BANN-TMGuard: Towards Touch Movement-Based Screen Unlock Patterns via Blockchain-enabled Artificial Neural Networks on IoT Devices

Meng, Weizhi; Li, Wenjuan; Calugar, Andrei Nicolae

Published in:
Ieee Internet of Things Journal

Link to article, DOI:
10.1109/JIOT.2024.3465891

Publication date:
2025

Document Version
Peer reviewed version

BANN-TMGuard: Towards Touch Movement-Based Screen Unlock Patterns via Blockchain-enabled Artificial Neural Networks on IoT Devices

Weizhi Meng, *Senior Member, IEEE*, Wenjuan Li, *Senior Member, IEEE*, and Andrei Nicolae Calugar

***Abstract*—Internet of Things (IoT) devices such as smartphones have become important to people's everyday usage, especially the number of smartphone shipment has surpassed six billion and is forecast to further grow. The smartphone security is the top priority as people may store various sensitive information on these devices. Currently, phone unlock patterns, e.g., Android unlock patterns, are one of the main protection methods to protect smartphones from unauthorized access. However, many research studies have revealed that cyber-attackers can easily compromise this type of unlock mechanism, i.e., learning the pattern from the touch residue. In this work, we advocate that an additional security layer should be added to enhance the security of Android unlock patterns, and thus develop a touch movement-based unlock mechanism via blockchain-enabled artificial neural networks (ANNs), named *BANN-TMGuard*, which can examine the biometric features of a user's touch movement as well as the input pattern. Further, *BANN-TMGuard* adopts blockchain technology to secure the robustness and reliability when building the ANN models. In the evaluation, we perform a user study with 100 participants in the aspects of authentication accuracy, time consumption and user feedback. As compared with similar schemes, our *BANN-TMGuard* demonstrates better results and is preferred by most participants in the user study.**

***Index Terms*—Smartphone Security, Smartphone lock, Blockchain technology, User authentication, Touch movement.**

I. Introduction

CONSUMER Consumer electronic devices and Internet-of Things (IoT) devices have become essential to an individual's daily lives, including computer, cell phone, game consoles, and television. The global consumer electronics market size was 729.11 billion USD in 2019 and was predicted to reach 989.37 billion USD in 2027, according to Fortune Business Insights [1]. Among these, smartphone is one of the most widely used IoT devices. According to a report from the International Data Corporation (IDC) [2], smartphone shipment was obviously affected by COVID situation, but still reached 301.9 million units in the third quarter of 2022. In 2026, the shipment was expected to go up to 1.53 billion units.

Due to the increasing capability and the wide adoption of smartphones, in addition to financial applications (e.g., bank App [27]), more individuals are likely to store personal or even sensitive information on their smart devices (e.g., pictures of credit card and personal ID card [23], [25]), which may become a major target for cyber attackers. If the phone is lost, an unauthorized person can access the stored information and use the phone maliciously. In this case, it is very important to enforce proper user authentication.

Currently, password-based user authentication, such as PIN (Personal Identity Number) code, is the most widely adopted method on smartphones, where users have to type the correct textual information to unlock the phone. However, such kind of authentication has been pointed out several security and usability limitations [9]. It is known that users have difficulty remembering a long and complex string for a long time, due to the long-term memory (LTM) limitations [32]. For this sake, many users may choose a simple and easy-to-remember textual passwords instead, i.e., the most used string is "123456" [3]. Also, textual passwords could be easily obtained by attackers via various exploits and attacks, such as phone charging attacks [30] and recording attacks [20]. The former can record the phone screen during the phone charging period (side channel), while the latter can record the phone screen by installing a malicious application (malware).

To complement the drawbacks of password-based authentication, graphical password-based user authentication has been proposed, in which users can interact with images for authentication. This is because human brain was believed to be better at remembering graphical information than textual information [33]. Android unlock patterns are one commonly deployed authentication means on smartphones, which require users to input a pattern within a $3X3$ touch-enabled grid. To create an unlock pattern, users can swipe the finger to select at least 4 dots and at most 9 dots.

Motivations. Face recognition is another popular unlocking method on phones, but due to the privacy concerns, some users may not prefer such method. For example, the stored facial features can be accessed by attackers [12]. As long as the face information has been leaked, it is hard to change a new one. Hence many users may choose Android unlock patterns to unlock their phones.

Though Android unlock patterns can greatly enhance the memorability and usability during user authentication, it is susceptible to many security threats. As an example, brute-force attack is still feasible, as users can only select a maximum of 9 dots in the touch-enabled grid, resulting in a pattern space of 389,112 [11]. In addition, smudge attack that can recover touch trails by adjusting lightning and photographic

W. Meng and A.N. Calugar are with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark. E-mail: weme@dtu.dk (Corresponding)

W. Li is with the Department of Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong SAR, China

setup, which can greatly decrease the possible patterns of a user's credential [8]. Hence, it is very demanding to enhance the security of Android unlock patterns in practical usage.

Contributions. To achieve the security improvement of unlock patterns, implementing an additional security layer is important and essential. For example, De Luca *et al.* [13] proposed to involve biometric features when inputting a pattern. In this work, we advocate that verifying behaviorial features can enhance the security of Android unlock patterns, and develop a touch movement-based scheme via artificial neural networks and blockchain technology. Our contributions can be summarized as follows.

- We design *BANN-TMGuard*, a touch movement-based unlock pattern scheme using blocchain-enabled artificial neural networks (ANNs), which considers various touch-related features during authentication, such as touch speed, angle, touch pressure, touch size and touch acceleration.
- Sharing the data is crucial for building an accurate and robust ANN model, but it may pose a hole for attackers to inject malicious samples. For this reason, we adopt blockchain technology for the data sharing process, and reach a more robust ANN model under adversarial scenarios.
- In the evaluation, we perform a user study with a total of 100 participants, and test the robustness of ANN model under an adversarial scenario. The study results demonstrate that our proposed *BANN-TMGuard* can provide better performance and usability, and ensure the robustness of ANN model against malicious input.

It is worth noting that this work extends our previous study - *TMGuard* [29], but there are many differences. First, *TMGuard* uses a statistical method with dot-dot pattern computation and proportional matching, while *BANN-TMGuard* adopts ANNs, a machine learning-based approach for authentication. Second, *BANN-TMGuard* considers more touch-related features than *TMGuard*, such as touch acceleration. Third, we compare the performance of *BANN-TMGuard* with two similar schemes in our user study. Further, our *BANN-TMGuard* adopts blockchain technology, which can provide a more robust and reliable ANN model against malicious input, and ensure the integrity of data sharing.

Roadmap. The rest of this article is structured as follows. Section II presents the background on Android unlock patterns and introduces related work on biometrics-enhanced unlock patterns. Section III introduces our proposed BANN-TMGuard with different components. Section IV presents our user study by considering several similar schemes in the comparison. Section V discusses open challenges and future directions. Finally, we conclude this work in Section VI.

II. Background and Related Work

This section will briefly introduce the background on Android unlock patterns and related work on biometrics-based unlock mechanisms.

A. Background on Android Unlock Patterns

Followed the main idea of Pass-Go [34], Android unlock patterns have been designed, which allows phone users to unlock their smartphones by swiping their finger to create an unlock pattern on the touchscreen. More specifically, a pattern can be generated by means of 4 dots at least and 9 dots at most, within a $3X3$ grid on the touchscreen. Figure 1 shows two examples of Android unlock patterns created by Berkeley Churchill - a pattern generator [4]: Figure 1(a) shows a 4-dot pattern and Figure 1(b) presents a 9-dot pattern.

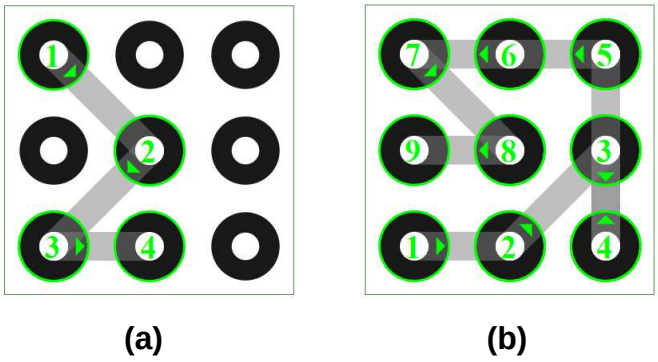

Fig. 1. Pattern Examples generated by Berkeley Churchill: (a) 4-dot pattern, and (b) 9-dot pattern. It is a JavaScript gesture generator for the unlock pattern on Android phones. Users can choose whether to avoid "Hard-to-enter" patterns.

The input pattern will be converted to byte array and then get encrypted via a hash function. The password file is stored as *gesture.key* on the phone. A similar Android unlock mechanism can be used in a jailbroken iPhone as well, such as *LockDroid* [5] - an Android-inspired pattern lock for iOS.

There are some concrete rules for creating a valid Android unlock pattern. (i) A valid pattern cannot select a dot more than once, as the selected dot is considered to be virtually removed. (ii) At least 4 dots must be selected for a valid pattern and only straight touch-lines can be used. (iii) It is not possible to not select the middle dot, when creating a touch-line with three dots, unless the latter has been previously visited.

By enumerating all possible patterns based on the above rules, Aviv *et al.* [8] figured out that there should be 389,112 (2^{19}) possible patterns. Ideally, the pattern space is sufficient if users can create the patterns uniformly, but the actual pattern space is much less in practical usage, i.e., the security is even worse than a 3-digit PIN [36].

Adversarial scenarios. The Android unlock pattern needs a user to start a pattern by touching one dot, which may be vulnerable to 'hot-dot'. Andriotis *et al.* [7] learned from a study that users may have bias on selecting the *start point* and *end point*, i.e., more than 52% users may start a pattern from the top-left dot, which would greatly reduce the search space of possible patterns. Also, smudge attacks [8] could be used to recover the unlock patterns as users will leave the oily residues on the touchscreen when drawing the patterns with finger. In this case, attackers can even just take a photo for the touchscreen and figure out the pattern by adjusting brightness. Botelho [11] studied how to brute-force a 4-dot unlock pattern, and found that less than 4 minutes are required to compromise

TABLE I
FEATURE COMPARISON AMONG EXISTING BIOMETRIC-BASED UNLOCKING SOLUTIONS.

Scheme	Biometric Type	Continuous Verification	Credential Replaceable	Data Integrity Insurance
Pass-O [35]	Patterns	No	Yes	No
Double Patterns [14]	Patterns	No	Yes	No
De Luca *et al.* [13]	Patterns / Touch	Yes	Yes	No
Zhao *et al.* [40]	Patterns / Touch	Yes	Yes	No
OpenSesame [15]	Patterns / Touch	Yes	Yes	No
Zheng *et al.* [41]	Passcode / Tap	No	Yes	No
Izuta *et al.* [18]	Taking Behavior	No	Yes	No
WearLock [39]	Acoustic	Yes	Yes	No
Wang *et al.* [37]	Heartbeat	Yes	Yes	No
SwipeVlock [21], [22]	Image / Touch	No	Yes	No
Double-X [25]	Patterns / Touch	Yes	Yes	No
Gleerup *et al.* [16]	Zoom action	Yes	Yes	No
Our Method	Patterns / Touch	Yes	Yes	Yes

a 4-dot pattern.

B. Biometric-based Unlock Mechanism

To enhance the security of Android unlock mechanism, one direct method is to increase the pattern complexity. For example, Tupsamudre *et al.* [35] changed the dot-layout to circular layout and proposed an unlock scheme called *Pass-O*, with a theoretical search space of 9,85,824 possible patterns. Forman and Aviv [14] then introduced *Double Patterns*, where users have to create and input two unlock patterns for authentication. These methods can increase the search space of possible patterns; however, they may also increase the burden on user side, i.e., users have to remember new rules or create more patterns.

Another important solution is to integrate behavioral biometrics, the features extracted from human actions [10]. The main idea is to authenticate users in terms of which pattern has been input and how the pattern was input. De Luca *et al.* [13] argued the need of an additional security layer for Android unlock patterns, and introduced an implicit approach by combining unlock mechanism with biometric features, such as touch pressure and size. They also proved the feasibility using a dynamic time warping (DTW) method. Zhao *et al.* [40] analyzed the Graphic Touch Gesture Feature (GTGF) for touch behavioral authentication and tried several touch actions such as flick up/down, flick right/left, zoom in/out.

Guo *et al.* [15] designed *OpenSesame*, a smartphone unlock mechanism that verifies users based on their shaking actions, based on users' habits and shaking action. With a support vector machine (SVM) classifier, OpenSesame could reach a false negative rate of 11% and a false positive rate of 15% under 200 participants. Zheng *et al.* [41] explored an approach of user verification by checking users' tapping actions when inputting a Passcode. In the evaluation, their approach could reach an averaged equal error rate of around 3.65%. Izuta *et al.* [18] presented a phone unlock solution, by examining how the user takes the phone out of the pocket, especially the touch pressure distribution. They scheme achieved a false acceptance rate of 43% with only 18 training instances. Meng *et al.* [29] introduced *TMGuard*, a touch movement-based unlock scheme to enhance the security of Android unlock patterns, by verifying how the user input the unlock pattern. Their scheme developed two matching methods called dot-dot pattern computation and proportional matching. In the study, the top user could achieve an authentication accuracy of 96%.

A phone unlock scheme of *WearLock* was proposed by Yi *et al.* [39], which could unlock the device based on acoustic tones via smartwatch. They made an assumption that the wireless link would be secure in-between that can be utilized as a control channel. In the evaluation, their scheme could reach a low average bit error rate of 8%. Wang *et al.* [37] then introduced an unlock scheme based on heartbeat signals on smartphones with more resources. Users can unlock the device by pressing the phone on the chest, then the device would collect a few heartbeat signals and complete the authentication. In the study, they included 35 users and the results provided an Equal Error Rate (EER) of 3.51% with only 5 heartbeat signals. Then Li *et al.* [21], [22] developed a phone unlock solution called *SwipeVlock*. Users can unlock the device by selecting a site on a selected image and performing a swiping. Their scheme could reach an authentication accuracy of 98% in the best scenario.

Li *et al.* [24] introduced a type of unlock scheme based on simple shapes, such as rectangle, square, circle, triangle and diamond. Users can draw two or three of them, and the scheme would check their behavioral features during authentication. In the evaluation, they identified that two-shape scheme was preferred by most participants. Then *Double-X* [25], a double-cross-based phone unlock solution was developed, which verified how users input two cross shapes on two dots on Android unlock patterns. With a study of 85 users, they found that the scheme could reach an accuracy rate of 95%. Alawami *et al.* [6] presented a hybrid unlock scheme by means of Wi-Fi signals and the light intensity of phone's fingerprints. That is, it utilizes the location information to judge whether the device can be unlocked. Gleerup *et al.* [16] introduced a phone unlock scheme by examining users' zoom actions such as zoom-in or zoom-out. It is a two-step scheme that requires a user to select two dots and perform two zoom actions. Some more relevant studies can refer to [17], [19], [26], [38].

Discussion. User authentication via behavioral biometrics is an important and transparent security solution against unauthorized access [28]. Android unlock patterns are the most

popular unlock scheme on Android phones, but may suffer many security issues, such as a) Smudge Attacks [8] where attackers can extract touch trails from fingerprint smudges, b) brute-force attacks as the potential pattern number is till small, and c) side channel attacks like phone recording attacks.

Table I summarizes the feature comparison of existing unlocking methods in the literature. It is found most current studies did not consider an adversary environment, so the data integrity cannot be secured. In this work, motivated by this line of research, we focus on enhancing Android unlock patterns with behavioral features. In particular, we developed *BANN-TMGuard*, a touch movement-based unlock scheme using ANNs. Our work aims to stimulate more research towards investigating the design of more secure and robust unlock schemes on IoT devices.

III. Our Proposed Approach

This section introduces the architecture of *BANN-TMGuard* (including data recording, pattern extraction, feature extraction, pattern comparison, ANN-based classification, and decision component), touch features (including related to touch movement, touch pressure, touch size, touch duration and acceleration) and our ANN-based approach with the implementation details.

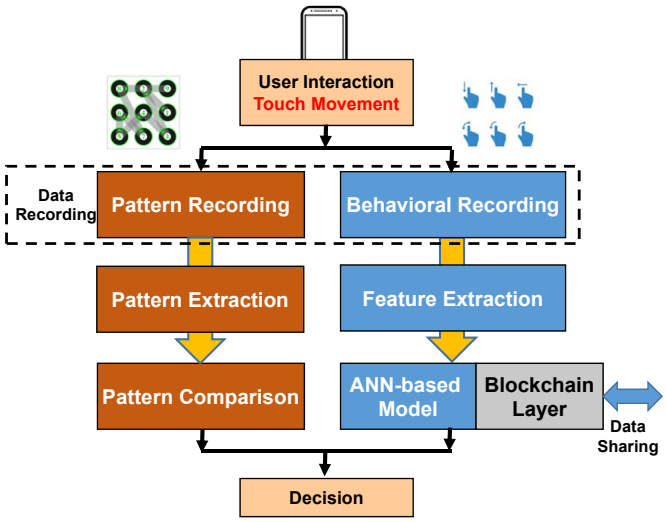

Fig. 2. The high-level architecture of BANN-TMGuard.

A. BANN-TMGuard

Based on the observations in the previous work [29], users would perform a touch movement differently when drawing an unlock pattern, while the touch action would become more stable after several trials. Hence, it is a promising scenario of combining Android unlock patterns with touch behavioral authentication. In this work, we develop *BANN-TMGuard*, a touch movement-based unlock mechanism via ANN.

Figure 2 illustrates the high-level architecture of our *BANN-TMGuard*, including data recording, pattern extraction, feature extraction, pattern comparison, ANN-based classification, and decision component.

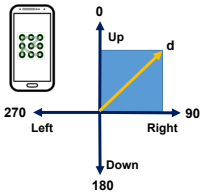

Fig. 3. Directions of a touch movement.

- *Data Recording.* This component consists of pattern recording and behavioral recording, which is responsible for recording the input the unlock pattern and raw data from touch movement actions, such as touch coordinates, system time, pressure, size, etc.
- *Pattern Extraction.* This component aims to extract the input unlock pattern. This is a conventional process of Android unlock mechanism.
- *Feature Extraction.* This component aims to calculate the touch behaviorial features, such as the speed and angle of touch movement based on the collected raw data.
- *Pattern Comparison.* This component can compare the input unlock pattern (after hashing) with the hashed pattern stored in the database, and then report whether there is a match to decision component.
- *ANN-based Model.* This component aims to examine the features of a user's touch movement action. It should first build a normal profile of users via artificial neural networks and then use the ANN model to authenticate users. The result will be reported to the *decision component*.
- *Blockchain Layer.* This layer is responsible for examining the integrity of shared data, e.g., training data or items, which can be used to initialize the accuracy of ANN models. This is especially useful when the training data is lacking and when optimizing the hyperparameters of an ANN model.
- *Decision Component.* This main task of this component is to gather the decision from both *pattern comparison* and *ANN-based classification*, and then make the final decision regarding the legitimacy of current user. A successful login requires both correct pattern and acceptable touch behavior.

B. Touch Feature

The previous study of *TMGuard* employed a statistical method so that only the touch speed and angle were considered. By given a threshold, *TMGuard* can judge whether the current touch behavior is normal.

In this work, we adopt a neural network classifier to help model a user's touch movement action. Thus, we can use more touch features. First, we adopt the same features to describe a touch movement: ***the speed of touch movement*** (*STM*) and ***the angle of touch movement*** (*ATM*). Figure 3 defines the directions of a touch movement on a 2D grid. Suppose two dots *D1* and *D2* with coordinates (x1, y1) and (x2, y2) are selected in an unlock pattern, and the corresponding system time is *S1* and *S2*, respectively. We can compute *STM* and *ATM* as below.

$$STM = \frac{\sqrt{(x2-x1)^2+(y2-y1)^2}}{S2-S1} \tag{1}$$

$$ATM \quad (d) = \arctan\frac{y2-y1}{x2-x1}, d \in [0, 360^\circ] \tag{2}$$

where d is the current touch angle. In this work, we can model the touch movement between any connecting dots in a pattern. It is similar to the *dot-dot pattern computation* in *TMGuard*. The main purpose is to build up a more accurate behavioral model. Hence, we can have:

$$Profile = \{\bigcup^{j}(STM,\ ATM)_i\}\ \ (4 \leq j \leq 9; i = 1, ..., j-1) \tag{3}$$

where j is the number of selected dots in a pattern, and i is the number of pairs of touch speed and angle. Below are more touch features including touch pressure, size, duration, and acceleration.

Touch pressure. This is the amount of pressure being sensed from a touch. Various research studies has indicated that many phones with built-in sensors are can record the values of touch pressure, which can be used to distinguish touch actions, i.e., some users may press more heavily on the screen than others.

Touch size. This feature measures the average size of a human finger pad on the touchscreen, which can be used to distinguish touch actions. For example, some users may have a bigger touch size than others.

Touch duration. This feature measures the time difference between a touch press-down even and a subsequent touch press-up event. It is a typical touch feature, which can be used to distinguish users. For instance, some users may touch the screen shorter than others.

Touch acceleration. Our scheme consider three vectors of touch acceleration: (a) the magnitude of acceleration when the touch is pressed down; (b) the magnitude of acceleration when the touch is released; and (c) the average value of magnitude of acceleration during touch-press to touch-release.

C. ANN-based Approach

Artificial neural networks (ANNs), inspired by the biological connected units, are an important tool in various domains including user authentication. In this work, we thus devise an ANN-based approach in our *BANN-TMGuard*, which involves a hyperparameter tuning process and several variations of ANNs, such as deep neural network, recurrent neural network, and convolutional neural network.

1) Deep Neural Network (DNN): DNN can be treated as a stacked neural network, which is composed of several layers. Generally, it can be much more computationally demanding than a simple neural network. This is not given only by higher number of neurons but also the fact that these neurons may influence the computation by triggering different actions.

Mathematically such neural network can be defined as

$$O : \mathbf{R}^m \times \mathbf{R}^n$$

with m means the size of the input vector $x = x_1, x_2, ..., x_m$ and n is the size of the output vector $y = y_1, y_2, ..., y_n$. We can define the computation of all the hidden layers h_i as:

$$h_i(x) = f(w_i^T x + b_i)$$

where :

$$h_i : \mathbf{R}^{d_i - 1} \rightarrow \mathbf{R}^{d_i}$$

$$f : \mathbf{R} \rightarrow \mathbf{R}$$

$$w_i \in \mathbf{R}^{d \times d_{i-1}}$$

$$b \in \mathbf{R}^{d_i},$$

where d_i denotes the size of the input, f is the nonlinear activation function that is either sigmoid or softmax. A multi-layer neural network can be defined as follows:

$$H(x) = H_i(H_{i-1}(H_{i-2}(...(H_1(x))))) \tag{4}$$

2) Convolutional Neural Network (CNN): CNN is another type of ANNs by extending traditional feed-forward neural networks. They are typically used as a model to process data with a grid pattern such as an image.

A conventional CNN requires a three-dimensional input (x rows, y columns, z depths). For example an image that has x width , y height and z channels. In this work, we mainly study the one-dimensional input data as a three-dimensional picture, which has one channel with the height value of one. The architecture of a CNN is comprised of several building blocks, such as: convolution layer, pooling layers and fully connected layers.

The convolution output is transferred through a nonlinear activation function. The most common one being used in current scientific research is the rectified linear unit or ReLU. Next, a pooling layer is used for the down-sampling operation. The advantage of using this operation is that it reduces the dimensionality of the feature map without losing important information. In practice, max-pooling has shown to be the most commonly used technique.

The final layer is the fully connected layer, which is in charge of classification. The other layers' output is flattened into a one-dimensional array of numbers connected to one or multiple fully connected layers, also called dense layers. Each fully connected layer has a nonlinear function such as ReLU, and the output layer has a Softmax activation function used for multiclass classification.

3) Recurrent Neural Network (RNN): RNN is a type of neural network that looks at the current input and considers the past state information to determine its output. Considering x of elements $x_1, x_2, ..., x_t$ as an input that maps them to hidden and output elements h of elements $h_1, h_2, ..., h_t$. For each time step, an input x_0 goes into the network and produces an output h_0. In the next step, the input x_1 is taken into block A and additional input from the previous block. Hence the neural network considers the context of the previous input to determine the next results.

Taking a closer look at a RNN unit, we find that Equations (5) and (6) can govern its computation at any time t, with the

```
...
#Create Sequential model with Dense layers, using the add method
dnn3= Sequential()

dnn3.add(Dense(512,activation='relu', input_shape=(features_train.shape[1],)))
dnn3.add(Dropout(0.1))
dnn3.add(Dense(256, activation='relu'))
dnn3.add(Dropout(0.1))
dnn3.add(Dense(128, activation='relu'))
dnn3.add(Dropout(0.1))
dnn3.add(Dense(15, activation = 'softmax'))
...
```

Fig. 4. DNN with three hidden layers.

activation function $g()$, X input , H output, b the bias, and W the weight matrix.

$$A_t = g \cdot (W_{ax} \cdot X_t + W_{aa} \cdot A_{t-1} + b_a) \quad (5)$$

$$H_t = g \cdot (W_{ya} \cdot A_t + b_y) \quad (6)$$

D. ANN Implementation

Keras library[1] is used to represent the actual neural network model through the use of Sequential API. The fundamental concept of Sequential API is to arrange the Keras layers one after another. Thus, the data flows from one layer to the next before it hits the output layer.

DNN. The implementation of DNN with three hidden layers is presented in Figure 4. The $Dense$ class is used to describe fully connected layers. The layer's number of neurons can be specified as the first argument, the activation function can be specified as the second argument, and the last argument refers to the number of input features. Regularization is also applied between each layer through the use of $Dropout$, which aims to avoid overfitting and accelerates the training of the DNN model by removing neurons and their connections randomly. The last layer represents the output layer and uses $softmax$ activation function for multi-class classification.

CNN. As shown in Figure 5, this work explores one Convolutional Layer, which anticipates a three-dimensional input. The first dimension is regarded as the number of samples in the input; in this example, there is just one sample. The second dimension is the length of each sample representing the number of features, which is 44 in this case. The third dimension represents the number of channels present in each sample; there is only one channel in this case. Therefore the model will expect an input sample to have the shape $[44, 1]$ together with a filter or kernel size of size 3 and an additional $ReLU$ activation function. Spatial pooling, also known as subsampling or downsampling, decreases the dimension of each function map while retaining the most critical detail. In our case a $2X2$ window of $MaxPooling1D$ is used.

RNN. Figure 6 presents the architecture of an RNN with two $SimpleRNN$ layers. Based on the Keras library, some of the parameters used in the $SimpleRNN$ are the number of units representing the dimensionality of the output space, the activation function together with the input shape, and the return sequence. In our case, the activation function is $ReLU$ with 32 units, and return sequence is set to true. $Dropout$ is used after each $SimpleRNN$ layer for regularization purposes, followed by flattening the output and adding additional

[1]https://keras.io/

```
...
# define input data
data = data.reshape(1, 44, 1)

# create model
cnn3 = Sequential()

cnn3.add(Conv1D(64, 8, activation='relu', input_shape=(44, 1)))
cnn3.add(MaxPooling1D((2, 2)))
cnn3.add(Dropout(0.3))

cnn3.add(Conv1D(32, 8, activation='relu'))
cnn3.add(MaxPooling1D((2, 2)))
cnn3.add(Dropout(0.4))

cnn3.add(Conv1D(32, 8),activation='relu')
cnn3.add(Dropout(0.4))

cnn3.add(Flatten())
cnn3.add(Dense(64,activation='relu'))

cnn3.add(Dense(5, activation='softmax'))
...
```

Fig. 5. CNN with three hidden layers.

```
...
# create model
rnn3 = Sequential()

# Adding the first RNN layer and some Dropout regularisation
rnn3.add(SimpleRNN(units = 32,activation='relu', return_sequences = True,
    input_shape = (77,1)))
rnn3.add(Dropout(0.1))

# Adding a second RNN layer and some Dropout regularisation
rnn3.add(SimpleRNN(units = 32,activation='relu', return_sequences = True))
rnn3.add(Dropout(0.1))

# Adding the Flatten layer
rnn3.add(Flatten())
rnn3.add(Dense(32, input_dim=77))
rnn3.add(LeakyReLU(alpha=0.1))
rnn3.add(Dense(16))
rnn3.add(LeakyReLU(alpha=0.1))

# Adding the output layer
rnn3.add(Dense(units = 15, activation='softmax'))
...
```

Fig. 6. RNN with two hidden layers.

dense layers with $LeakyReLU$ as the activation function a version of $ReLU$ that allows a smaller gradient when the unit is not active.

Hyperparameter tuning process. It can be observed that that tuner search loop runs each time with a different hyperparameter choice. The training is done with some set parameters on each iteration, after which some evaluation metrics are computed. Finally, after all, possible combinations have been computed, the models with the best results can be chosen. This process is detailed in Figure 7, and it is qualified as a grid search. In the beginning, data items can be obtained from other peers or cloud services provided by the vendor in order to optimize the hyperparameters.

IV. Evaluation

In this section, we introduce a user study aiming to explore the scheme performance, and analyze the obtained results and users' feedback. Also, we test the robustness of our blockchain-based ANN models under malicious input.

A. Use Study

To study the performance of *BANN-TMGuard*, we conducted an IRB-approved user study involving 100 regular smartphone users, including 70% Android phone users and 30% iPhone & Android phone users (those who are using two smartphones). Table II presents the background of all participants. More specifically, we have 55 males and 45

```
cnn_dense_layers =[ 1, 2, 3,]
cnn_conv_layers = [ 1 , 2 ,3 ]
cnn_layer_sizes  = [64, 128 ]
# Build model for each possible value
for cnn_dense_layer in cnn_dense_layers:
    for cnn_layer_size in cnn_layer_sizes :
        for cnn_conv_layer in cnn_conv_layers :
            model = Sequential()
            model.add(Conv1D(cnn_layer_size ,3,input_shape=(78, 1)))
            cnn3.add(MaxPooling1D((2, 2)))
            model.add(LeakyReLU(alpha=0.1))
            model.add(Dropout(0.4))
        for l in range(cnn_conv_layer - 1):
            model.add(Conv1D(cnn_layer_size ,3))
            cnn3.add(MaxPooling1D((2, 2)))
            model.add(LeakyReLU(alpha=0.1))
            model.add(Dropout(0.4))
        model.add(Flatten())
        for l in range(cnn_dense_layer):
            model.add(Dense(cnn_layer_size, input_dim=78)
            model.add(LeakyReLU(alpha=0.1))

        model.add(Dense(15, activation='softmax'))
        model.compile(loss ='categorical_crossentropy', optimizer= 'adam',
            metrics = ['accuracy'])
        model.fit(features_train,labels_train,epochs=25,batch_size = 1000)
        # Print model
        print(NAME ="Number of conv layers-{}-Number of nodes-{}-number of
            dense layers-{}.".format(cnn_conv_layer, cnn_layer_size,
            cnn_dense_layer))
        # Evaluating model accuracy.
        model.evaluate(features_test , labels_test)
```

Fig. 7. Grid search tuning method.

females who aged from 21 to 60, including bachelor, master and doctoral students, restaurant staff, university technical staff, and university academic members. Each participant can get a $50 gift voucher after the user study.

TABLE II
PARTICIPANTS' INFORMATION IN THE USER STUDY.

Information	Male	Female	Occupation	Male	Female
Age 21-30	38	29	Students	45	36
Age 30-40	13	11	Faculty&Staff	7	6
Age 40-50	4	5	Restaurant Staff	3	3

Data Collection. Similar to *TMGuard* [29], we employed the same data collection method to facilitate the comparison: namely a modified Google/HTC Nexus One Android phone with a capacitive touchscreen (480×800 px). The phone was reflashed with a modified Android Operating System (OS) based on *CyanogenMod*[2]. The changes are mainly made on the application framework layer to collect and record raw data from the screen, such as touch type (e.g., touch press down), touch coordinates of x and y, and event time. It is worth noting that our scheme is scalable to other phone types as long as the device can collect the data to calculate the touch features.

Figure 8(a) shows the screen logo of our modified Android OS, Figure 8(b) presents the layout of Android unlock patterns, and Figure 8(c) provides an example of raw data.

B. Study Procedure

To better evaluate the performance, we compared *BANN-TMGuard* with two relevant schemes: *DeLucaUnLock* [13] and *TMGuard* [29]. As aforementioned, DeLucaUnLock scheme authenticates a user's touch behavioral features, e.g., touch pressure, size, when they draw Android unlock patterns, based on dynamic time warping (DTW). *TMGuard* authenticates a user's touch movement when drawing the unlock pattern via a statistical method, called proportional matching.

[2]http://www.cyanogenmod.com/.

TABLE III
SUCCESS RATE IN THE CONFIRMATION, LOGIN AND RETENTION PHASE FOR GROUP-1.

Phase	DeLucaUnLock	BANN-TMGuard
Confirmation	472/500 (94.4%)	481/500 (96.2%)
Login	469/500 (93.8%)	476/500 (95.2%)
Retention	455/500 (91.0%)	465/500 (93.0%)

Before the study, we randomly divided the participants into two groups (50 participants per group): ***Group-1*** focuses on DeLucaUnLock and *BANN-TMGuard*, while ***Group-2*** focuses on TMGuard and *BANN-TMGuard*. For each group, the starting scheme was random, in order to avoid a pre-impression to the participants. In this case, a participant could start from either *BANN-TMGuard*, DeLucaUnLock, and TMGuard.

To explain our research goals and what kind of data would be gathered and recorded, we provided each participant with a set of guidelines and a consent form to sign. Also, each participant could have ten trials to get familiar with each scheme and ask any questions. All participants then conducted the study in our lab environment. The detailed study procedure and steps are summarized as below, including creation phase, confirmation phase and retention phase.

- Step (1) Creation phase: all participants have to generate their unlock credentials according to the scheme steps (e.g., TMGuard, BANN-TMGuard, and DeLucaUnLock).
- Step (2) Confirmation phase: all participants have to re-enter the credentials for verification (e.g., examining pattern and touch behavior) for 10 times. Participants can modify their credentials if they fail or want to change a pattern.
- Step (3) Distributed memory task: one paper-based finding tasks are given to distract them for 15 minutes.
- Step (4) Login phase: all participants should re-enter their unlock credentials for 10 times.
- Step (5) First feedback form: a *feedback from* with several questions is given to participants regarding the scheme usage.
- Step (6) Retention. After three days, all participants are invited to unlock the phone again for 10 times in our lab.
- Step (7) Second feedback form: another *feedback from* is given to participants regarding the scheme usage.

C. Result Analysis

During the user study, a total of 500 trials could be recorded for each confirmation, login and retention phase. Similar to other schemes, we used 60% trials in the training stage and the rest in the testing stage. We used Chi-square tests to decide whether the results are statistically different between two conditions. Table III and Table IV depict the success rate for confirmation, login and retention phase under two groups. We have the following observations in each phase.

- *Confirmation phase.* The participants in Group-1 could achieve a higher success rate of 96.2% under BANN-TMGuard than 94.4% under DeLucaUnLock. For both schemes, the errors were primarily caused due to the verification of behavioral input: 1) For DeLucaUnLock

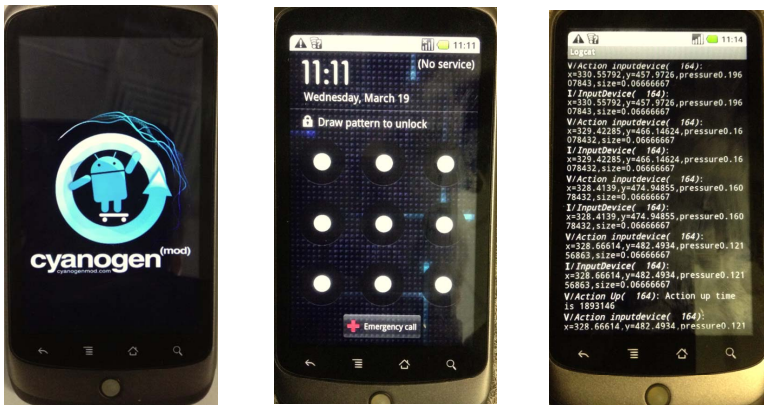

Fig. 8. (a) CyanogenMod Android OS; (b) The layout of Android unlock patterns; (c) An example of raw data.

TABLE IV
SUCCESS RATE IN THE CONFIRMATION, LOGIN AND RETENTION PHASE FOR GROUP-2.

Phase	TMGuard	BANN-TMGuard
Confirmation	477/500 (95.4%)	483/500 (96.6%)
Login	474/500 (94.8%)	478/500 (95.6%)
Retention	460/500 (92.0%)	470/500 (94.0%)

scheme, the touch pressure and touch time are the main affecting factors; 2) For BANN-TMGuard, touch speed and acceleration play an important role in authentication failures. It is worth noting that the pattern errors are few. For Group-2, participants could achieve a success rate of 95.4% and 96.6% for TMGuard and BANN-TMGuard, respectively. The errors made under both schemes were quite similar: touch speed and angle.

- *Login phase.* For Group-1, participants could reach a success rate of 93.8% and 95.2% under DeLucaUnLock and BANN-TMGuard, respectively. Several trials made a wrong pattern input, but most errors were made because of a malicious touch speed. For Group-2, participants could achieve a success rate of 94.8% and 95.6% for TMGuard and BANN-TMGuard. It is observed that BANN-TMGuard could provide a better authentication accurate rate for two groups.
- *Retention phase.* After three days, we invited all participants to joint our retention phase, and 92 of them were successfully returned (45 for Group-1 and 47 for Group-2). It is found that participants could achieve a success rate of 91% and 93% under DeLucaUnLock and BANN-TMGuard for Group-1, and 92% and 94% under TMGuard and BANN-TMGuard for Group-2. We found that the results were very positive as only a small decrease in success rate was noticed. Also, we notice there is a statistically significant difference, indicating that BANN-TMGuard could perform better than DeLucaUnLock and TMGuard.

The authentication results show the usability of all schemes, but our BANN-TMGuard could reach a better success rate than the other two schemes. Especially, BANN-TMGuard can outperform the others with a statistically significant difference in the retention phase.

TABLE V
TIME CONSUMPTION IN THE LOGIN FOR DIFFERENT SCHEMES.

Unlock Scheme	DeLucaUnLock	TMGuard	BANN-TMGuard
Time consumption (s)	4.1	4.3	4.1
Standard Deviation (s)	1.3	1.6	1.4

Table V describes the time consumption for each scheme. It is seen that there is no statistically significant difference in time consumption. More specifically, DeLucaUnLock scheme required around 4.1 second, TMGuard requires 4.3 seconds and BANN-TMGuard requires 4.1 seconds. This is because all these schemes adopted a similar authentication design by combining Android unlock pattern with touch behavioral authentication.

D. User Feedback

As stated in our study steps, each participant was given two feedback forms relating to the scheme usage, including both security and usability. Ten-point Likert scales were used for each question, where 1-score indicates strong disagreement and 10-score indicates strong agreement. Table VI describes the major questions and relevant scores. The main observations are described as below.

- *Group-1.* For the first two questions regarding pattern generation, both schemes received a good score: 8.9 vs. 9.1. Regarding time consumption, most participants gave a high score, i.e., both schemes received 9.3. Also, most participants supported that both schemes were easily to log in, and it is easy to remember the scheme credentials. For the last two questions, we explore the user's attitude towards the usability and security. It is found that a bit more participants considered BANN-TMGuard could be more usable and secure than DeLucaUnLock and TMGuard, i.e., the scores were less than 5.0. However, the result is not statistically significant, as the design of these schemes is quite similar.
- *Group-2.* Similarly, for pattern creation, BANN-TMGuard and TMGuard received a similar score of

TABLE VI
FEEDBACK FORM QUESTIONS AND AVERAGE SCORES FROM THE PARTICIPANTS.

Questions (Group-1)	Average Scores
1. I could easily create a pattern under DeLucaUnLock scheme	8.9
2. I could easily create a pattern under BANN-TMGuard	9.1
3. The time consumption for DeLucaUnLock scheme is acceptable	9.3
4. The time consumption for BANN-TMGuard is acceptable	9.3
5. I could easily log in to DeLucaUnLock scheme	8.7
6. I could easily log in to BANN-TMGuard	8.9
7. I could remember DeLucaUnLock credentials easily	8.8
8. I could remember BANN-TMGuard credentials easily	9.1
9. I think DeLucaUnLock scheme is more easily to use than BANN-TMGuard	4.2
10. I think DeLucaUnLock scheme is more secure than BANN-TMGuard	3.6
Questions (Group-2)	Average Scores
1. I could easily create a pattern under TMGuard	8.7
2. I could easily create a pattern under BANN-TMGuard	8.8
3. The time consumption for TMGuard scheme is acceptable	9.0
4. The time consumption for BANN-TMGuard is acceptable	9.1
5. I could easily log in to TMGuard	8.7
6. I could easily log in to BANN-TMGuard	8.8
7. I could remember TMGuard credentials easily	9.2
8. I could remember BANN-TMGuard credentials easily	9.3
9. I think TMGuard scheme is more easily to use than BANN-TMGuard	4.6
10. I think TMGuard is more secure than BANN-TMGuard	4.1

8.8 and 8.7. Most participants were satisfied with the time consumption, i.e., the score is 9.0 and 9.1 for TMGuard and BANN-TMGuard. They also found that the authentication was satisfied in the login phase (i.e., TMGuard 8.7 vs. BANN-TMGuard 8.8). The participants also considered the memorability of these credentials is not difficult. In the end, most participants felt these two schemes are very similar, but BANN-TMGuard was better in the retention phase.

Based on the received feedback, we can find most participants in Group-1 believed that DeLucaUnLock and BANN-TMGuard can have satisfied performance in the aspects of pattern generation, time consumption, and login. While more than half participants might vote for BANN-TMGuard to be more secure and usable. The main reason is that BANN-TMGuard adopted more touch features than DeLucaUnLock, while using ANN for modeling a user's touch behavior, as the statistical threshold in TMGuard can more easily cause errors.

E. Study on Blockchain-enabled ANNs

To examine the robustness and reliability of our blockchain-enabled ANN models, we set up an environment with five nodes, based on DevLeChain [31], which is an open blockchain development platform. More specifically, the environment was built by Intel Xeon E-2286M @2.4GHz x8, 128 GB ECC DDR4-2666, 1TB NVMe SSD and Ethereum 1.10.16. There is a need for 2/3 nodes in the network to sign a block to be appended to the blockchain.

In the study, we simulated adversarial scenarios with one and two malicious nodes, which would attempt to share false information. Then we measured the error rate of the trained ANN models in classification, as described in Table VII. It is found that without blockchain, the error rate could become 17.6% and 24.8% under one and two malicious nodes. While our blockchain-enabled method could address this issue by detecting and discarding malicious input. The results indicate that our *BANN-TMGuard* could ensure a more reliable ANN model under adversarial situations.

TABLE VII
ERROR RATE OF TRAINED ANN MODEL WITH AND WITHOUT BLOCKCHAIN.

Error Rate (%)	*Without blockchain*	*With blockchain*
One malicious node	17.6	0
Two malicious nodes	24.8	0

V. DISCUSSION

The study results indicate the promising performance of our BANN-TMGuard, but there are still some challenges and limitations on this topic.

- *Pattern selection.* In our current study, we did not consider which pattern the user created, e.g., using 4 dots or 9 dots. Basically, with different patterns, the unlock difficulty and deviations may be not the same. This is an interesting but open issue in the field. Also, it is an important topic to explore any dot-selection bias during the pattern creation.
- *The diversity of participants.* Our study involves a total of 100 participants, including bachelor, master and doctoral students, restaurant staff, university technical staff, and university academic members. We believe the background of participants is diverse; however, the majority is still student role. We plan to recruit more diverse participants in our future work.
- *Classifier performance.* In this work, we compared the performance among several machine learning classifiers. In the literature, there are many different algorithms including some updated classifiers. In the future, we plan

to consider more machine learning classifiers, e.g., hybrid learning, in the comparison.

- *Shoulder surfing attacks.* In the user study, though most participants may consider BANN-TMGuard is more secure, we did not perform an attack to test the schemes, e.g., shoulder surfing attack that is a common threat in public and crowded places. This is an open issue in the domain, and we plan to investigate the impact in our future work.
- *Blockchain involvement.* In our study, it showed that the integration of blockchain can benefit the ANN models against malicious input. Indeed, blockchain has been widely used to protect the integrity of shared data. However, blockchain itself may suffer some main constrains such as speed and latency, this is an important topic when involving blockchain in user authentication.
- *Face unlocking method.* As mentioned earlier, both Android unlock patterns and face recognition are popular unlocking method on smartphones. Compared with the use of faces, our BANN-TMGuard allows users to easily change their unlock patterns (credentials) with additional protection by checking touch behaviors. It also uses blockchain to secure data integrity during communication especially in an untrusted environment. In the literature, some privacy-preserving face recognition schemes have been proposed, hence it is promising to combine these two methods to achieve kind of multimodel user authentication for better security.

VI. Conclusions

In this work, we design *BANN-TMGuard*, a touch movement-based unlock mechanism based on blockchain-enabled artificial neural networks (ANNs). The goal is to enhance the previously scheme of *TMGuard* by considering more touch features such as touch pressure and touch acceleration. The use of ANNs aims to build a user's profile in an intelligent way, as *TMGuard* used a statistical method via proportional matching, which might cause errors easily. In the user study, we involved up to 100 participants and made two groups to compare the performance with TMGuard and DeLucaUnLock scheme separately. In terms of authentication accuracy, time consumption and users' feedback, it is observed that our *BANN-TMGuard* could provide a better success rate than the other schemes, especially in the retention phase with statistically significant result, and receive higher scores from users. Further, our test showed that *BANN-TMGuard* could provide a more reliable ANN model against malicious input.

Acknowledgment

We would like to thank all the participates for their hard work in the study. This work was partially funded by EU H2020 CyberSec4Europe.

References

[1] Consumer Electronics Market Size, Share & COVID-19 Impact. (Accessed on 1 October 2022) https://www.fortunebusinessinsights.com/consumer-electronics-market-104693

[2] Smartphone Market Suffers Fifth Consecutive Decline in Global Shipments in Q3 2022. (Accessed on 5 October 2022) https://www.idc.com/getdoc.jsp?containerId=prUS49809922

[3] 100 Most Common Passwords Of 2022. Can You Spot Your Password? (Accessed on 5 October 2022) https://techcult.com/most-common-passwords/

[4] Berkeley Churchill: Unlock Pattern Generator (2013). https://www.berkeleychurchill.com/software/android-pwgen/pwgen.php

[5] LockDroid C Android-inspired pattern lock for iOS. (Accessed on 5 October 2022) https://yalujailbreak.net/android-pattern-lock-for-iphone/

[6] M.A. Alawami, A.R. Vinay, and H. Kim, "LocID: A Secure and Usable Location-Based Smartphone Unlocking Scheme Using Wi-Fi Signals and Light Intensity," *IEEE Internet Things J.* 9(23): 24357-24372, 2022.

[7] P. Andriotis, T. Tryfonas, G.C. Oikonomou, C. Yildiz, "A pilot study on the security of pattern screen-lock methods and soft side channel attacks," *Proc. WISEC*, pp. 1-6, 2013.

[8] A.J. Aviv, K.L. Gibson, E. Mossop, M. Blaze, and J.M. Smith, "Smudge Attacks on Smartphone Touch Screens," *Proc. WOOT*, pp. 1-10, 2010.

[9] I. Becker, S. Parkin, and M.A. Sasse, "The Rewards and Costs of Stronger Passwords in a University: Linking Password Lifetime to Strength," *USENIX Security Symposium*, pp. 239-253, 2018.

[10] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Trans. Inf. Syst. Secur.* 5(4), pp. 367-397, 2002.

[11] B.A.P. Botelho, E.T. Nakamura, and N. Uto, "Security Analysis of Touch Inputted Passwords - A Preliminary Study Based on the Resistance against Brute Force Attacks," *Proc. NSS*, pp. 714-720, 2013.

[12] M.A.P. Chamikara, P. Bertok, I. Khalil, D. Liu, S. Camtepe, Privacy Preserving Face Recognition Utilizing Differential Privacy. Comput. Secur. 97: 101951 (2020)

[13] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," *Proc. ACM CHI*, pp. 987-996, 2012.

[14] T.J. Forman and A.J. Aviv, "Double Patterns: A Usable Solution to Increase the Security of Android Unlock Patterns," *Proc. ACSAC*, pp. 219-233, 2020.

[15] Y. Guo, L. Yang, X. Ding, J. Han, and Y. Liu, "OpenSesame: Unlocking smart phone through handshaking biometrics," *Proc. INFOCOM*, pp. 365-369, 2013.

[16] T. Gleerup, W. Li, J. Tan, Y. and Wang, "ZoomPass: A Zoom-Based Android Unlock Scheme on Smart Devices," *Proc. SciSec*, pp. 245-259, 2022.

[17] T. Hupperich and K. Dassel, "On the Usefulness of User Nudging and Strength Indication Concerning Unlock Pattern Security," *Proc. TrustCom*, pp. 1646-1654, 2020.

[18] R. Izuta, K. Murao, T. Terada, T. Iso, H. Inamura, M. Tsukamoto, "Screen Unlocking Method using Behavioral Characteristics when Taking Mobile Phone from Pocket," *Proc. MoMM*, pp. 110-114, 2016.

[19] K. Jiokeng, G. Jakllari, and A.L. Beylot, "I Want to Know Your Hand: Authentication on Commodity Mobile Phones Based on Your Hand's Vibrations," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6(2): 58:1-58:27, 2022.

[20] T. Kwon and J. Hong, "Analysis and Improvement of a PIN-Entry Method Resilient to Shoulder-Surfing and Recording Attacks," *IEEE Trans. Inf. Forensics Secur.* 10(2), pp. 278-292, 2015.

[21] W. Li, J. Tan, W. Meng, Y. Wang, and J. Li, "SwipeVLock: A Supervised Unlocking Mechanism Based on Swipe Behavior on Smartphones," *Proc. ML4CS*, pp, 140-153, 2019.

[22] W. Li, J. Tan, W. Meng, Y. Wang, "A Swipe-based Unlocking Mechanism with Supervised Learning on Smartphones: Design and Evaluation," *Journal of Network and Computer Applications*, vol. 165, 102687, 2020.

[23] W. Li, J. Tan, and N. Zhu, and Y. Wang, "Designing Double-Click-based Unlocking Mechanism on Smartphones," *Proc. 2020 EISA*, pp. 573-585, 2020.

[24] W. Li, Y. Wang, J. Li, and Y. Xiang, "Toward Supervised Shape-based Behavioral Authentication on Smartphones," *Journal of Information Security and Applications*, vol. 55, 102591, 2020.

[25] W. Li, J. Tan, and N. Zhu, "Double-X: Towards Double-Cross-based Unlock Mechanism on Smartphones," *Proc. 37th IFIP SEC*, pp. 412-428, 2022.

[26] W. Li, Y. Wang, J. Tan, and N. Zhu, "DCUS: Evaluating Double-Click-Based Unlocking Scheme on Smartphones," *Mob. Networks Appl.* 27(1), pp. 382-391, 2022.

[27] T. Li, T. Xia, H. Wang, Z. Tu, S. Tarkoma, Z. Han, and P. Hui, "Smartphone App Usage Analysis: Datasets, Methods, and Application,". *IEEE Commun. Surv. Tutorials* 24(2), pp. 937-966, 2022.
[28] W. Meng, D.S. Wong, S. Furnell, and J. Zhou, "Surveying the Development of Biometric User Authentication on Mobile Phones," *IEEE Commun. Surv. Tutorials* 17(3), pp. 1268-1293, 2015.
[29] W. Meng, W. Li, D.S. Wong, and J. Zhou, "TMGuard: A Touch Movement-Based Security Mechanism for Screen Unlock Patterns on Smartphones," *Proc. ACNS*, pp. 629-647, 2016.
[30] W. Meng, L. Jiang, K.K.R. Choo, Y. Wang, and C. Jiang, "Towards Detection of Juice Filming Charging Attacks via Supervised CPU Usage Analysis on Smartphones," *Computers and Electrical Engineering*, vol. 78, pp. 230-241, 2019.
[31] W.Y. Chiu and W. Meng, "DevLeChain - An Open Blockchain Development Platform for Decentralized Applications," *Proc. IEEE Blockchain*, pp. 167-176, 2022.
[32] J. Yan, A.F. Blackwell, R.J. Anderson, and A. Grant, "Password Memorability and Security: Empirical Results," *IEEE Secur. Priv.* 2(5), pp. 25-31, 2004.
[33] X. Suo, Y. Zhu, and G.S. Owen, "Graphical Passwords: A Survey," *Proc. ACSAC*, pp. 463-472, 2005.
[34] H. Tao and C. Adams, "Pass-Go: A Proposal to Improve the Usability of Graphical Passwords," *Int. J. Netw. Secur.* 7(2), pp. 273-292, 2008.
[35] H. Tupsamudre, V. Banahatti, S. Lodha, and K. Vyas, "Pass-O: A Proposal to Improve the Security of Pattern Unlock Scheme," *Proc. AsiaCCS*, pp. 400-407, 2017.
[36] S. Uellenbeck, M. Durmuth, C. Wolf, and T. Holz, "Quantifying the security of graphical passwords: the case of android unlock patterns," *Proc. CCS*, pp. 161-172, 2013.
[37] L. Wang, K. Huang, K. Sun, W. Wang, C. Tian, L. Xie, and Q. Gu, "Unlock with Your Heart: Heartbeat-based Authentication on Commercial Mobile Phones," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2(3): 140:1-140:22, 2018.
[38] L. Wang, W. Meng, and W. Li, "Towards DTW-based Unlock Scheme using Handwritten Graphics on Smartphones," *Proc. MSN*, pp. 486-493, 2021.
[39] S. Yi, Z. Qin, N. Carter, and Q. Li, "WearLock: Unlocking Your Phone via Acoustics Using Smartwatch," *Proc. ICDCS*, pp. 469-479, 2017.
[40] X. Zhao, T. Feng, W. Shi, and I.A. Kakadiaris, "Mobile User Authentication Using Statistical Touch Dynamics Images," *IEEE Trans. Inf. Forensics Secur.* 9(11), pp. 1780-1789, 2014.
[41] N. Zheng, K. Bai, H. Huang, AND H. Wang, You Are How You Touch: User Verification on Smartphones via Tapping Behaviors. *Proc. ICNP*, pp. 221-232, 2014.

PLACE PHOTO HERE

Weizhi Meng is currently an Associate Professor in the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Denmark. He obtained his Ph.D. degree in Computer Science from the City University of Hong Kong (CityU). Prior to joining DTU, he worked as research scientist in Institute for Infocomm Research, Singapore. He won the Outstanding Academic Performance Award during his doctoral study, and is a recipient of the Hong Kong Institution of Engineers (HKIE) Outstanding Paper Award for Young Engineers/Researchers in both 2014 and 2017. His primary research interests are cyber security and intelligent technology in security including intrusion detection, smartphone security, biometric authentication, HCI security, cloud security, trust management, blockchain in security, cyber-physical system security and IoT security. He is a senior member of IEEE.

PLACE PHOTO HERE

Wenjuan Li is currently a Research Assistant Professor in the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, China. Her research interests include network management and security, intrusion detection, trust management, blockchain security, and E-commerce security. She is a senior member of IEEE.

PLACE PHOTO HERE

Andrei Nicolae Calugar was a Master Student at Department of Applied Mathematics and Computer Science, Technical University of Denmark. His research interests include machine learning application, IoT security and intrusion detection.