KOLLECTOR: Detecting Fraudulent Activities on Mobile Devices Using Deep Learning

Lichao Sun, Bokai Cao, Ji Wang, Witawas Srisa-an, Philip S. Yu, *Fellow, IEEE*, Alex D. Leow, and Stephen Checkoway

Abstract—With the rapid growth in smartphone usage, preventing leakage of personal information and privacy has become a challenging task. One major consequence of such leakage is impersonation. This type of illegal usage is nearly impossible to prevent as existing preventive mechanisms (e.g., passcode and fingerprinting), are not capable of continuously monitoring usage and determining whether the user is authorized. Once unauthorized users can defeat the initial protection mechanisms, they would have full access to the devices including using stored passwords to access high-value websites. We present KOLLECTOR, a new framework to detect impersonation based on a multi-view bagging deep learning approach to capture sequential tapping information on the smartphone's keyboard. We construct a sequential-tapping biometrics model to continuously authenticate the user while typing. We empirically evaluated our system using real-world phone usage sessions from 26 users over eight weeks. We then compared our model against commonly used shallow machine techniques and find that our system performs better than other approaches and can achieve an 8.42 percent equal error rate, a 94.24 percent accuracy and a 94.41 percent H-mean using only the accelerometer and only five keyboard taps. We also experiment with using only three keyboard taps and find that the system still yields high accuracy while giving additional opportunities to make more decisions that can result in more accurate final decisions.

Index Terms—Mobile, authorization, privacy, deep learning, multi-view learning

1 INTRODUCTION

The widespread adoption of smart-mobile devices has provided users with "any time, anywhere computing" capability. At the same time, such mobility has also lead to theft as these smart-mobile devices contain as much sensitive and private information as that contained in less mobile devices such as desktops and laptops. In 2013, over three millions Americans were the victims of smartphone theft [1]. When stolen, electronic impersonation is often used to access personal data through these devices as if the actions are performed by the device's owner. Impersonation fraud is a serious problem that costs U.S. companies nearly $180 million between 2013 and 2014 [2].

While setting access codes can safeguard these devices, studies have shown that most owners choose very simple passwords or even no passwords to protect their mobile devices [3], [4], [5]. Many studies have also shown that when passcodes are used, attackers can still reverse engineer passwords by observing screens for taps, fingerprints, and/or smudge patterns [6], [7], [8], [9]. A key aspect of existing protection mechanisms is that they only try to prevent unauthorized users from unlocking devices. Once they are able to bypass these mechanisms, there are no additional mechanisms to continuously prevent them from using the device. As such, it is highly desirable to enhance the authentication mechanisms in smartphones to include additional defensive measures designed to be non-intrusive but which can continuously monitor the user activity such as web browsing or entering information to web applications to prevent unauthorized usage.

Currently, existing continuous monitoring approaches use a variety of standard smartphone sensors coupled with (shallow/traditional) machine learning techniques such as support vector machines (SVMs) to continuously authenticate the user. One notable aspect of much of this existing work is a "closed-world" model, wherein the device owner *as well as all potential attackers* are in the training set. This is clearly an unrealistic model. Furthermore, SVMs and other shallow machine learning techniques are unable to learn the complicated relationships inherent in sequential activities like typing. Despite this limitation, some systems combine several smartphone sensors in order to achieve high accuracy rates and low equal error rates. This additional sensor usage has been shown to degrade battery life [10]. Furthermore, SVM based models have an additional problem in that they perform classification quite slowly [11].

In light of these issues, we set out to construct a continuous identification system subject to the following design requirements. *First*, the system should operate in an "open world" where attackers are not in the training set. This requires a new cross-validation system. *Second*, the system should be able to identify the device user after just a few

- *L. Sun, B. Cao, P.S. Yu, A.D. Leow, and S. Checkoway are with the University of Illinois at Chicago, Chicago, IL 60607 USA. E-mail: {lsun29, caobokai, psyu, aleow, sfc}@uic.edu.*
- *J. Wang is with the College of Systems Engineering, National University of Defense Technology, Changsha, Hunan 410073, P.R. China. E-mail: wangji@nudt.edu.cn.*
- *W. Srisa-an is with the University of Nebraska-Lincoln, Lincoln, NE 68588 USA. E-mail: witty@cse.unl.edu.*

Manuscript received 2 Apr. 2019; revised 17 Dec. 2019; accepted 30 Dec. 2019. Date of publication 6 Jan. 2020; date of current version 4 Mar. 2021. (Corresponding author: Ji Wang.) Digital Object Identifier no. 10.1109/TMC.2020.2964226

keyboard inputs while yielding a high accuracy and a low equal error rate. *Third*, the system should be able to perform the classification quickly while minimizing both the number of employed smartphone sensors as well as the duration they are enabled.

The proposed system, KOLLECTOR, employs a deep learning approach to meet our aforementioned requirements. It continuously monitors and studies sequential tapping biometric behaviors *while a user is typing*. Our system can detect when an unauthorized user begins using the keyboard by collecting sequential tapping features including the duration of a tap and the horizontal and vertical distances between successive taps. KOLLECTOR can protect the phone while user is holding or typing on the mobile. Unlike prior work that employ many sensors [10], our approach only uses one built-in sensor, the accelerometer, to record the position of the device as part of the continuous authentication process.

To evaluate the effectiveness and efficiency of our proposed system, we implemented a specially-designed keyboard for data collection and installed it on Android smartphones. We let 26 volunteers use the phones for eight weeks. These volunteers generated over 14 million accelerometer data points, 1.3 million keystrokes, and over 37 thousand sessions. Each session is created by our system from the user starts to end of typing on the mobile each time. We then evaluated KOLLECTOR's ability to detect unauthorized usage by training and testing our deep learning models. We found that the proposed system can yield 8.42 percent equal error rate, 94.24 percent accuracy, and 94.41 percent h-mean while using only five keystrokes and taking only 1 milliseconds to perform classification. The traditional machine learning algorithms when applied to our dataset take shorter time to perform classification, but also produce higher equal error rate (by 9.69 percentage points), lower accuracy (by 7.64 percentage points) and lower h-mean (by 8.43 percentage points).

This paper makes the following contributions:

- We propose KOLLECTOR, a protection framework to prevent impersonation. Our approach monitors and studies sequential tapping information to continuously authenticate users through our special implementation of a keyboard. The proposed learning system is efficient and can detect the sequential tapping behavior of an unauthorized user within three to five keystrokes.
- We develop multi-view bagging classifier with deep learning structures. Our system is able to detect unauthorized usage while allowing the device's owner to safely access the device.
- We empirically evaluate the approach by comparing its performance to traditional machine learning techniques. Our approach yields more accurate classification results while utilizing only five keystrokes while performing classification within a reasonable time.
- We also discuss some potential policies that could be built on top of KOLLECTOR. In addition, we deploy the proposed model on both mobile and desktop system, and evaluate its efficiency and energy consumption. These results are reported in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2020.2964226.

2 BACKGROUND

In this section, we briefly introduce the deep learning techniques which are integral to our approach.

The problem of continuous identification lends itself well to machine learning. Shallow machine learning algorithms are frequently used for behavioral biometrics. The most common technique is to construct a binary classifier (e.g., asking a question such as "Is the current user the owner of the device?"). Support vector machines and decision trees (DTs) are commonly used for this purpose. However, one major limitation of shallow machine learning is that it can not capture the complex and hidden relationships between the features and sequential keystrokes. To the best of our knowledge, this is the first work to apply deep learning to tapping patterns in order to detect unauthorized users. Prior efforts in NLP and computer vision have shown that deep learning can produce accurate models that can achieve high detection rates [12].

Previous works (see Section 7) use traditional—or shallow—machine learning techniques (e.g., SVM) which cannot capture the sequential information of keystrokes. To improve the classification performance, traditional machine learning must use more features by running more sensors background which leads to excessive energy consumption. Multi-view bagging (MVB) learning is an attempt to solve shortcomings in machine learning and deep learning by constructing multiple "views" of the same data and by focusing on a different set of features for each view. These views are analogous to views in a database. By using multiple views when we train a model, we can achieve higher classification accuracy than we could with a single view.

Bagging learning solves high variance problems by training multiple models and using a voting strategy for evaluation. To the best of our knowledge, this work represents the first approach to use multi-view and bagging for continuous identification on smartphones.

In the next section, we discuss our approach to capture sequential tapping information that our system uses to detect typing by unauthorized users. Our goal is to construct a deep learning system leveraging MVB models that can produce accurate identification with only a few taps on the keyboard.

3 KOLLECTOR: A FRAMEWORK TO DETECT FRAUDULENT USAGE OF SMART-PHONES

We propose KOLLECTOR, a multi-view bagging fraudulent usage detection framework via a deep structure. The proposed framework contains three main steps to detect fraudulent usage. This process is illustrated in Fig. 1 and summarized below:

1) In the first step, we create a custom software keyboard that can monitor and collect keyboard usage information. The new keyboard is then installed on smartphones that will be used in this study.

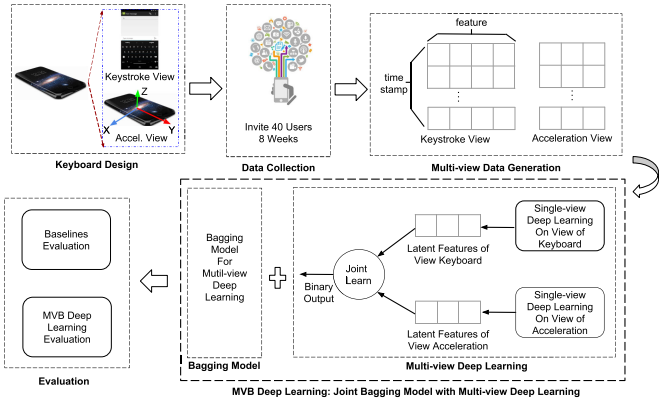


Fig. 1. Overview of KOLLECTOR.

2) In the second step, the usage information is collected. The scale and scope of data collection can vary but in this study, we invited 40 volunteers to use smartphones equipped with the custom keyboard for 8 weeks. We retrieved the sequential tapping information from each phone in real time. We then process the collected data to build a multi-view version that can be used to detect fraudulent usage.
3) Once the data has been collected, we use the multi-view data and perform multi-view bagging learning via a deep structure. To evaluate the effectiveness of MVB, we also compare the performance of the proposed approach with those of the traditional machine learning techniques to perform similar fraudulent usage detection. The techniques that we used to compare include support vector machine, decision tree and random forest.

Next, we describe each of these steps in turn.

3.1 Design of Keyboard to Collect Sequential Tapping Information

We implemented our keyboard by extending an open-source keyboard software.[1] Our extension, consisting of several hundred lines of code, provides the keyboard logging capability to track information that we are interested in (e.g., pause, duration, distance). The modified keyboard serves two purposes. First, we want it to be able to operate like a normal keyboard that records tapping data information. In addition, we enhance the keyboard to also collect sequential information that can be used to help identify users and detect usage patterns indicating fraudulent usages.

The sequential information that we collect from the keyboard includes tap duration, pause (i.e., time since last key), and vertical and horizontal distance from the previous tap. We do not collect the individual key presses; however, we do record the type of key pressed (letters, numbers, symbols, space, and backspace). One consequence of this is if two 2-gram tap sequences are the same, then the keystroke distances must be the same, whereas if the two 2-grams are different, then the keystroke distances are likely to be, but not necessarily, different. So if two users like to use common phrases or 2-grams, we will observe some common stroke distances in their typing. That is to say although the exact word sequences are not provided, we will still see their similarity from the keystroke distances.

In addition to differences in what users tapping, each user may hold a device in a specific way while typing. As such, being able to capture tilting information of the device can also help with user identification. To capture this feature as part of training and monitoring, we use the accelerometer to provide the X,Y,Z coordinates information.

The collected tapping information forms one view of our data and the acceleration measurements form the second view. Our multi-view learning approach described below makes use of these two views to provide more accurate classification than that could be achieved if they were combined into a single view.

Using only a few features means that we need only a few sensors to operate and only for a short time period. This can lead to higher energy efficiency than approaches which use many sensors or which require the sensors to be on all the time.

3.2 Data Collection and Processing

To preserve user's privacy, we applied for IRB approval and was approved prior to any data collection. Data was collected and securely stored. The information was anonymized prior to processing.

We recruited 40 volunteers for this study, all of whom had extensive prior experience with smartphones. The volunteers ranged in age from 30 to 63.

Out of the 40 volunteers, we find that 26 of them (17 women and 9 men) used the provided phones at least 20 times in 8 weeks; the most active participant used the phone 4,702 times while the least active participant only used the phone 29 times. Since deep learning requires more training data than traditional machine learning, we do not use any data from the volunteers who used the phone less than 20 times.

When collecting tapping data, not every tap has all of the features we use. For example, the distance from and time since the previous tap can not be computed for the first tap in a typing session. In this case, we set a value of 0 for such features. All features are normalized to the range [0,1].

A typical use of the keyboard results in a *session* consisting of one or more keystrokes. For example, a simple message such as "How are you?" contains sequential keystrokes as well as multiple types of key presses (letters, spaces, and special characters).

We denote the ith user's jth session by s_{ij}. Each session contains two different types of sequential data: the tapping data and the acceleration data. Let $x = (c^{(1)}, c^{(2)})$ where $c^{(1)}$ is the time series of keystrokes information, and $c^{(2)}$ is the time series of accelerometer values. Note that we also use $\mathbf{V}$ to represent different views of the data set, e.g., $\mathbf{V_1}$ is view of keystrokes data information, and $\mathbf{V_2}$ is view of accelerometer data information. It is difficult to align the sequential features in different views because of different timestamps and sampling rates. For example, accelerometer values are much denser than special character keystrokes. Therefore, it is intuitive to treat $c_{ij}^{(1)}$, and $c_{ij}^{(2)}$ as multi-view time series that contains the complementary information for user identification.

1. The software keyboard is available from https://github.com/AnySoftKeyboard/AnySoftKeyboard.

TABLE 1
Symbols and Notation

Symbol	Meaning
$\mathbf{c}$	A record generated during keystroke
$\mathbf{x}$	A keystroke session, inputs of the model
$\mathbf{V}$	Views of Inputs
$\mathbf{1}$	A vector of 1s
$y(\mathbf{x})$	$y(\mathbf{x}) = 1$ means the session $\mathbf{x}$ is predicted as owner, 0 otherwise
$\sigma_g(\cdot)$	The sigmoid function
$\sigma_h(\cdot)$	The hyperbolic tangent

3.3 Generating Training Models

We create two machine learning models to help detect fraudulent smartphone usage. First, we create a multi-view deep learning model to distinguish any two people in the data set based on their unique sequential tapping information. This enables us to validate that individual users really do have different sequential tapping behaviors that we can measure. Next, we create a multi-view bagging deep learning model which we use to distinguish a device's owner from an unauthorized user.

3.3.1 Single View on Deep Learning Structure

Deep learning [13] is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data which also called deep structured learning, deep neural network learning or deep machine learning. Deep learning is a concept and a framework instead of a particular method. In deep learning area, there are two main branches: Recurrent Neural Network (RNN) [14] and Convolutional Neural Networks (CNN) [15]. CNN is frequently used in computer vision areas and RNN is applied to solve sequential problems such as nature language process. In the model, we implement Gated Recurrent Unit (GRU) [16], which is a special case of Long Short Term Memory networks (LSTM) [17] in the framework of RNN.

Long Short Term Memory network is a special case of RNN, capable of learning long-term dependencies [17]. RNN only captures the relationship between recent keystroke information and uses it for prediction. LSTM, on the other hand, can capture long-term dependencies. Consider trying to predict the tapping information in the following text "I plan to visit China... I need find a place to get some Chinese currency". The word "Chinese" is relevant with respect to the word "China", but the distance between these two words is long. We need to use LSTM to capture information of long-term dependencies instead of the standard RNN model.

While LSTM can be effective, it is a complex deep learning structure that can result in high overhead. Gated Recurrent Unit is a special case of LSTM but with simpler structures (e.g., using fewer parameters) [16]. In many problem domains including ours, GRU can produce similar results to LSTM. In some cases, it can even produce better results than LSTM. We illustrate the GRU framework and describe the components next. Note that we describe symbols and notations used to define these components in Table 1.

The output vector h_t is a hidden state which can be considered as a compact representation of input sequence $[x_1, \ldots, x_t]$. It is a linear interpolation between the previous output vector h_{t-1} and the candidate state $\tilde{h}_t$ at order t in GRU,

$$h_t = z_t \tilde{h}_t + (1 - z_t) h_{t-1}, \tag{1}$$

where z_t is the update gate which decides how much the unit updates its activation or content. Here, z_t is computed as

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1}). \tag{2}$$

where W, U are parameter matrices, $\sigma_g(\cdot)$ is the sigmoid function $\sigma_g = 1/(1 + e^{-x})$, and x_t is the sequential input at order t in the input data set.

The GRU reset gate r_t can effectively make the unit act as if it is reading the first symbol of an input sequence. In essence, the reset gate allows it to forget its previously computed state. It is computed similarly to the update gate

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1}). \tag{3}$$

The candidate activation $\tilde{h}_t$ is computed similarly to that of the traditional recurrent unit

$$\tilde{h}_t = \sigma_h(W x_t + U(\mathbf{r}_t \odot h_{t-1})), \tag{4}$$

where $\mathbf{r}_t$ is a set of reset gates and $\odot$ is an element-wise multiplication, and $\sigma_h = \tanh$ is the hyperbolic tangent.

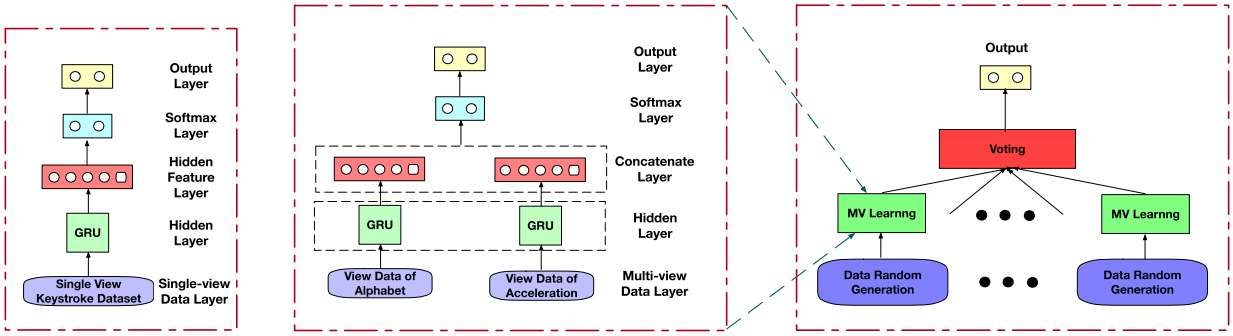
One advantage of the GRU over LSTM is that it is easy for each unit to remember the existence of a specific pattern in the input stream over a long series of time steps. Any information and patterns will be overwritten by update gate due to its importance.

We can build single-view binary-classification deep learning model using GRU as shown in Fig. 2a. We choose any single view of the dataset, such as a view of letters. We use the normalized dataset as the input of GRU. GRU will produce a final output vector which can help us to continuous identification.

3.4 Multi-View Bagging Deep Learning

Now, we discuss the approach to apply deep learning for constructing the fraudulent usage detection model. The approach is based on multi-view bagging learning with a deep structure.

As mentioned previously, we employ two different views. Each view V_i contains different number of features and different number of samples. One possible approach is to concatenate the two views into a single view as shown in Fig. 2a, which is also named early fusion. However, this does not work because the number of features, and the number of records in each view of a session are different. Instead of using early fusion to concatenate multiple views into one view, we decide to use multi-view deep learning with late fusion. First, we choose to model each view separately. Then, we use deep learning model to find the latent vector representation of each view. Last, we concatenate the latent vectors of each view for fraudulent activity prediction, which is named late fusion. Multi-view learning based on late fusion can avoid losing information as in the case when multiple views are combined to create single view. One key piece of information that we want to preserve is the sequence of keystrokes. By using multi-view, we are



(a) Single-view deep learning (b) Multi-view deep learning (c) Multi-view bagging deep learning

Fig. 2. A comparison of different learning frameworks.

able to maintain each view separately but then use multiple views to make predictions.

In this manner, we develop a multi-view deep learning model. This model performs better than the single-view since it can better utilize the information from different views of the dataset. In our work, we generate the multi-view learning model to detect attackers when we can well separate one user with other users in our training dataset. The model is shown in Fig. 2b. First, we randomly separate the data set into training and testing data with multiple views. In this case, we have a view of keystrokes information, and a view of accelerometer information. Then, we use GRU-BRNN to build the hidden layers for each view. We then concatenate the last hidden layer information from each GRU-BRNN model. We use the last concatenated layer, which contains all information from different views for authorization. In the output layer, we use softmax function to perform binary classification. As the GRU extracts a latent feature representation out of each time series, where the notions of sequence length and sampling time points are removed from the latent space, this avoids the problem of dealing directly with the heterogeneity of the time series from each view.

Our approach uses fully connected deep learning model. This is a straightforward and efficient model. As the first step, we apply GRU on each view to generate the latent vectors of each view. Then, we apply late fusion to concatenate latent feature vectors from multiple views together, i.e., $h = [h^{(1)}; \ldots; h^{(k)}] \in R^d$, where d is the total number of multi-view features, and $d = 2kd_v$ for BRNNs, where k is the number of the views and d_h is the number of recurrent units. We then feed forward h into one or more fully connected neural network layers with a nonlinear function $\sigma(\cdot)$ between each layer. The number of views of KOLLECTOR, k, is two, then the output $\hat{y}$ of multi-view deep learning is defined as

$$\hat{y} = W^{(2)} * \sigma(W^{(1)}[h^{(1)}; h^{(2)}; 1]), \tag{5}$$

where $W^{(1)} \in R^{q\times(d+1)}$, $W^{(2)} \in R^{c\times q}$, q is the number of hidden units, c is the number of classes, and the constant signal "1" is to model the global bias.

Note that in deep learning, different optimization functions can greatly influence the training speed and the final performance. There are several optimizers such as RMSprop and Adam [18]. In our work, we use an improved version of Adam called Nesterov Adam (Nadam) which is a RMSprop with Nesterov momentum.

After we generate the multi-view deep learning model for fraudulent usage detection in the training process, we apply a bagging policy in the testing process as shown in Fig. 2c. Generally, we use random initialization of an odd number m of the same dataset, and then choose to generate learning models for each initialization. Next, we use different trained models with one testing data set. Finally, we apply a majority voting strategy for the final output. For example, KOLLECTOR has three models for three random initializations in the training process. In testing process, for each input sample, we apply all three models on the this sample to generate three detection results and output the majority result by the bagging policy. The bagging policy is majority voting,

$$\hat{y}_{output} = vote\left(\sum_{j=1}^{m} \hat{y}_j\right), \tag{6}$$

where $\hat{y}_i \in [0, 1]$, $i \in \{1, 2, 3, \ldots m\}$, and $vote(x)$ is 1 when $x > m/2$, 0 otherwise.

After applying the bagging strategy with multi-view deep learning, the proposed system becomes more robust, since one initialization of the dataset may cause wrong classification for some samples which can be well-classified with other initializations. The idea is applied similarly in traditional machine learning techniques such as random forests.

In our experiments, bagging produced a more accurate classifier than using a single model.

4 EXPERIMENTAL METHODOLOGY

To examine the performance of KOLLECTOR in identifying fraudulent activities in mobile apps, we conducted extensive experiments on our real-world data set and compared the results with those from several state-of-the-art shallow machine learning methods. As shown in Table 2, our dataset include over 1.3 million keystrokes and 14 million accelerometer data points. These data points can be grouped into more than 37 thousand sessions. Note that the length of a sequence is measured in terms of the number of data points in a sample rather than a duration in time.

Next we describe our features and the method that we used to collect the data, the baseline systems that we used for comparisons with KOLLECTOR, the metrics used for evaluation, and lastly, the cross-validation system.

TABLE 2
Dataset Statistics

Statistics	Keystroke	Acceleration
Number of data points	1,374,547	14,237,503
Number of sessions	34,993	37,647
Minimum session length	14	259
Maximum session length	538	90,193

4.1 User Data and Pattern Analysis

In this section, we study the collected data for participants who had used the phones at least 20 times during the study duration of eight weeks. Through studying the feature patterns of the dataset and the difference of feature patterns between different users, we can find which pattern is most effective for user authorization. Since we have two views of dataset and we have different features in each view, we now introduce each view in the following sections.

4.1.1 Keystroke Information Data

In the keystrokes information view, we first analyze the four features including duration of a keystroke, time since last keystroke, and distances from last key along two axes. In Fig. 3, we shows the complementary cumulative distribution functions (CCDFs) of four different features. We illustrate the CCDFs of (a) the duration of a keystroke, (b) the time since last keystroke, (c) the distance from last key along the x-axis, and (d) the distance from last key along the y-axis. From the CCDFs of the duration of a keystroke in Fig. 3a, it is clear that most users have their own unique distribution. We also find that for most users, their durations are very fast with median of 85ms. Although tap duration is a good feature for distinguishing different users, the distributions of the durations of some users are very close, so we need other features to help separate them.

Next we see the CCDFs of time since last keystroke in Fig. 3b. From the figure, the pause time since the last keystroke is also a good feature for distinguishing different users due to different distribution of pauses between different users. Overall, this distribution is heavily skewed, with most time intervals being very short with median of 380ms. Like the duration of a keystroke, we cannot use only the time since the previous keystroke to separate all users.

From the CCDFs of the distance from the previous key along the x-axis (Fig. 3c) and the distance from the previous key along the y-axis (Fig. 3d), it is clear that the distributions of different users are very similar. This is not surprising since most users use similar keys on the keyboard. They also often use common words. If we use traditional machine learning, these two features may hardly help to distinguish the users. However, by using MVB with GRU, we can capture the sequence information of the keystroke. So, these two features help to further improve the performance of KOLLECTOR.

Fig. 5 shows the scatter plot between rates of different types of keystroke. We use one-hot encoding for typing behaviors including alphanumeric characters, backspace, space, and other. In these four different type of keystrokes, compared to alphanumeric characters, the others are usually much sparser. From the figures, we can observe some interesting patterns. For example, the rate of alphanumeric keys is negatively correlated with the rate of backspace (from the subfigure at the 1nd row, 4th column). On the diagonal there are kernel density estimations. It shows that the rate of alphanumeric characters is generally high in a session, followed by space and backspace. In the next section, we will analyze the other view of the inputs.

4.1.2 Accelerometer Information Data

We also collect the accelerometer value while users are tapping on the keyboard. No matter how the user holds the phone or moves around, the accelerometer values of users are different which can be used for fraudulent detection. Accelerometer values are recorded by the accelerometer

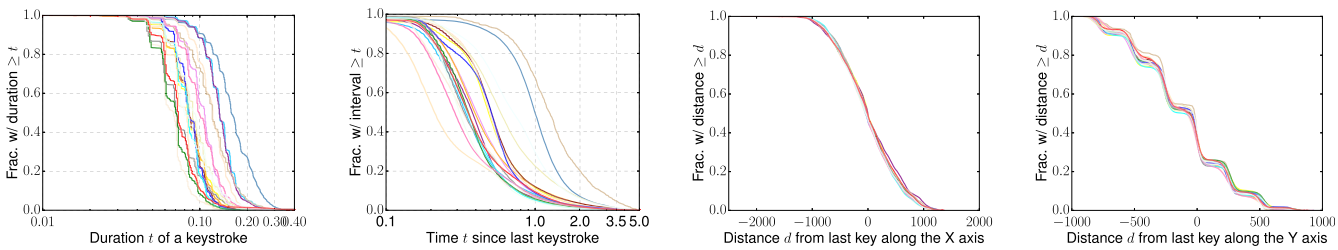


Fig. 3. Complementary Cumulative Distribution Functions (CCDFs) of four features. Each colored line represents one user.

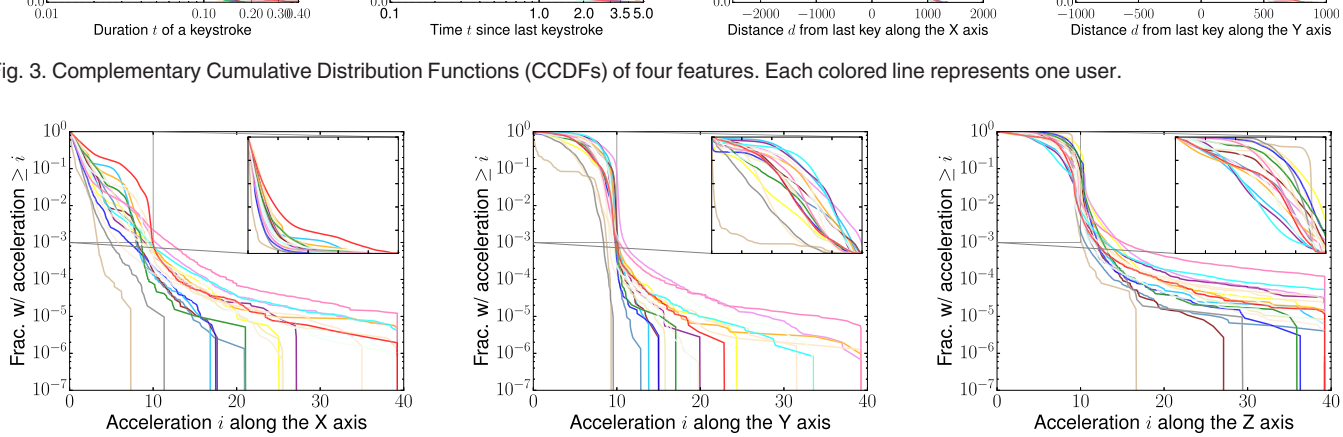


Fig. 4. CCDFs of absolute acceleration along the three axes.

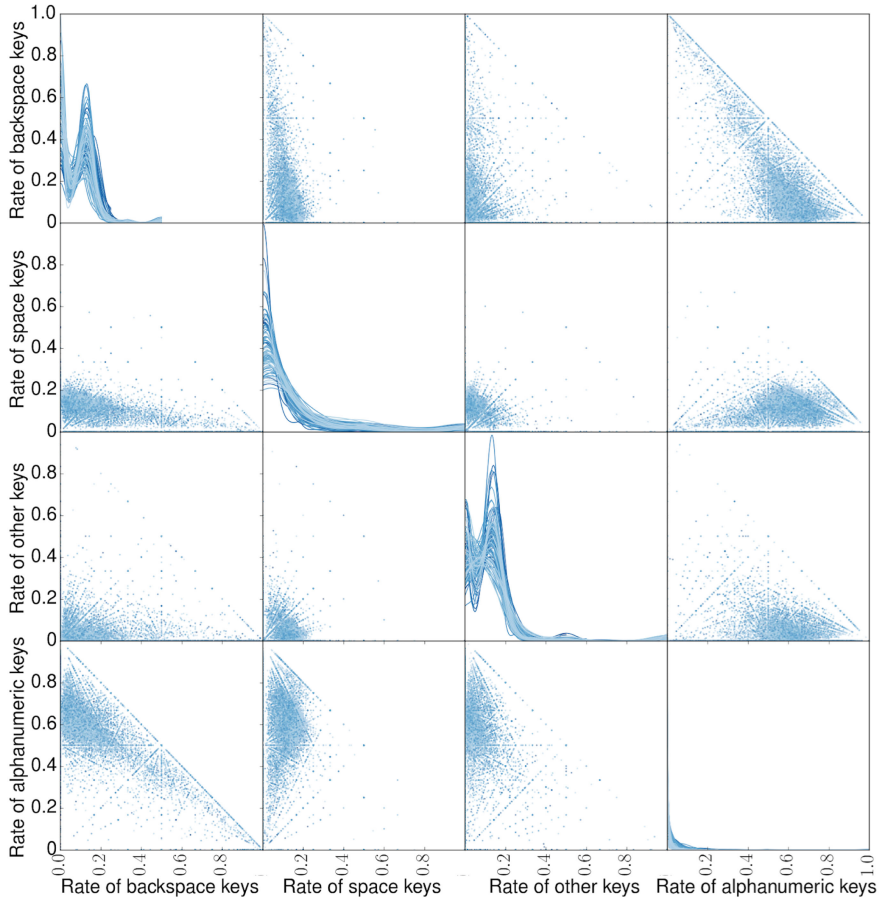


Fig. 5. Scatter plot between rates of different type of keystrokes.

sensor which is running in the background during an active session regardless of a person's typing speed. In Figs. 4a, 4b, and 4c, the CCDFs of accelerometer values around three axes are displayed. We can see that accelerometer values on three axes have different CCDFs, and all of accelerometer features can help to distinguish users due to unique distribution. To discover more values from the accelerometer, we also study the distribution of two or more axes of accelerometer in Figs. 6a, 6b, and 6c. We find that the distribution of two axes of accelerometer can do better separation between different users. For example, the distributions among five users between y- and z-axes show distinct characteristics (Fig. 6c).

4.2 Baseline Systems: Keystroke-Based Behavior Biometric Methods for Continuous Identification

Previous work on keystroke-based continuous identification with machine learning techniques [6], [19], [20], [21], often utilizes shallow learning techniques such as Support Vector Machine, Decision Tree, and Random Forest [22], [23], [24]. As such, we compare the performance of KOLLECTOR against those shallow learning techniques. To do so, we apply these techniques on our dataset, which contains keystroke and accelerometer information. It is worth noting that compared to some prior work on performing continuous identification based on keystroke information [19], our approach uses fewer sensors. Next, we briefly describe these baseline techniques in turn.

Support Vector Machine (SVM). SVM is widely used in many previous works [6], [19], [20]. SVM is a linear model that finds the best hyperplane by maximizing the margin between two classes. In this case, we need to find the maximum margin with hyperplane to distinguish between authorized and unauthorized usage of a device. Bo *et al.* [19] achieves 80 percent accuracy by using 10 keystrokes, and Alghamdi *et al.* [20] achieves 12.2 percent EER by using SVM

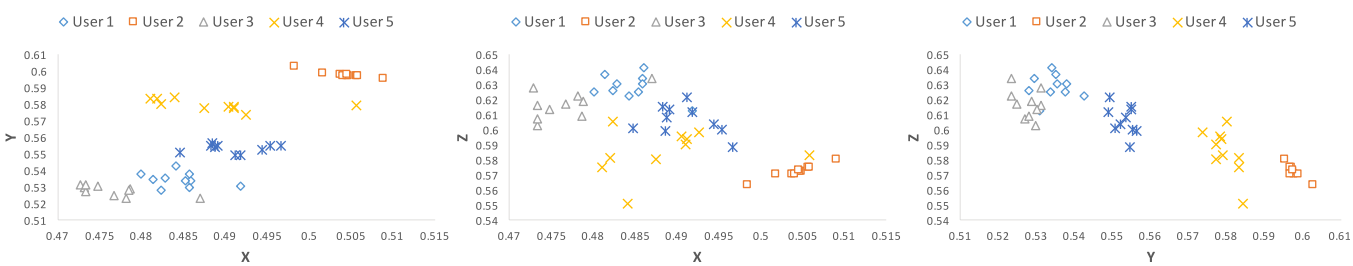


Fig. 6. Distribution of five users' accelerometer readings.

and keystroke information. A C-SVC type SVM is used in the following experiments. The cost is set to 10. We use the radial basis function kernel.

Random Forests and Decision Trees. Other learning methods such as random forests and decision trees have not been widely adopted for continuous identification based on keystroke information. A decision tree is an interpretable classification model for binary classification. It is a tree structure, and features form patterns that are nodes in the tree. Random forests are an ensemble learning method for classification that builds many decision trees during training and combines their outputs for the final prediction. Sun *et al.* [25] shows tree structure methods can work more efficiently than SVM, and can do better binary classification compared to SVM. For the random forests, the number of trees is set as 3 and the max depth of the tree is 3. The max depth of the decision tree classifier used in the following experiments is set as 3.

For all shallow models, we first collect the data from each view. Then, for each session, we concatenate each view information as a long vector. Since we only use the first three keystrokes data in one session, the size of input is fixed.

4.3 Metrics

We use three measures to evaluate the performance of KOLLECTOR. These measures are *accuracy*, *H-mean*, and *equal error rate*. For each testing session $\mathbf{s}$, we make a binary prediction. There are four possible outcomes.

1) A session by an owner is classified as owner's session by a fraudulent activities detector. This is referred to as *True Positive* (*TP*).
2) A session by an owner is *not* classified as owner's session by a fraudulent activities detector. This is referred to as *False Negative* (*FN*).
3) A session by others is classified as others by a fraudulent activities detector. This is referred to as *True Negative* (*TN*),
4) A session by others is *not* classified as others by a fraudulent activities detector. This is referred to as *False Positive* (*FP*).

Our first standard metric, accuracy, is the proportion of correctly classified sessions in all sessions

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + TN}. \tag{7}$$

Our second standard metric, H-mean [26], [27], is the harmonic mean of the *True Positive Rate* $TPR = \frac{TP}{TP+TN}$ and the *True Negative Rate* $TNR = \frac{TN}{TN+FP}$. That is,

$$\text{H-mean} = 2\frac{TPR \cdot TNR}{TPR + TNR}. \tag{8}$$

Our final standard metric, equal error rate (EER) is a biometric security system algorithm used to measure the system performance against another system (the lower the better). First we need to calculate False Alarm Rate (FAR) and False Reject Rate (FRR, or missing rate), where $FAR = 1 - TNR$ and $FRR = 1 - TPR$. When the two rates are equal, the common value is referred to as the EER. However, if two rates are not equal, we find the EER by finding the at least nearly equal or the min distance.

4.4 Our Cross-Validation System

Because KOLLECTOR operates in an "open world" fashion, activities from attackers are not included as part of the training set. However, KOLLECTOR needs to be able to distinguish between two possible users. Thus, we also need to include positive data (owner's activities) and negative data (activities by others). Therefore, we need to develop a new cross-validation system to evaluate our work. Our cross-validation system adheres to the following rules.

1) We set the data of the device owner as positive data. Then we select 80 percent of the remaining data as negative data in the training process. We save 20 percent as attackers in the testing process. So for example, if we have 11 users (user1 to user11) in our study, user1 is a device owner. We then use data from other 8 users (e.g., user2-user9) as negative data for training and cross-validating. Data generated by user10 and user11 is used during the testing process.
2) Next we repeat the process but using positive data from another randomly selected owner (e.g., user2) and again negative data from another 8 users (e.g., user3 to user10). The remaining data from the other two users (e.g., user1 and user11) is used as attacker's data during the testing process.
3) Each person is once used as an attacker in the system.

To implement these rules, we separate negative people besides the owner into five groups. Two different groups would share same users' data. Each time, we use four groups as negative data in the training process and the last group as attackers in the testing process. Every group would be once used as attackers in the testing set.

5 RESULTS

In this section, we report the results of our experiments to evaluate: (1) the ability of KOLLECTOR to detect fraudulent activities conducted by unauthorized users, and (2) the efficiency of KOLLECTOR.

Next, we report the results of our investigation in turn.

5.1 Fraudulent Activities Detection

In this section, we evaluate the effectiveness of KOLLECTOR to detect fraudulent activities carried out by unauthorized users. The parameter configuration is shown in Table 6. We construct two separate models to apply to two different situations: one is for performing authentication using five keystrokes whereas the second uses only three keystrokes.

Unlike the previous section where we tried to distinguish users for which we have training data, in this section, we consider an "open world" where we want to distinguish a device owner from an attacker we have never seen before. To this end, we perform a series of experiments using the data from the 26 volunteers from our study who produced at least 20 typing sessions. In each experiment, we select one volunteer as the device owner, five as attackers, and the remaining 20 are used along with the owner to construct

TABLE 3
Results of Detectors During Three Keystrokes Information

System	TPR (%)	TNR training (%)	TNR attacker (%)	Accuracy (%)	H-mean (%)	EER (%)
KOLLECTOR	88.73	94.08	93.38	94.07	93.66	8.94
KOLLECTOR (no bagging)	84.10	90.07	90.56	90.07	90.28	12.67
Random forest	84.30	88.95	79.20	88.96	81.99	18.25
Decision tree	88.26	92.01	69.27	91.49	77.48	21.24
SVM	83.50	88.97	71.31	88.84	73.92	22.59

TABLE 4
Results of Detectors During Five Keystrokes Information

System	TPR (%)	TNR training (%)	TNR attacker (%)	Accuracy (%)	H-mean (%)	EER (%)
KOLLECTOR	88.55	94.24	94.60	94.24	94.41	8.43
KOLLECTOR (no bagging)	83.89	89.79	89.82	89.79	89.79	13.15
Random forest	83.82	88.71	79.80	88.75	82.21	18.19
Decision tree	88.62	92.19	68.51	91.65	76.82	21.43
SVM	84.79	89.82	64.21	89.50	69.27	25.50

our models. The attackers' data do not appear in the training set for a given model. The owner and other 20 users data is split into no-overlapping training and testing sets.

Our goal is to evaluate how well KOLLECTOR can distinguish the device owner from the attackers. To that end, we measure the TPR, the TNR for attackers, the accuracy, H-mean, and the EER. For completeness, we also report the TNR when we instead consider the other 20 volunteers whose training data we used to build our models. Tables 3 and 4 show our results when considering only the first three keystrokes and the first five keystrokes of a typing session.

For comparison with shallow machine learning approaches, Tables 3 and 4 show the classification results when using random forests, decision trees, and SVMs. To show the effect that bagging has on our results, we also report our results when we do not use bagging. As the tables show, KOLLECTOR (with or without bagging) consistently outperforms the shallow learning techniques.

Unlike the case of distinguishing two users whose training data we have from the previous section, random forest shows best performance on fraudulent detection instead of decision tree. The difference between random forest and decision tree is that random forest is a bagging decision trees. This indicates that bagging provides a not insubstantial improvement. Indeed, after we apply bagging with multi-view, we can see that KOLLECTOR shows much better performance compared to KOLLECTOR without bagging. In particular, for the five keystroke model, the accuracy improves by 4.45 percentage points, the H-mean by improves by 4.62 percentage points, and the EER improves by 4.72 percentage points. We see a similar, if smaller, improvement with the three keystroke model.

TABLE 5
Results Using Different Views of the Metadata

Parameter	Accuracy	H-mean	EER
Keystroke Only	89.23%	84.73%	16.73%
Accelerator Only	89.26%	85.66%	15.62%
All (no bagging)	90.07%	90.28%	12.67%
All (bagging)	94.07%	93.66%	8.94%

Table 5 gives the performance of using different features of the data. As we can see, both keystroke information and accelerometer information can detect fraudulent activities by itself with a relatively high accuracy. Their performance is similar to each other. Further, if we combine them together, we can achieve much better performance by using the bagging strategy.

These results show that with just a few taps, KOLLECTOR can distinguish the device owner. This is precisely what we need to perform continuous identification while meeting all of the design constraints listed in Section 1. (The runtime performance is discussed in the following section.)

5.2 Efficiency

To evaluate relative efficiency of KOLLECTOR compared to shallow machine learning techniques, we employ a 15" MacBook Pro with a 2.5 GHz Intel Core i7 and 16 GB of 1600 MHz DDR3 memory, and NVIDIA GeForce GT 750M with 2 GB of video memory. We test each technique five times. Table 7 reports the per-session classification time for each technique, including the average running time and the variance of running times. We see that SVM, when used for continuous identification of sequential data set is the slowest technique. Decision trees are the fastest and KOLLECTOR and random forests are smaller than 1×10^{-3} ms per classification.

TABLE 6
Parameter Configuration

Parameter	Value
#class	2
#hidden units (h)	16
#epochs	371
batch size	256
learning rate	0.1
dropout fraction	0.1
maximum sequence length	3,5
minimum sequence length	2
train/test split	0.2
loss	binary_crossentropy
optimizer	Nadam()

TABLE 7
Per-Session Classification Time

System	Time (ms)
KOLLECTOR	0.99 ± 0.020
SVM	1.37 ± 0.084
Random forest	6.30e-4 $\pm$ 4.1e-5
Decision tree	3.42e-4 $\pm$ 2.6e-5

Based on the run times on a laptop, we believe that whenever a SVM is sufficiently performant for use in continuous identification—as with much prior work—KOLLECTOR is as well. For more experimental results on a phone, please refer to the supplemental material, available online.

6 DISCUSSION

In this section, we briefly discuss some potential policies that could be built on top of KOLLECTOR. For more discussions, please refer to the supplemental material, available online.

Policies: KOLLECTOR only performs identification. We have not prescribed any action that should occur when KOLLECTOR detects that the smartphone's owner is not the one typing on the phone. Indeed, this sort of policy decision is out of scope of our paper. Nevertheless, we layout a few potential policies that could be implemented on top of KOLLECTOR.

- Reporting only. One simple policy is to send a report of the unauthorized access to a server. Under this policy, an attacker would be able to use the phone as normal, but could not do so in a stealthy fashion. The policy does have the advantage that the impact of KOLLECTOR on the device owner is minimal since false alarms do not block functionality.
- Multiple decisions. KOLLECTOR can perform accurate detection by using only a short sequence. This would mean that it can divide a long input sequence into smaller chunks, each can then be used to form a decision. Based on all the decisions from these chunks, we can develop a policy that considers all these decisions to form the final decision. This can potentially improve the accuracy over an approach that considers the entire long input sequences. For example, KOLLECTOR can make 5 decisions with 15 keystrokes inputs. If more than half of these five decisions are the final decision, with the reported 8.94 percent EER, the percentage to get a wrong final decision after using 15 keystrokes is only 0.715 percent (i.e., $C_5^2 * (8.94\%)^3 = 0.715\%$).
- Immediate lockout. On the other extreme, we can consider a policy of immediately locking the phone until the owner can provide some additional form of authentication. In this case, an attacker would be unable to do anything with a phone that involves typing without being quickly locked out. However, any false alarms also lock the owner out. This policy is unlikely to be useful in practice.
- Less restrictive than the previous policy. We can also consider a configurable threshold of the number of allowable alarms in a given period. As long as KOLLECTOR does not raise more alarms than the threshold, nothing happens. Once the number of alarms reaches the threshold, the phone locks. Separate from and lower than the lock threshold, we can envision a report threshold where once the number of alarms reaches the report threshold, a report is sent to the server. This policy has the advantage of decreasing the number of times the phone locks but gives the attacker the ability to do more damage.

7 RELATED WORK

This section provides an overview of prior efforts on continuous user identification based on biometrics.

Continuous authorization approaches have been studied over the past several years. Most techniques have been deployed in desktop environment and the features they focused on mainly include physiological information such as facial features and iris features [28], [29], [30]. These approaches often require high computing power to work well as they tend to be intensively processing images. As such, deploying them in desktop environments is sensible. However, mobile devices have more stringent energy constraints so these approaches are not commonly used. Instead, keystroke information has also been used for continuous identification in smart-mobile platforms [31], [32], [33]. Many researchers in behavioral biometric have been studying tapping behavior in smartphone platforms, since these are the most straightforward features and information can be easily collected from the smartphone.

Furthermore, smartphones often contain more sensors such as accelerometer and gyroscope that can be helpful in doing continuous identification. Miluzzo *et al.* [6] introduced a framework called TAPPRINTS to illustrate tapping locations on a touch screen can be computed by using accelerometer and gyroscope sensor data. Bo *et al.* [34] showed how to use inertial sensors to determine whether users are texting while driving. The following up work [19] performed continuous user identification based on taps and motion sensors. Zheng *et al.* [35] discovered that individual users have their own interacting behavior patterns on the touch screen, and the use of motion sensors can help identify different users. These prior efforts provide us with approaches that we can use to implement the keyboard to capture sequential tapping information. However, these approaches tend to rely on shallow machine learning to do binary classification to tackle unauthorized and fraudulent device usage. Our approach employs deep learning to uncover inherent tapping behaviors that are useful for identifying users.

Most recent efforts also focus on studying touchscreen gestures [36], [37], [38], [39] or behavioral biometric behaviors such as reading, walking, driving and tapping [19], [34], [39]. Sitova *et al.* and Bo *et al.* [19], [39] discovered that people exhibit different behaviors that can be detected by sensors based on their current activities (e.g., sitting is a much easier task than walking). Work by Sitova *et al.* [39] also showed that using features such as keystroke and gestures can help to improve the classification performance. However, they did not disclose their methods to train their classification system. Instead, they only claimed to use shallow learning and ten cross-validation techniques. Many of them did not work. Work by Bo *et al.* [19] is closet to our

work as their approach also collects continuous tapping and other information to distinguish guests and the owner. But the accuracy of their approach is only 80 percent when they have 10 keystrokes as input. Our approach can produce higher accuracy while using only 5 keystrokes for identification. None of prior systems using keystrokes information can achieve our performance with only 3 keystrokes.

A state-of-art systems can achieve 5.84 percent EER by studying multi-touch gestures [38]. They studied different gestures for authorization. By studying different movements with different number of fingers, they can achieve high accuracy with only 5.84 percent EER. However, their goal is not to target detection of fraudulent activities that can occur through keyboard usage. Instead, their work focuses on general authorization by leveraging a large amount of information. Our work, on the other hand, mainly focuses on short sequential input for identification. Typing is a more straight-forward way for an attacker to interact with a device to perform unauthorized activities. As such, our work is more applicable than approaches that perform identification and authorization mainly through gestures.

As previously mentioned that existing approaches tend to employ shallow learning for identification. We would like to point out that the major benefit of using deep learning over shallow learning is that deep learning can generate many latent features to capture different sequential information and give a final embedding space for detection. These latent features are only generated by algorithms instead of collecting them directly. Because of using latent features from our designed model, KOLLECTOR can use fewer features, but achieve 94.07 percent accuracy and 8.9 percent EER which is better than the existing approaches [19], [39]. When we use five keystrokes, we can achieve even better performance with 94.24 percent accuracy and 8.4 percent EER.

8 CONCLUSION

We propose KOLLECTOR, a new framework for continuous user identification. We use sequential tapping information to construct a powerful detector by using state-of-the-art learning methods. We also experiment with using only three keystrokes and find that the system still yields high accuracy while giving additional opportunities to make more decisions that can result in more accurate final decisions. Comparing to other shallow machine learning methods, KOLLECTOR is more effective at detecting fraudulent usage while being highly efficient. Therefore, it is feasible to use KOLLECTOR for real life.

ACKNOWLEDGMENTS

This work was supported in part by NSF under grants III-1526499, III-1763325, III-1909323, CNS-1930941, and CNS-1626432, as well as the Scientific Research Project of National University of Defense Technology under grant ZK19-03.

REFERENCES

[1] T. Mogg, "Study reveals americans lost $30 billion worth of mobile phones last year," Mar. 2012. [Online]. Available: http://www.digitaltrends.com/mobile/study-reveals-americans-lost-30-billion-of-mobile-phones-last-year/

[2] S. Watson, "Impostor fraud: A cyber risk management challenge," May 2015. [Online]. Available: http://www.treasuryandrisk.com/2015/05/05/impostor-fraud-a-cyber-risk-management-challenge?slreturn=1485885396

[3] K. Knibbs, "Start memorizing your six-digit iPhone passcode," Sep. 2015. [Online]. Available: http://gizmodo.com/start-memorizing-your-six-digit-iphone-passcode-1710072672

[4] D. Amitay, "Most common iPhone passcodes," Jun. 2011. [Online]. Available: http://danielamitay.com/blog/2011/6/13/most-common-iphone-passcodes

[5] Panda Security, "75 million smartphones in the us don't have their passwords set on," Sep. 2015. [Online]. Available: http://www.pandasecurity.com/mediacenter/tips/smartphone-risk-dont-use-password/

[6] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, Jun. 2012, pp. 323–336.

[7] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "ACCessory: Password inference using accelerometers on smartphones," in *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, Feb. 2012, Art. no. 9.

[8] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Security Privacy Wireless Mobile Netw.*, Apr. 2012, pp. 113–124.

[9] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. 4th USENIX Conf. Offensive Technol.*, Aug. 2010, pp. 1–7.

[10] F. Ben Abdesslem, A. Phillips, and T. Henderson, "Less is more: Energy-efficient mobile sensing with senseless," in *Proc. 1st ACM Workshop Netw. Syst. Appl. Mobile Handhelds*, 2009, pp. 61–62.

[11] L. Duan, N. Li, and L. Huang, "A new spam short message classification," in *Proc. 1st Int. Workshop Educ. Technol. Comput. Sci.*, 2009, pp. 168–171.

[12] L. Ma, T. Tan, Y. Wang, and D. Zhang, "Personal identification based on iris texture analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1519–1533, Dec. 2003.

[13] L. Deng *et al.*, "Deep learning: Methods and applications," *Found. Trends® Signal Process.*, vol. 7, no. 3/4, pp. 197–387, 2014.

[14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[15] P. Le Callet, C. Viard-Gaudin, and D. Barba, "A convolutional neural network approach for objective video quality assessment," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1316–1327, Sep. 2006.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[19] C. Bo, L. Zhang, T. Jung, J. Han, X.-Y. Li, and Y. Wang, "Continuous user identification via touch and movement behavioral biometrics," in *Proc. IEEE 33rd Int. Perform. Comput. Commun. Conf.*, 2014, pp. 1–8.

[20] S. J. Alghamdi and L. A. Elrefaei, "Dynamic user verification using touch keystroke based on medians vector proximity," in *Proc. 7th Int. Conf. Comput. Intell. Commun. Syst. Netw.*, 2015, pp. 121–126.

[21] R. Murmuria, A. Stavrou, D. Barbará, and D. Fleck, "Continuous authentication on mobile devices using power consumption, touch gestures and physical movement of users," in *Proc. Int. Workshop Recent Advances Intrusion Detection*, 2015, pp. 405–424.

[22] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, pp. 121–167, 1998.

[23] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.

[24] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

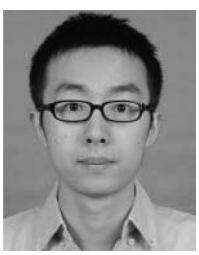
[25] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine learning based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[26] A. K. Menon, H. Narasimhan, S. Agarwal, and S. Chawla, "On the statistical consistency of algorithms for binary classification under class imbalance," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 603–611.

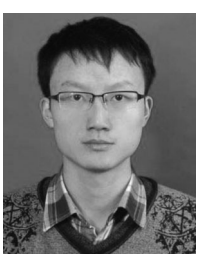
[27] K. Kennedy, B. Mac Namee, and S. J. Delany, "Learning without default: A study of one-class classification and the low-default portfolio problem," in *Proc. Irish Conf. Artif. Intell. Cognitive Sci.*, 2009, pp. 174–187.

[28] A. Goh and D. C. Ngo, "Computation of cryptographic keys from face biometrics," in *Proc. IFIP Int. Conf. Commun. Multimedia Security*, 2003, pp. 1–13.
[29] F. Prokoski, "History, current status, and future of infrared identification," in *Proc. IEEE Workshop Comput. Vis. Beyond Visible Spectrum: Methods Appl.*, 2000, pp. 5–14.
[30] D. de Martin-Roche, C. Sanchez-Avila, and R. Sanchez-Reillo, "Iris recognition for biometric identification using dyadic wavelet transform zero-crossing," in *Proc. IEEE 35th Int. Carnahan Conf. Security Technol.*, 2001, pp. 272–277.
[31] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics: A survey," *J. Pattern Recognit. Res.*, vol. 7, no. 1, pp. 116–139, 2012.
[32] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *Proc. Int. Joint Conf. Biometrics*, 2011, pp. 1–8.
[33] K. Niinuma, U. Park, and A. K. Jain, "Soft biometric traits for continuous user authentication," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 771–780, Dec. 2010.
[34] C. Bo, X. Jian, X.-Y. Li, X. Mao, Y. Wang, and F. Li, "You're driving and texting: Detecting drivers using personal smart phones by leveraging inertial sensors," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 199–202.
[35] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, 2014, pp. 221–232.
[36] T. Feng *et al.*, "Continuous mobile authentication using touchscreen gestures," in *Proc. IEEE Conf. Technol. Homeland Security*, 2012, pp. 451–456.
[37] X. Zhao, T. Feng, and W. Shi, "Continuous mobile authentication using a novel graphic touch gesture feature," in *Proc. IEEE 6th Int. Conf. Biometrics: Theory Appl. Syst.*, 2013, pp. 1–6.
[38] Y. Song, Z. Cai, and Z.-L. Zhang, "Multi-touch authentication using hand geometry and behavioral information," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 357–372.
[39] Z. Sitová *et al.*, "HMOG: New behavioral biometric features for continuous authentication of smartphone users," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 877–892, May 2016.

Lichao Sun is currently working toward the PhD degree at the University of Illinois at Chicago, Chicago, Illinois. His research interests include deep learning and data mining. He has published more than 15 research articles in the areas of security and privacy, social network, and natural language processing applications.



Bokai Cao received the PhD degree in computer science from the University of Illinois at Chicago, Chicago, Illinois. He is currently a research scientist at Facebook. His research interests invlude data mining and machine learning. In particular, he focuses on the development and analysis of algorithms for brain, social and information networks, as well as modeling feature interactions and broad learning for healthcare.



Ji Wang received the PhD degree in information system from the National University of Defense Technology, Changsha, China, in 2019. He is currently an assistant professor with the College of Systems Engineering, National University of Defense Technology, Changsha, China. His research interests include deep learning and edge intelligence. He has published more than 20 research articles in refereed journals and conference proceedings, such as the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel & Distributed Systems*, SIGKDD, and AAAI. He was a visiting PhD student at the University of Illinois at Chicago, Chicago, Illinois from March 2017 to September 2018 under the supervision of Prof. Philip S. Yu.

Witawas Srisa-an received the PhD degree in computer science from the Illinois Institute of Technology, Chicago, Illinois, in 2002. He is currently an associate professor of computer science and engineering at the University of Nebraska-Lincoln, Lincoln, Nebraska. His research interest spans the areas of programming languages, software engineering, operating systems, runtime systems, and security.

Philip S. Yu received the BS degree in EE from National Taiwan University, Taipei, Taiwan, the MS and PhD degrees in EE from Stanford University, Stanford, California, and the MBA degree from New York University, New York. He is currently a distinguished professor in computer science at the University of Illinois at Chicago, Chicago, Illinois and also holds the Wexler chair in information technology. Before joining UIC, he was with IBM, where he was manager of the Software Tools and Techniques Department, Watson Research Center. His research interest include big data, including data mining, data stream, database and privacy. He has published more than 1,200 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. He is a fellow of the ACM and IEEE. He is the recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion and anonymization of big data, the IEEE Computer Societys 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining and anonymization of big data, and the Research Contributions Award from IEEE Intl. Conference on Data Mining (ICDM), in 2003 for his pioneering contributions to the field of data mining. He also received the ICDM 2013 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award (2014). He was the editor-in-chief of the *ACM Transactions on Knowledge Discovery from Data* (2011-2017) and *IEEE Transactions on Knowledge and Data Engineering* (2001-2004).

Alex D. Leow is a tenured associate professor with the Departments of Psychiatry, Bioengineering, and Computer Science, University of Illinois at Chicago (UIC), Chicago, Illinois and an attending physician with the University of Illinois Hospital. She co-founded the Computational Neuroimaging and Connected Technologies (CoNeCT) Lab at UIC, in 2015. With both a doctoral degree in applied mathematics and a medical board certification in adult psychiatry, her research efforts have included both the development of computational methods and their translational applications in neuroimaging and, most recently, mobile technology. In addition, clinically she has personally evaluated, diagnosed, treated, and followed thousands of individuals with complex neuropsychiatric conditions.

Stephen Checkoway is currently an assistant professor with the Department of Computer Science, Oberlin College, Oberlin, Ohio. He researches the security of computer systems including embedded and cyberphysical systems. His work includes the first comprehensive analysis of automotive computers, as well as the first independent analysis of X-ray backscatter, full-body scanners formerly deployed in airports.