95)**	75.42%
	50	7.18 (3.28)	10.11 (4.08)	17.81 (6.67)	12.98 (7.60)	11.17 (6.04)	10.93 (6.04)	**0.63*** **(0.50)**	91.23%
	100	4.10 (2.85)	6.53 (3.83)	11.73 (5.95)	7.96 (6.37)	6.26 (4.74)	6.17 (4.79)	**0.36*** **(0.32)**	91.22%

EER of all models decreases when either higher numbers of training or test keystrokes are provided. Second, the proposed RNN–KDA yielded the lowest EER for all combinations of training and test keystrokes, followed by the K–S statistic. Because the number of test keystrokes were fewer than those of the training keystrokes in our experiment, R and A measures and their variations resulted in higher EER than other methods, mainly because of insufficient common digraphs available to compare the typing behaviors of two keystroke sets. Compared with the second best model, i.e. K–S statistic, the EER ratio of RNN–KDA decreases by at least 68.62% and up to 91.23%, and these improvements are strongly supported at a significance level of 0.001. Third, the RNN–KDA can be practically applied, because its EER did not exceed 7% in the most difficult case (500 training keystrokes and 10 test keystrokes) while the EERs of other methods exceeded 20%. In practice, 10 test keystrokes are obtained when a user types two or three words. If the system stores only 500 previous keystrokes of the valid user, the RNN–KDA is able to achieve 93.3% detection accuracy after only two or three words are typed. This detection ability improves up to 94.83% when 10,000 training keystrokes are available. When the length of test keystrokes increases, the detection



(a) Trn: 500, Tst: 10

(b) Trn: 500, Tst: 50

(c) Trn, 500, Tst: 100

(d) Trn: 1000, Tst: 10

(e) Trn: 1000, Tst: 50

(f) Trn, 1000, Tst: 100

(g) Trn: 3000, Tst: 10

(h) Trn: 3000, Tst: 50

(i) Trn, 3000, Tst: 100

Figure 10. Average EER and standard deviation of each method with short to moderate training keystrokes (y-axis: EER, error bars represent the standard deviation)

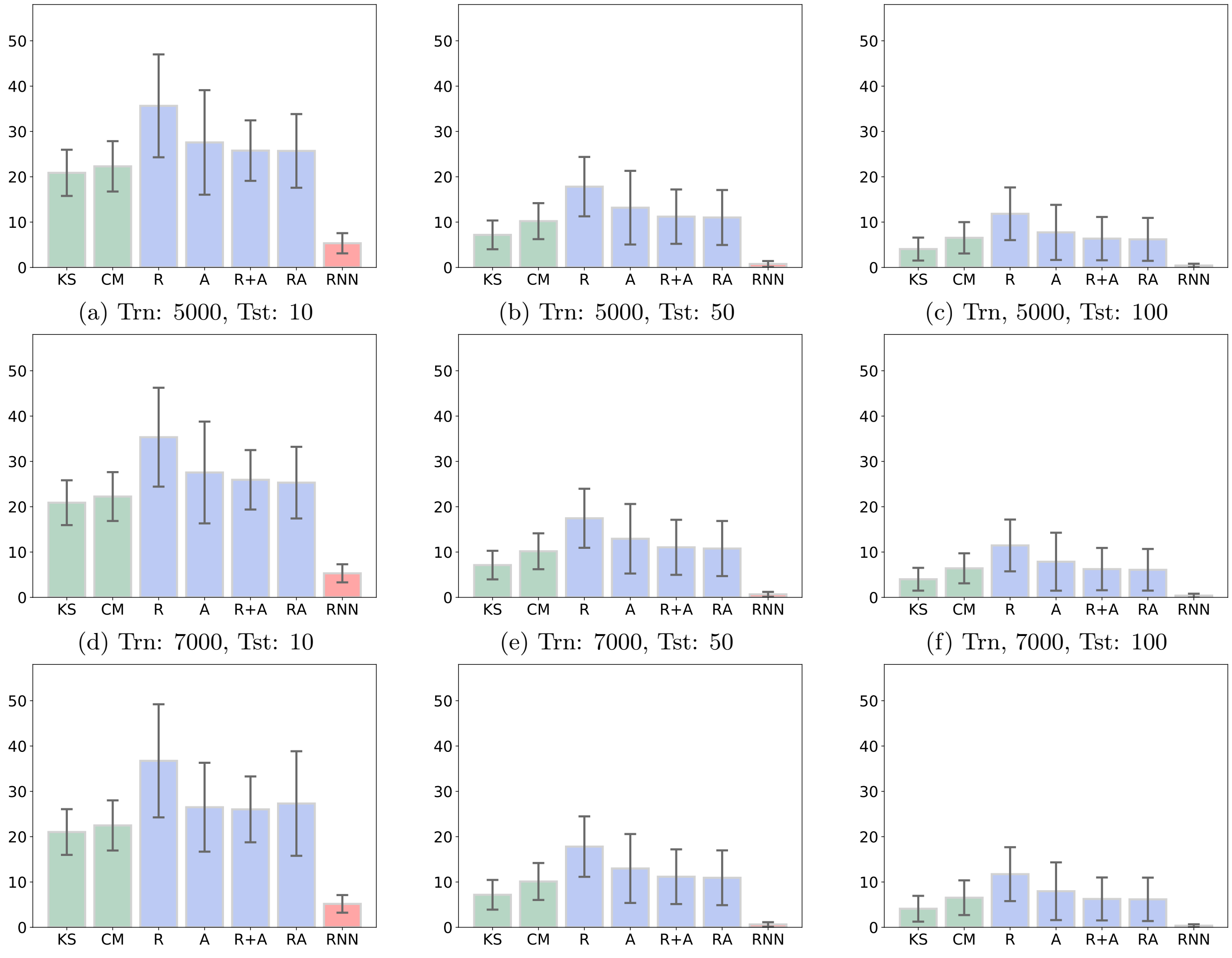
performance significantly improves as well. With 500 training keystrokes, the EER was only 0.63% with 100 test keystrokes, and decreased to 0.36% with 10,000 training keystrokes. In practice, it is relatively easy to collect a large set of training data because a user's typing data can be obtained without disturbing the user. The most significant issue for long and freely typed text-based KDA systems is how early they can identify whether the current keystroke data is being generated by a valid user or not.

Figure 10 and 11 show the EERs according to different lengths of training and test keystrokes, respectively. As demonstrated by the two figures, EER decreases when either the length of training or test keystrokes increases. Between them, the length of test keystrokes is more influential than training keystrokes. Again, when one set of keystrokes (the test set in our experiment) is significantly less than the other set, K–S statistic and C–M criterion, which do not consider the key sequence information, resulted in lower EERs than R and A measures, which do consider the key sequence information. Note that the R measure had difficulty generalizing the valid user's typing behavior because the relative typing speeds of common digraphs were less stable than for the A-measure due to the small set of common digraphs. Therefore, combining the R measure and A measure did not produce any synergy, especially when the test keystroke length is very short.

As the K–S statistic was the second-best authentication model in our experiment, we compute the improvement ratio of the RNN–KDA using Eq. (21) over the K–S statistic, as shown in Figure 12.

$$\text{Improvement ratio} = \frac{\text{EER of K-S statistic}}{\text{EER of RNN–KDA}}. \tag{21}$$

The higher the improvement ratio, the more the EER



(a) Trn: 5000, Tst: 10

(b) Trn: 5000, Tst: 50

(c) Trn, 5000, Tst: 100

(d) Trn: 7000, Tst: 10

(e) Trn: 7000, Tst: 50

(f) Trn, 7000, Tst: 100

(g) Trn: 10000, Tst: 10

(h) Trn: 10000, Tst: 50

(i) Trn, 10000, Tst: 100

Figure 11. Average EER and standard deviation of each method with moderate to long training keystrokes (y-axis: EER, error bars represent the standard deviation)

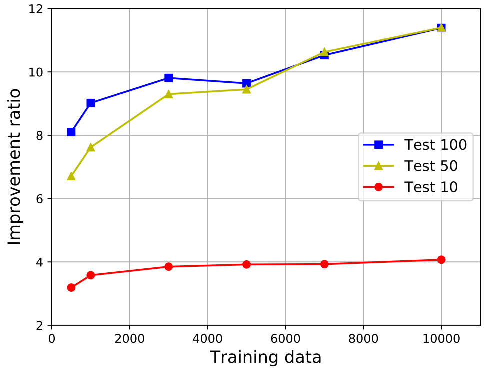


Figure 12. Improvement ratio of RNN–KDA compared to K–S statistic as training data increases

is reduced by the RNN–KDA compared to the K–S statistic. An interesting observation is that, as shown in Table 3, Figure 10, and Figure 11, although the EERs of both methods decrease if either the length of training or test keystrokes increases, the magnitude of improvement is higher for RNN–KDA than for the K–S statistic. For a fixed training dataset, the improvement ratio is the highest for 100 test keystrokes and the lowest for 10 keystrokes. In addition, for a fixed test keystroke length, the improvement ratio in Figure 12 generally increases. The performance improvement is more significant with longer test keystrokes such that the slope of 100 test keystrokes is greater than for 10 test keystrokes. In actual circumstances, a valid user's keystroke data can be collected continuously. Thus, it can be expected that as more valid user's keystroke data become available, RNN–KDA can consistently improve authentication performance compared with other traditional KDA methods.

7. Conclusion

In this study, we proposed an RNN-based KDA model to enhance user authentication performance based on freely typed keystroke data. Our proposed model takes the concatenated one-hot vectors of digraphs as input and attempts to predict the four keystroke timing values, i.e. DD–time, UD–time, UU–time, and DU–time. Based on experiments with 120 participants and six benchmark methods, the proposed RNN–KDA model yielded the best authentication performance in terms of EER. The RNN–KDA model yielded the lowest EER for all combinations of training and test keystrokes, followed by the K-S statistic. With 500 training keystrokes, the EER is only 0.63% with 100 test keystrokes and decreased to 6.70% with 10 test keystrokes. With 10,000 training keystrokes, the EER was 0.36% with 100 test keystrokes and 5.17% with 10 test keystrokes. Compared with the K-S statistic, the EER decreasing ratio for the RNN–KDA model was at least 68.62% and was as high as 91.23%, and these improvements are strongly supported at a significance level of 0.001. In addition, the RNN–KDA model exhibits a more rapid performance improvement trend than the benchmark methods when the number of training and test keystroke data increase.

Despite the favorable experimental results, there are some limitations to the current study, which imply future research directions. First, the RNN–KDA model was validated based only on keystroke data collected from PC keyboards. As many input devices other than PC keyboards are commonly used, such as soft keyboards in smartphones, the RNN–KDA model should be validated with keystroke data collected from various input devices. Second, the language used in our experiment was Korean. To ensure language independence in the proposed RNN–KDA model, it should be verified based on different languages with different alphabetical systems.

References

Alpar, Orcan. Frequency spectrograms for biometric keystroke authentication using neural network based classifier. *Knowledge-Based Systems*, 116:163–171, 2017.

Alsultan, Arwa, Warwick, Kevin, and Wei, Hong. Non-conventional keystroke dynamics for user authentication. *Pattern Recognition Letters*, 89:53–59, 2017.

Bakelman, Ned, Monaco, John V, Cha, Sung-Hyuk, and Tappert, Charles C. Continual keystroke biometric authentication on short bursts of keyboard input. *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, 2012.

Banerjee, Salil P and Woodard, Damon L. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

Bhatt, Shanthi and Santhanam, T. Keystroke dynamics for biometric authentication—a survey. In *Pattern Recognition, Informatics and Mobile Engineering (PRIME)*, pp. 17–23. IEEE, 2013.

Biddle, Robert, Mannan, Mohammad, van Oorschot, Paul C, and Whalen, Tara. User study, analysis, and usable security of passwords based on digital objects. *IEEE Transactions on Information Forensics and Security*, 6(3):970–979, 2011.

Bleha, Saleh, Slivinsky, Charles, and Hussien, Bassam. Computer-access security systems using keystroke dynamics. *IEEE Transactions on pattern analysis and machine intelligence*, 12(12):1217–1222, 1990.

Bours, Patrick. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Information Security Technical Report*, 17(1):36–43, 2012.

Buch, Tarjani, Cotoranu, Andreea, Jeskey, Eric, Tihon, Florin, and Villani, Mary. An enhanced keystroke biometric system and associated studies. *Proc. CSIS Research Day, Pace Univ*, pp. C4.2–C4.7, 2008.

Chang, Ting-Yi. Dynamically generate a long-lived private key based on password keystroke features and neural network. *Information Sciences*, 211:36–47, 2012.

Chantan, Charoon, Sinthupinyo, Sukree, and Rungkasiri, Tippakorn. Improving accuracy of authentication process via short free text using bayesian network. *International Journal of Computer Science Issues*, 9(3), 2012.

Clarke, Nathan and Furnell, Steven. Biometrics–the promise versus the practice. *Computer Fraud & Security*, 2005(9):12–16, 2005.

Connolly, Jean-François, Granger, Eric, and Sabourin, Robert. An adaptive classification system for video-based face recognition. *Information Sciences*, 192: 50–70, 2012.

Curtin, Mary, Tappert, Charles, Villani, Mary, Ngo, Giang, Simone, Justin, Fort, Huguens St, and Cha, S. Keystroke biometric recognition on long-text input: A feasibility study. *Proc. Int. MultiConf. Engineers & Computer Scientists (IMECS)*, 2006.

Davoudi, H and Kabir, E. User authentication based on free text keystroke patterns. In *3rd Joint Congress on Fuzzy and Intelligent Systems*, 2010a.

Davoudi, Homa and Kabir, Ehsanollah. A new distance measure for free text keystroke authentication. In *Computer Conference, 2009. CSICC 2009. 14th International CSI*, pp. 570–575. IEEE, 2009.

Davoudi, Homa and Kabir, Ehsanollah. Modification of the relative distance for free text keystroke authentication. In *Telecommunications (IST), 2010 5th International Symposium on*, pp. 547–551. IEEE, 2010b.

Dowland, P, Singh, H, and Furnell, S. A preliminary investigation of user authentication using continuous keystroke analysis. In *Proceedings of the International Conference on Information Security Management and Small Systems Security*, pp. 215–226, 2001.

Fairhurst, Michael and Da Costa-Abreu, Márjory. Using keystroke dynamics for gender identification in social network environment. *4th International Conference on Imaging for Crime Detection and Prevention*, pp. 1–6, 2011.

Farooq, Muhammad Umar, Waseem, Muhammad, Khairi, Anjum, and Mazhar, Sadia. A critical analysis on the security concerns of internet of things (iot). *International Journal of Computer Applications*, 111(7):1–6, 2015.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Graves, Alex, Mohamed, Abdel Rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

Greff, Klaus, Srivastava, Rupesh K, Koutník, Jan, Steunebrink, Bas R, and Schmidhuber, Jürgen. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.

Gunetti, Daniele and Picardi, Claudia. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3):312–347, 2005.

Gunetti, Daniele and Ruffo, Giancarlo. Intrusion detection through behavioral data. In *In International Symposium on Intelligent Data Analysis*, volume 99, pp. 383–394. Springer, 1999.

Hempstalk, Kathryn, Frank, Eibe, and Witten, Ian H. One-class classification by combining density and class probability estimation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 505–519. Springer, 2008.

Hiltgen, Alain, Kramp, Thorsten, and Weigold, Thomas. Secure internet banking authentication. *IEEE Security & Privacy*, 4(2):21–29, 2006.

Hinton, G, Srivastava, N, and Swersky, K. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*, 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hochreiter, Sepp, Bengio, Yoshua, Frasconi, Paolo, Schmidhuber, Jürgen, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Hosseinzadeh, Danoush and Krishnan, Sridhar. Gaussian mixture modeling of keystroke patterns for biometric applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(6):816–826, 2008.

Hu, Jiankun, Gingrich, Don, and Sentosa, Andy. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. In *Communications, 2008. ICC'08. IEEE International Conference on*, pp. 1556–1560. IEEE, 2008.

Huber, Peter J et al. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Janakiraman, Rajkumar and Sim, Terence. Keystroke dynamics in a general setting. *Advances in Biometrics*, pp. 584–593, 2007.

Joyce, Rick and Gupta, Gopal. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.

Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2342–2350, 2015.

Kang, Pilsung. The effects of different alphabets on free text keystroke authentication: A case study on the korean–english users. *Journal of Systems and Software*, 102:1–11, 2015.

Kang, Pilsung and Cho, Sungzoon. A hybrid novelty score and its use in keystroke dynamics-based user authentication. *Pattern recognition*, 42(11):3115–3127, 2009.

Kang, Pilsung and Cho, Sungzoon. Keystroke dynamics-based user authentication using long and free text strings from various input devices. *Information Sciences*, 308:72–93, 2015.

Kang, Pilsung, Park, Sunghoon, Hwang, Seongseob, Lee, Hyoungjoo, and Cho, Sungzoon. Improvement of keystroke data quality through artificial rhythms and cues. *computers & security*, 27(1):3–11, 2008.

Karnan, Marcus, Akila, Muthuramalingam, and Krishnaraj, Nishara. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2):1565–1573, 2011.

Karpathy, Andrej, Johnson, Justin, and Fei-Fei, Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Kim, Junhong, Kim, Haedong, and Kang, Pilsung. Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection. *Applied Soft Computing*, 62:1077–1087, 2018.

Kołakowska, Agata. A review of emotion recognition methods based on keystroke dynamics and mouse movements. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, pp. 548–555. IEEE, 2013.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Liu, Pengfei, Qiu, Xipeng, and Huang, Xuanjing. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

Lou, Wenjing and Ren, Kui. Security, privacy, and accountability in wireless access networks. *IEEE Wireless Communications*, 16(4):80–87, 2009.

Ma, Li, Tan, Tieniu, Wang, Yunhong, and Zhang, Dexin. Personal identification based on iris texture analysis. *IEEE transactions on pattern analysis and machine intelligence*, 25(12):1519–1533, 2003.

Messerman, Arik, Mustafić, Tarik, Camtepe, Seyit Ahmet, and Albayrak, Sahin. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *Biometrics (IJCB), 2011 International Joint Conference on*, pp. 1–8. IEEE, 2011.

Monrose, Fabian and Rubin, Aviel. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM conference on Computer and communications security*, pp. 48–56. ACM, 1997.

Monrose, Fabian, Reiter, Michael K, and Wetzel, Susanne. Password hardening based on keystroke dynamics. *International journal of Information security*, 1(2):69–83, 2002.

Montalvão Filho, Jugurta R and Freire, Eduardo O. On the equalization of keystroke timing histograms. *Pattern Recognition Letters*, 27(13):1440–1446, 2006.

Nahin, AFM Nazmul Haque, Alam, Jawad Mohammad, Mahmud, Hasan, and Hasan, Kamrul. Identifying emotion by keystroke dynamics and text pattern analysis. *Behaviour & Information Technology*, 33(9):987–996, 2014.

Ngugi, Benjamin, Kahn, Beverly K, and Tremaine, Marilyn. Typing biometrics: impact of human learning on performance quality. *Journal of Data and Information Quality*, 2(2):11, 2011.

Park, Sunghoon, Park, Jooseoung, and Cho, Sungzoon. User authentication based on keystroke analysis of long free texts with a reduced number of features. In *Communication Systems, Networks and Applications (ICCSNA), 2010 Second International Conference on*, volume 1, pp. 433–435. IEEE, 2010.

Peacock, Alen, Ke, Xian, and Wilkerson, Matthew. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

Pisani, Paulo Henrique and Lorena, Ana Carolina. Emphasizing typing signature in keystroke dynamics using immune algorithms. *Applied Soft Computing*, 34:178–193, 2015.

Prabhakar, Salil, Pankanti, Sharath, and Jain, Anil K. Biometric recognition: Security and privacy concerns. *IEEE security & privacy*, 99(2):33–42, 2003.

Samura, Toshiharu and Nishimura, Haruhiko. Keystroke timing analysis for individual identification in japanese free text typing. In *ICCAS-SICE, 2009*, pp. 3166–3170. IEEE, 2009.

Shu, Wei and Zhang, David. Automated personal identification by palmprint. *Optical Engineering*, 37(8): 2359–2363, 1998.

Singh, Saurabh and Arya, KV. Key classification: a new approach in free text keystroke authentication system. In *Circuits, Communications and System (PACCS), 2011 Third Pacific-Asia Conference on*, pp. 1–5. IEEE, 2011.

Stefan, Deian, Shu, Xiaokui, and Yao, Danfeng Daphne. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *computers & security*, 31(1):109–121, 2012.

Teh, Pin Shen, Teoh, Andrew Beng Jin, Tee, Connie, and Ong, Thian Song. A multiple layer fusion approach on keystroke dynamics. *Pattern Analysis and Applications*, 14(1):23–36, 2011.

Teh, Pin Shen, Teoh, Andrew Beng Jin, and Yue, Shigang. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013:1–24, 2013.

Uzun, Yasin and Bicakci, Kemal. A second look at the performance of neural networks for keystroke dynamics using a publicly available dataset. *Computers & Security*, 31(5):717–726, 2012.

Vermesan, Ovidiu and Friess, Peter. *Building the hyperconnected society: Internet of things research and innovation value chains, ecosystems and markets*, volume 43. River Publishers, 2015.

Villani, Mary, Tappert, Charles, Ngo, Giang, Simone, Justin, Fort, H St, and Cha, Sung-Hyuk. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop*, pp. 39–39. IEEE, 2006.

Voth, Danna. Face recognition technology. *IEEE Intelligent Systems*, 18(3):4–7, 2003.

Yager, Neil and Dunstone, Ted. The biometric menagerie. *IEEE transactions on pattern analysis and machine intelligence*, 32(2):220–230, 2010.

Yan, Jeff, Blackwell, Alan, Anderson, Ross, and Grant, Alasdair. Password memorability and security: Empirical results. *IEEE Security & privacy*, 2(5):25–31, 2004.

Zhang, Xiang and LeCun, Yann. Which encoding is the best for text classification in chinese, english, japanese and korean? *arXiv preprint arXiv:1708.02657*, 2017.

Zhang, Zhi-Kai, Cho, Michael Cheng Yi, Wang, Chia-Wei, Hsu, Chia-Wei, Chen, Chong-Kuan, and Shieh, Shiuhpyng. Iot security: ongoing challenges and research opportunities. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pp. 230–234. IEEE, 2014.