TouchID: User Authentication on Mobile Devices via Inertial-Touch Gesture Analysis

XINCHEN ZHANG, State Key Laboratory for Novel Software Technology, Nanjing University, China
YAFENG YIN*, State Key Laboratory for Novel Software Technology, Nanjing University, China
LEI XIE, State Key Laboratory for Novel Software Technology, Nanjing University, China
HAO ZHANG, State Key Laboratory for Novel Software Technology, Nanjing University, China
ZEFAN GE, State Key Laboratory for Novel Software Technology, Nanjing University, China
SANGLU LU, State Key Laboratory for Novel Software Technology, Nanjing University, China

Due to the widespread use of mobile devices, it is essential to authenticate users on mobile devices to prevent sensitive information leakage. In this paper, we propose TouchID, which combinedly uses the touch sensor and the inertial sensor for gesture analysis, to provide a touch gesture based user authentication scheme. Specifically, TouchID utilizes the touch sensor to analyze the on-screen gesture while using the inertial sensor to analyze the device's motion caused by the touch gesture, and then combines the unique features from the on-screen gesture and the device's motion for user authentication. To mitigate the intra-class difference and reduce the inter-class similarity, we propose a spatial alignment method for sensor data and segment the touch gesture into multiple sub-gestures in space domain, to keep the stability of the same user and enhance the discriminability of different users. To provide a uniform representation of touch gestures with different topological structures, we present a four-part based feature selection method, which classifies a touch gesture into a start node, an end node, the turning node(s), and the smooth paths, and then select effective features from these parts based on Fisher Score. In addition, considering the uncertainty of user's postures, which may change the sensor data of same touch gesture, we propose a multi-threshold kNN based model to adaptively tolerate the posture difference for user authentication. Finally, we implement TouchID on commercial smartphones and conduct extensive experiments to evaluate TouchID. The experiment results show that TouchID can achieve a good performance for user authentication, i.e., having a low equal error rate of 4.90%.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing**.

Additional Key Words and Phrases: User authentication, Touch gesture, Mobile devices, Inertial-touch gesture analysis

ACM Reference Format:
Xinchen Zhang, Yafeng Yin, Lei Xie, Hao Zhang, Zefan Ge, and Sanglu Lu. 2020. TouchID: User Authentication on Mobile Devices via Inertial-Touch Gesture Analysis. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 162 (December 2020), 29 pages. https://doi.org/10.1145/3432192

*Corresponding Author

Authors' addresses: Xinchen Zhang, xczhang@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China; Yafeng Yin, yafeng@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China; Lei Xie, lxie@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China; Hao Zhang, h.zhang@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China; Zefan Ge, zefan@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China; Sanglu Lu, sanglu@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Ave, Nanjing, 210023, China.

2474-9567/2020/12-ART162 $15.00
https://doi.org/10.1145/3432192

1 INTRODUCTION

With the widespread use of mobile devices (e.g., smartphones, smartwatches, and tablets) in daily life, more and more sensitive information such as photos, emails, chat messages and bank accounts is stored on mobile devices, thus it is essential to authenticate users to prevent sensitive information leakage [21]. Traditionally, the knowledge based authentication schemes were used, which mainly required the user to input the predefined PIN codes or patterns for authentication. Unfortunately, these schemes are vulnerable to various attacks such as smudge attack [2], shoulder surfing attack [49, 54], and password inference attack [56, 57, 62]. To overcome these issues, the physiological feature based authentication schemes were proposed, which mainly utilized the unique fingerprints [39, 41] and face features [13, 20] for authentication. It is convenient to stretch out the finger or show the face for authentication, thus these schemes are adopted in many devices. However, capturing fingerprints or face features often require dedicated or expensive hardwares, e.g., fingerprint sensor or high-resolution camera. Besides, the physiological feature based authentication schemes can be vulnerable to replaying and spoofing attacks [12, 25, 37, 40]. For example, the fake fingerprint generated by 3D printer can achieve an average attack success rate of 80% [25], while the images, videos or 3D face masks can be used to spoof the face authentication systems [12].

In fact, whatever the knowledge based or the physiological feature based authentication schemes, they utilize "what" the user knows or "what" the user has for user authentication, while ignoring "how" the user performs during the authentication process, which is the focus of this paper. To solve this problem, the behavioral biometrics based authentication schemes were proposed, which focused on the difference of user behaviors in performing gestures for authentication. For example, using the unique vibration characteristics of finger knuckles when being tapped for authentication [7], performing a sequence of rhythmic taps/slides on a device screen [8] for authentication, using the hand's geometry in taps generated by different fingers for authentication [51], using the inter-stroke relationship between multiple fingers in touch gestures for authentication [42]. These methods often worked with common sensors embedded in many off-the-shelf devices instead of the dedicated sensors used in physiological feature based authentication schemes. However, the existing work often designed customized gestures for user authentication, while the gestures were rarely adopted in commercial mobile devices.

Different from the existing work, this paper aims to provide an online user authentication scheme TouchID, which is expected to resist the common attacks like smudge attack [2], shoulder surfing attack [49, 54], password inference attack [56, 57, 62], and spoofing attacks [25, 37, 40]. In TouchID, the user performs a widely adopted graphic pattern based touch gesture with one finger on the commercial mobile device for user authentication, as shown in Fig. 1. The unlock operation is the same with the existing graphic pattern based unlocking method and easy to use. In regard to the graphic pattern, it is common and has been integrated into many COTS devices (e.g., Android-powered smartphones which have a 87% global market share [27]) and applications (e.g., payment software Alipay [4], image management software Safe Vault [58]). When compared with the existing work [42, 47, 51] designing customized gestures, TouchID has no access to the specific features like the displacements between different fingers and the relationship between a series of customized gestures, thus user authentication in TouchID can be more challenging. As shown in Fig. 1, when performing a touch gesture, TouchID leverages the touch sensor and the inertial sensor to capture the on-screen gesture and the device's motion, respectively. Then, TouchID utilizes the unique biometric feature of the user's finger and the touch behavior, i.e., the geometry of on-screen gesture and the micro movement of the device, to perform user authentication in an online manner. However, to achieve the above goal, it is necessary to mitigate the intra-class difference and reduce the inter-class similarity of gestures, i.e., enhancing the stability of gestures from the same user while enhancing the discriminability of gestures from different users. Specifically, we need to solve the following three challenges, to provide the user authentication scheme TouchID.

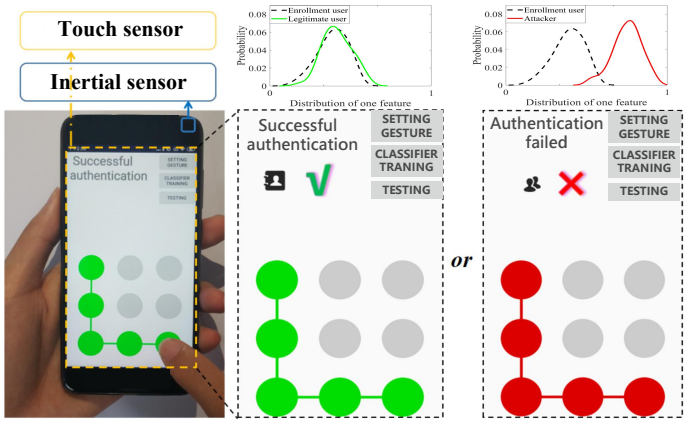


Fig. 1. A typical application scenario of TouchID

The first challenge is how to mitigate the intra-class difference and reduce the inter-class similarity among gestures? To address this challenge, we first conduct extensive experimental studies to observe how the finger moves in a touch gesture and how the device moves with touch gesture. We find that time differences among touch gestures can affect intra-class difference and inter-class similarity. Therefore, we propose a spatial alignment method to align the sensor data in space domain, and then segment the touch gesture into multiple sub-gestures to highlight the sub-gesture which contributes more for enhancing the stability of the same user and the discriminability of different users.

The second challenge is how to represent the touch gesture with different topological structures in a uniform way? The touch gestures corresponding to different graphic patterns have a different number of sub-gestures and the sub-gestures can also be different, which may lead to the different representation of a touch gesture. To address this challenge, we propose a four-part feature selection method, which classifies a touch gesture into four parts, i.e., a start node, an end node, turning node(s), and smooth paths, whatever the topological structure of the touch gesture is. Then, we select effective features for each part based on Fisher Score. Finally, for each gesture, we can represent it with a feature vector consisted of the uniform feature set.

The third challenge is how to tolerate the uncertainty caused by different body postures and hand postures? When performing a touch gesture, the user can sit, lay or stand, and she/he can interact with the device with one hand or two hands, the different postures will lead to the inconsistency of the sensor data for the same touch gesture. To address this challenge, we design a multi-threshold KNN based model to separate the touch gestures under different postures into different clusters adaptively, and then perform user authentication in each cluster. In addition, to reduce the computation overhead of multi-threshold kNN, we only use a small number of samples for training.

We make three main contributions in this paper. 1) We conduct an extensive experimental study to observe the finger's movement and the device's motion when performing a touch gesture, and then propose a spatial alignment method to align the touch gesture in space domain and segment the gesture into sub-gestures, to enhance the stability of the same user and the discriminability of different users. 2) Based on a comprehensive analysis of touch sensor data and inertial sensor data, we propose a four-part feature selection method to represent the touch gestures with different topological structures in a uniform way, and select effective features based on Fisher Score by considering both the intra-class stability and the inter-class discriminability. In addition, we also propose a multi-threshold kNN based model to mitigate the effect of different postures. 3) We implement TouchID on an Android-powered smartphone and conduct a lot of experiments to evaluate the efficiency of

TouchID. The experiment results show that TouchID can achieve a low equal error rate for user authentication and outperforms the existing solutions.

2 RELATED WORK

When considering the unique features in user behaviors, a variety of gesture based user authentication schemes have been proposed. The gestures include hand gestures [7, 19], eye movements [10, 16], lip motions [36, 48], heartbeats [24, 35, 52], touch gestures [1, 8, 11, 15, 30, 33, 42, 43, 46, 47, 51, 55, 59–61], and so on. Among the gestures, touch gestures are often used to authenticate owners of mobile devices, since users often interact with mobile devices with touch gestures. In this paper, we will summarize the related work using touch gestures for user authentication on mobile devices. In addition, from the perspective of sensors used for monitoring the touch gestures, we mainly classify the related work into three categories, i.e., touch sensor based, inertial sensor based, and inertial-touch based user authentication.

Touch sensor based user authentication: When performing an on-screen gesture, the touch sensor can provide the coordinates of fingertips and sizes of touch areas, which can be used for inferring the moving directions, moving speeds, moving trajectories of fingertips, relative distances among multiple fingers, etc. The unique features in touch gestures can be used to differentiate users. Until now, many touch-related gestures, e.g., swiping [1, 8, 11, 15, 55, 59, 60], tapping [8, 33, 61], zooming in/out [60] and some user-defined gestures [47], have been proposed for user authentication. When performing a single-touch swiping gesture on the screen, Frank et al. [15] investigated whether a classifier can continuously authenticate users. Chen et al. [8] required users to perform a sequence of rhythmic tapping or swiping gestures on a multi-touch mobile device, then it extracted features from rhythmic gestures to authenticate users. Moreover, Song et al. [47] studied multi-touch swiping gestures and showed that both the hand geometry and the behavioral biometric can be recorded in these gestures and used for user authentication. Besides, some methods also proposed the usage of image-based features for modeling touchscreen data. Zhao et al. [60] proposed a novel Graphic Touch Gesture Feature (GTGF) to extract the identity traits from swiping and zooming in/out gestures, where the intensity values and shapes of the GTGF represent the moving trace and pressure dynamically. The existing work indicates the touch sensor can be used for on-screen gesture authentication. However, only with the touch sensor, these methods mainly focus on the geometry features on the screen. To get more distinguishable features, they may require the user to perform gestures in a rhythm or adopt the user-defined gestures.

Inertial sensor based user authentication: The inertial sensor means the accelerometer, gyroscope, magnetometer, or any combination of the three sensors. It can measure the device's motion related to the touch gesture for user authentication. For example, when tapping on the smartphone, Sitová et al. [46] used the accelerometer, gyroscope, and magnetometer to capture the micro hand movements and dynamic orientation changes during several tapping gestures for user authentication. Chen et al. [7] used the accelerometer in smartwatchs to capture the vibration characteristics when a user taps her/his finger knuckles, then they extracted features from vibration signals to authenticate users. Shen et al. [43] used the accelerometer, gyroscope, magnetometer to capture behavioral traits during swiping gestures and built a one-class Markov-based decision procedure to authentication users. Guerra-Casanova et al. [19] used the accelerometer in a mobile device to depict the behavioral characteristics when a user performs hand gestures while holding the mobile device. The existing work indicates that the inertial sensor can be used for touch gesture based user authentication. However, different from the touch sensor, the inertial sensor mainly captures the indirect sensor data of touch gesture, i.e., using the sensor data corresponding to device motions to infer the characteristics in touch gestures, and it is often used to authenticate simple or short gestures like tapping and swiping.

Inertial-touch based user authentication: When combining the touch sensor and inertial sensor, it is possible to capture the on-screen gesture and the device's motion at the same time, thus enriching the sensor data

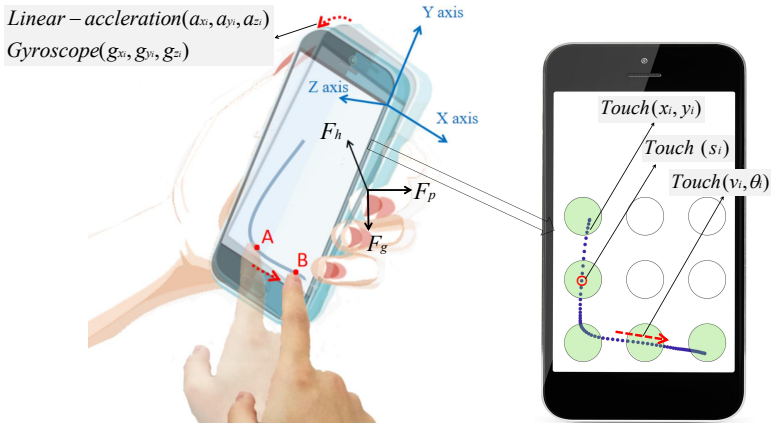


Fig. 2. Performing a touch gesture on a smartphone

for user authentication. When using the customized touch gestures on the smartphone for user authentication, Shahzad et al. [42] designed and chose 10 effective gestures that including 3 single-touch gestures and 7 multi-touch gestures, and then proposed GEAT, which extracted behavioral features from the touch sensor and the accelerometer for user authentication. Jain et al. [30] utilized the accelerometer, orientation and the touch sensor to capture the touch characteristics during swiping gestures, and used the modified Hausdorff distance as the classifier to authenticate users. Wang et al. [51] also utilized the accelerometer, gyroscope, and the touch sensor to capture the geometry of the user's hand when performing the user-defined gestures that require the user to tap four times with one finger or tap once with four fingers. The existing work indicates that using the touch sensor as well as the inertial sensor can capture both on-screen features and the motion features of the device. However, the existing work tends to introduce multi-touch gestures and user-defined gestures, to capture more specific features corresponding to a user (e.g., the relations between fingers in a gesture) for better authentication.

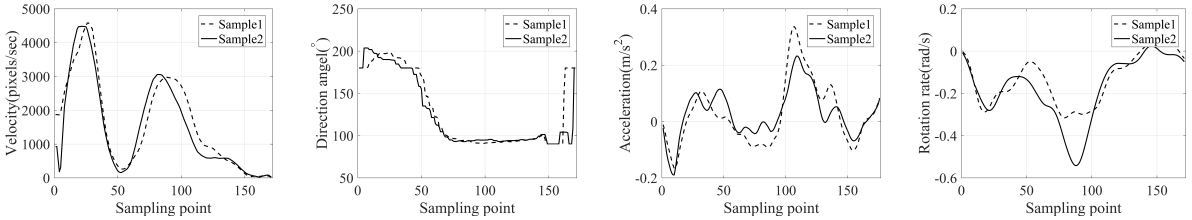
To capture both the on-screen gesture and device motions, we utilize the touch sensor, accelerometer, and gyroscope for user authentication. In regard to the touch gesture, the existing work mainly used the customized gestures for user authentication, while the gestures are rarely used in commercial mobile devices. In this paper, the user only performs a widely adopted unlock gesture, i.e., the graphic pattern based touch gesture, with one finger for user authentication. Due to the lack of specific features like displacements between different fingertips in multi-touch gestures and the relationship between a series of customized gestures, user authentication in this paper which only uses a widely adopted single touch gesture can be more challenging.

3 OBSERVATIONS AND MODELING

In this section, we will conduct extensive experiments to observe how the user performs a touch gesture and how the gesture affects the sensor data. Unless otherwise specified, the user performs a common unlock gesture on a 3×3 grid on Samsung Galaxy S9 smartphone with a 5.8-inch screen, as shown in Fig. 2. Then, we use the touch sensor and the inertial sensor (i.e., accelerometer and gyroscope) to collect the sensor data for analysis. The sampling rates of the touch sensor, accelerometer, gyroscope are set to 60 Hz, 100 Hz, and 100 Hz, respectively.

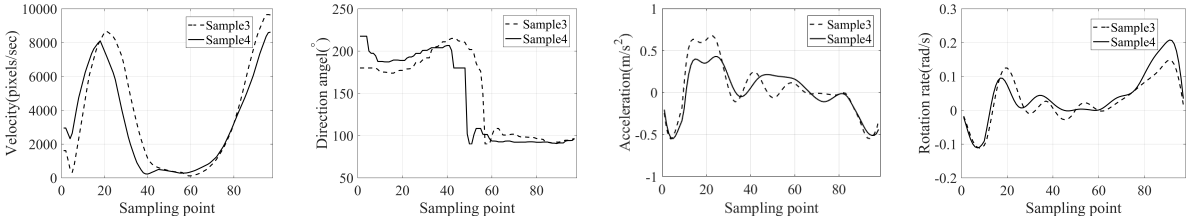
3.1 Finger Movements and Device Motions during a Touch Gesture

A touch gesture can be measured with the fingertip's coordinates on the screen, the fingertip's touch sizes along the trajectory, and the device's motions along the time. When the fingertip touches the screen, it will generate the following data, i.e., the coordinate, the touch size, the pressure of the fingertip. However, due to the limitation of



(a) Velocity (Touch sensor) (b) Direction (Touch sensor) (c) Acceleration-Z (Inertial sensor) (d) Rotation-Y (Inertial sensor)

Fig. 3. Touch sensor data and inertial sensor data of two touch gestures for user 1



(a) Velocity (Touch sensor) (b) Direction (Touch sensor) (c) Acceleration-Z (Inertial sensor) (d) Rotation-Y (Inertial sensor)

Fig. 4. Touch sensor data and inertial sensor data of two touch gestures for user 2

Android API [26], the measured pressure in many smartphones is either 0 or 1, i.e., touching or non-touching, which is too coarse-grained to analyze the touch force. Therefore, we use the device's motion caused by the touch gesture to represent the pressure indirectly. As shown in Fig. 2, when performing a touch gesture, we can obtain the moving trajectory and touch sizes of the fingertip along the time from the touch sensor. In regard to the device's motion, it is caused by the resultant force from the gravity F_g, the hand grasping the phone F_h and the fingertip pressing the screen F_p. When holding the device statically in a fixed orientation, the forces from the gravity and the hand can be treated as constant forces, thus the device's motion is mainly caused by the finger's pressure. That is to say, the device's motion measured by the embedded accelerometer and gyroscope can represent the finger's pressure. Consequently, we can combine the on-screen gesture and the device's motion to describe a touch gesture on the mobile device.

3.2 Feasibility of User Authentication

The touch gestures demonstrate the stability of gestures from the same user, while demonstrating the discriminability of gestures from different users. To explore whether the touch gesture can be used for user authentication, we first invite two users and each one performs the gesture 'L' twice on the same smartphone, as shown in Fig. 2. In Fig. 3 and Fig. 4, we show the velocity and direction inferred from the touch sensor data, as well as linear acceleration in z-axis and angular velocity in y-axis measured from the inertial sensor for each user, respectively. The solid and dashed lines in the same figure indicate that the gestures from the same user have a high consistency. When comparing the figures in the same column of Fig. 3 and Fig. 4, we can conclude that the gestures from different users have differences in sensor data.

To measure the similarity (or difference) between sensor data from different gestures, we introduce the operations of normalization and interpolation, as well as the metric of Root Mean Squard Error (RMSE) [6]. For simplicity, we use $d_{1_i}, i \in [1, n_1]$, $d_{2_i}, i \in [1, n_2]$ to represent the time-series data for the first gesture and

the second gesture, and then we normalize $d_{p_i}, p \in [1,2]$ with Eq. (1). Here, d'_{p_i} means the normalized data, $d_{p_{min}} = \{d_{p_i} | d_{p_i} \leq d_{p_j}, \forall j \neq i\}, d_{p_{max}} = \{d_{p_i} | d_{p_i} \geq d_{p_j}, \forall j \neq i\}$.

$$d'_{p_i} = \begin{cases} \dfrac{d_{p_i} - d_{p_{min}}}{d_{p_{max}} - d_{p_{min}}} & ,d_{p_{min}} \neq d_{max}, \\ 0 & ,d_{p_{min}} = d_{p_{max}}. \end{cases} \tag{1}$$

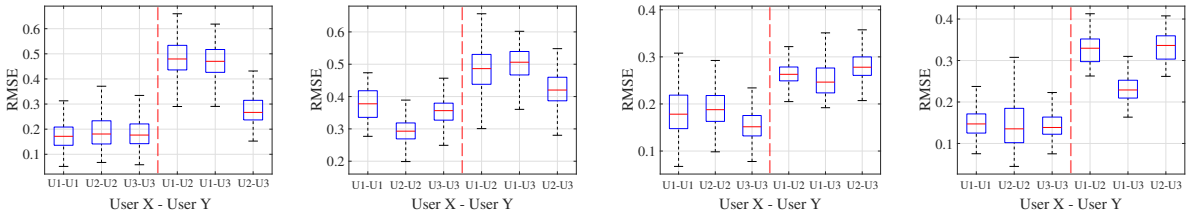
Considering that the length of d'_{1_i} and d'_{2_i} can be different, i.e., $n_1 \neq n_2$, we introduce a linear interpolation algorithm [9] to make the length of d'_{1_i} and d'_{2_i} be equal. Suppose $n_1 > n_2$, we need to interpolate data points into d'_{2_i} to change the length of d'_{2_i} as n_1. Consequently, the interval between consecutive data points is changed to $\frac{n_2-1}{n_1-1}$, and the kth data point for the second gesture is changed to Eq. (2), where $k \in [2, n_1]$. When $k = 1$, $d''_{2_1} = d'_{2_1}$.

$$d''_{2_k} = d'_{2_i} + (d'_{2_{i+1}} - d'_{2_i}) * (k \cdot \frac{n_2 - 1}{n_1 - 1} - i), i = \lfloor k \cdot \frac{n_2 - 1}{n_1 - 1} \rfloor \tag{2}$$

At this time, we have obtained the time-series data d'_{1_i} and d''_{2_k} with the same length n_1. Then, we can use Eq. (3) to calculate the similarity (or difference), i.e., RMSE value r_{12}, between them. Here, $r_{12} \in [0,1]$, the smaller the value of r_{12}, the higher the similarity (i.e., the smaller the difference).

$$r_{12} = \sqrt{\frac{1}{n_1} \sum_{k=1}^{k=n_1} (d'_{1_k} - d''_{2_k})} \tag{3}$$

To use the RMSE value to illustrate the stability or discriminability among gestures, we invite three users and each user performs gesture 'L' 50 times. Then we calculate the RMSE value of sensor data from the same user and that from different users, respectively. According to Fig. 5, the RMSE value corresponding to the same user (i.e., 'Ui-Uj', $i = j$) is generally less than that corresponding to different users (i.e., 'Ui-Uj', $i \neq j$). It indicates that the gestures from the same user keep the similarity while the gestures from different users have unavoidable difference.

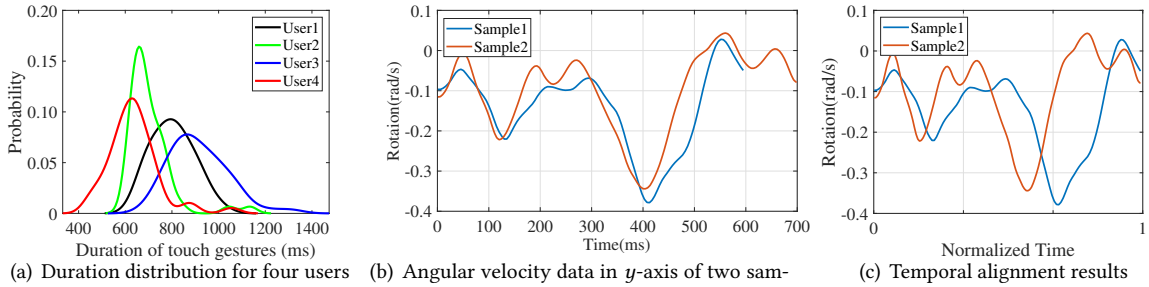


(a) Velocity (b) Direction (c) Acceleration-Z (d) Rotation-Y

Fig. 5. The RMSE values between three users along each sensor data

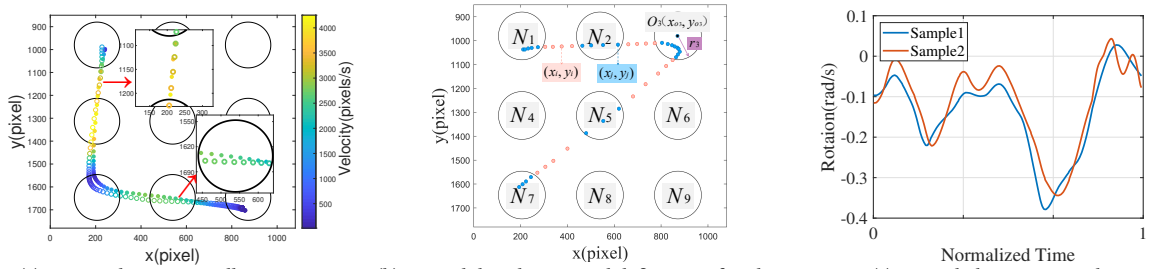
3.3 Sensor Data Alignment in Space Domain

The sensor data of touch gestures has unavoidable differences in time domain while keeping non-negligible consistencies in space domain. Due to the uncertainty of user behaviors, when a user performs the same gesture multiple times, the duration of each gesture can be different, which will reduce the stability of gestures from the same user and lead to an authentication error. However, due to the layout constraint of on-screen gesture, i.e., the fixed locations of nodes, the trajectories of gestures corresponding to the same graphic pattern keep essential consistencies. This motivates us to align the sensor data of gestures based on the graphic pattern, to improve the stability of gestures from the same user.



(a) Duration distribution for four users (b) Angular velocity data in y-axis of two samples from the same user (c) Temporal alignment results

Fig. 6. Temporal characteristics of touch gestures



(a) Sensor data is spatially consistent (b) Spatial distribution and definition of nodes (c) Spatial alignment results

Fig. 7. Spatial characteristics of touch gestures

In fact, data alignment is an effective way in data processing and has been used in many scenarios, such as signal alignment in communications (e.g., beam alignment in RADAR [17], optical axis alignment of the transmitter and receiver in LiDAR [14], C/A code alignment of the receiver and satellite in GPS [31]), point matching in point set registration [32, 38], sequence alignment in videos [5], and so on. Take the sequence alignment task [5] as an example, they leverage both spatial displacement and temporal variations between image frames as cues, to correlate two different video sequences of the same dynamic scene in time and in space. Differently, we adopt the layout constraint in space domain as spatial cues to align the time-series sensor data, as described below.

Unavoidable time difference among touch gestures: To demonstrate the time difference among touch gestures, we invite four users to perform the gesture 'L' on the screen, as shown in Fig. 2. Each one performs the same gesture 50 times. As shown in Fig. 6(a), the durations of gestures corresponding to the same graphic pattern 'L' can be different, whether the gestures are performed by the same user or different users. Specifically, in Fig. 6(b), we show the angular velocities in y-axis of two gestures corresponding to 'L' from the same user. The duration difference between the two gestures (i.e., sample 1 and sample 2) is about 100 ms. At this time, to calculate the similarity between them, the temporal alignment method is often adopted, e.g, using the linear interpolation algorithm [9] in time domain to make the number of data points in sample 1 and that in sample 2 be equal, as shown in Fig. 6(c). However, this temporal alignment method may break the consistency between gestures, i.e., decreasing the stability of gestures from the same user, as the misaligned peaks shown in Fig. 6(c). It indicates that it is inappropriate to align the sensor data in time domain.

Non-negligible space consistency among gestures: Different from the sensor data in time domain, the touch gestures in space domain are constrained by the layout of lock screen, e.g., the 3×3 grid in Fig. 2. Consequently, the gestures corresponding to the same graphic pattern will keep the consistency in space domain.

As shown in Fig. 7(a), we show the moving velocity of the fingertip in each touch gesture, whose graphic pattern is 'L'. We can find that the moving velocities of two gestures have a high consistency in space domain, e.g., having smaller velocities in nodes and larger velocities between nodes. It indicates that it is possible to align the sensor data in space domain to keep the stability of gestures.

To achieve the above goal, we first define the touch gesture on the screen in space domain. As shown in Fig. 7(b), we use $p_i(x_i, y_i), i \in [1, n]$ to represent the coordinate of the ith point in the moving trajectory of a touch gesture, while using the pair $< O_k, r_k >$ to represent the kth node in the lock screen. Here, $O_k(x_{o_k}, y_{o_k}), k \in [1, m]$ and r_k represent the center and radius of the node. When considering the layout constraint of lock screen, the points in a moving trajectory can be classified into in-node points (i.e., blue points in Fig. 7(b)) and out-node points (i.e., red points in Fig. 7(b)). That is to say, a touch gesture can be represented with in-node points and out-node points by turns along with time.

For an on-screen point $p_i(x_i, y_i)$, if it satisfies Eq. (4), it is located in the kth node. Otherwise, it is out of the kth node.

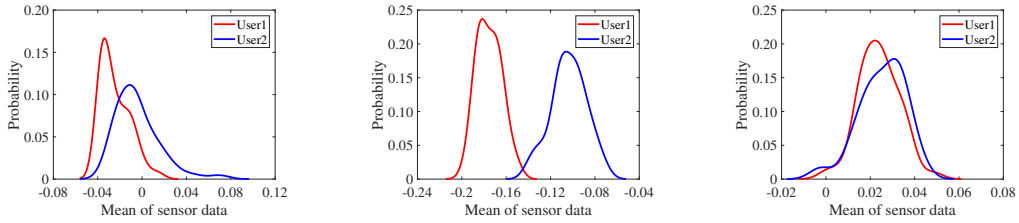
$$\sqrt{(x_i - x_{o_k})^2 + (y_i - y_{o_k})^2} \leq r_k, k \in [1, m] \tag{4}$$

In this way, we can represent all the n_k points in the kth node as $p_{k_j}, j \in [1, n_k], k_j < k_{j+1}$ in sequence, based on the occurrence time of point. In regard to the non-node points occurring between the kth node and $(k+1)$th node, they are represented as $[p_{k_{n_k}+1}, p_{(k+1)_1-1}]$. For simplicity, we use N_k and $C_{k,k+1}$ to represent the set of points in the kth node and the connection part between the kth node and $(k+1)$th node, as shown in Fig. 7(b). For a node $N_k, k \in [1, m]$ (or a connection part $C_{k,k+1}$), we use the linear interpolation algorithm shown in Eq. (2) to align the sensor data of different gestures in the kth node (or $C_{k,k+1}$). That is to say, in a node N_k, the number of data points from different gestures is the same, while in a connection part $C_{k,k+1}$, the number of data points from different gestures is the same. In regard to applying the linear interpolation in N_k or $C_{k,k+1}$, everytime we align the sensor data in one dimension, e.g., coordinates in x-axis, coordinates in y-axis, touch areas along the time, etc. Finally, we can align all the sensor data in space domain.

By introducing the spatial alignment, the angular velocity in Fig. 6(b) is transformed into Fig. 7(c), which can solve the problem of the time difference between gestures corresponding to the same graphic pattern. When compared with the temporal alignment result shown in Fig. 6(c), the spatial alignment result in Fig. 7(c) keeps a higher consistency of gestures from the same user. To quantitatively measure the similarity (difference) of the spatial (temporal) aligned sensor data, we collect 50 samples of gesture 'L' performed by the same user and calculate the RMSE value of the sensor data. The average RMSE value of the temporal and the spatial aligned sensor data is 0.282 (standard deviation=0.081) and 0.157 (standard deviation=0.045), respectively. As mentioned before, low RMSE value means high similarity. Therefore, our spatial alignment method can keep the stability among gestures and reduce the intra-class difference for better user authentication.

3.4 Fine-grained Modeling for Gesture Segmentation

The touch gestures in nodes and that out of nodes have different properties, especially for the gestures located in turning points. Therefore, after sensor data alignment in space domain, we further segment the touch gesture into several sub-gestures, to highlight the sub-gestures which contribute more for user authentication. As shown in Fig. 8, we illustrate the mean acceleration in x-axis of 50 samples corresponding to a whole touch gesture 'L', the ith and the jth segmented sub-gesture of 'L' from different users, respectively. The overlap in Fig. 8(a) indicates the low discriminability in whole gestures. However, the little overlap in Fig. 8(b) indicates that the high discriminability in the ith segmented sub-gesture, while the large overlap in Fig. 8(c) indicates the very poor discriminability in the jth segmented sub-gesture. It means that the gesture in different segments has different stability and discriminability. Thus it is necessary and meaningful to segment the whole touch gesture into



(a) Acceleration in x-axis of the entire gesture

(b) Acceleration in x-axis of i-th sub-gesture

(c) Acceleration in x-axis of j-th sub-gesture

Fig. 8. Distribution of feature value for the whole gesture and sub-gestures

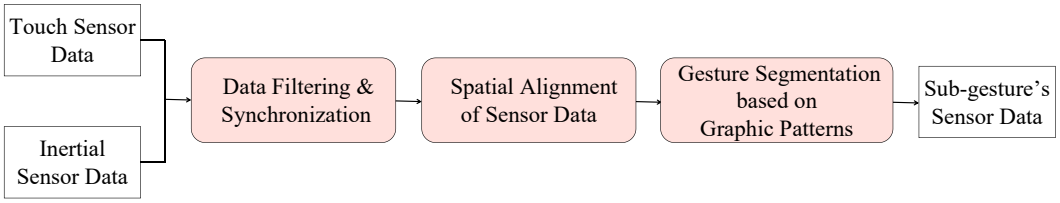


Fig. 9. Gesture segmentation scheme

sub-gestures, to extract the sub-gestures having a good stability for the same user and the sub-gestures having a good discriminability for different users.

To segment the touch gesture, we need to split the inertial sensor data and touch sensor data into each sub-gesture. As shown in Fig. 9, the gesture segmentation consists of three steps, i.e., data filtering and synchronization, spatial alignment of sensor data, gesture segmentation based on the graphic pattern. **Firstly**, we use a moving average filter to remove the high-frequency noises in inertial sensor data and touch sensor data. Besides, considering the difference between the sampling rates of touch sensor (i.e., 60 Hz) and inertial sensor (i.e., 100 Hz), we introduce the linear interpolation described in Eq. (2) to synchronize the sensor data, and make them have a uniform sampling rate, i.e., 100 Hz. **Secondly**, we use the spatial alignment method described in Section 3.3 to align the sensor data of gestures corresponding to the same graphic pattern, to keep the stability of gestures from the same user. **Thirdly**, we use the layout of lock screen, i.e., the locations of nodes, to segment the touch gesture as in-node sub-gestures and out-node sub-gestures by turns, as the blue segments and red segments shown in Fig. 7(b). As mentioned in Section 3.3, we use $[p_{k_1}, p_{k_{n_k}}]$ to represent the in-node points in the kth node. Accordingly, the time of the first, the last data point occurring in kth node is represented as t_{k_1}, t_{n_k}, respectively. Therefore, the sensor data occurring in $[t_{k_1}, t_{k_{n_k}}]$ is split into the sub-gesture located in the kth node, while the sensor data occurring in $[t_{k_{n_k+1}}, t_{(k+1)_1-1}]$ is split into the sub-gesture located in the connection part between the kth node and the $(k+1)$th node.

4 DATA ANALYSIS AND FEATURE SELECTION

According to the observations in Section 3, the touch sensor data and inertial sensor data of touch gestures can be used for authentication, since the sensor data shows the similarity of gestures from the same user and the difference of gestures from different users. However, the uncertainty of user behaviors may reduce the intra-class similarity and reduce the inter-class difference. Therefore, it is necessary to analyze the sensor data detailedly and select effective features from sensor data, to improve the stability of gestures from the same user while improving the discriminability of gestures from different users.

4.1 Candidate Feature Set Generated by Sensor Data

As described in Section 3.4, a touch gesture is segmented into multiple sub-gestures, since sub-gestures have different contributions for user authentication. Therefore, we analyze the sensor data and extract features in each sub-gesture and the whole touch gesture. The features or sub-gestures which decrease the performance of user authentication will be eliminated. During a touch gesture, at time $t_i, i \in [1, n]$, the collected sensor data after synchronization which is described in Section 3.4 is represented as $\{(x_i, y_i), s_i, (a_{x_i}, a_{y_i}, a_{z_i}), (g_{x_i}, g_{y_i}, g_{z_i})\}$. Here, (x_i, y_i) means the coordinate of fingertip on the screen, s_i means the touch area size, $(a_{x_i}, a_{y_i}, a_{z_i})$ means the linear acceleration in x-axis, y-axis, z-axis of the device's coordinate system, $(g_{x_i}, g_{y_i}, g_{z_i})$ means the angular velocity in x-axis, y-axis, z-axis of the device's coordinate system. Suppose there are w sub-gestures in the touch gesture, we use n'_k to represent the number of data points in the kth sub-gesture, then the sensor data in the kth sub-gesture is represented as $D_k = \{(x_i, y_i), s_i, (a_{x_i}, a_{y_i}, a_{z_i}), (g_{x_i}, g_{y_i}, g_{z_i})\}, i \in [\sum_{j=1}^{k-1} n'_j + 1, \sum_{j=1}^{k} n'_j], k \in [2, w]$. When k=1, $i \in [1, n'_1]$. With the sensor data in the whole gesture and each sub-gesture, we first provide the candidate feature set inferred from the touch sensor data and the inertial sensor data, i.e., the touch feature set, and the motion feature set.

Candidate touch feature set: To describe both the temporal and spatial information of a touch gesture on the screen, we select seven types of features from the touch sensor data. The features are selected from both the whole gesture and the sub-gestures. In each type of feature, there are one or more feature elements, as described below.

- Duration: We use $\Delta_t = t_n - t_1$ to represent the time of duration for a whole touch gesture.
- Position: We use the coordinates (x_1, y_1), (x_n, y_n) to represent the start and the end of a touch gesture on the screen.
- Displacement: We use $\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}$ to represent the displacement between the start and the end of a touch gesture on the screen.
- Distance: We use $\sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$ to represent the moving distance of the fingertip on the screen.
- Touch size: We use $\bar{s}_k, k \in [1, w]$ to represent the mean touch area size in each sub-gesture. Here, $\bar{s}_k = \frac{\sum_{i=k_1}^{i=k_2} s_i}{n'_k}$, where $k_1 = \sum_{j=1}^{j=k-1} n'_j + 1$ and $k_2 = k_1 + n'_k$.
- Velocity: We use $\bar{v}_k, k \in [1, w]$ to represent the mean velocity in each sub-gesture. Here, $\bar{v}_k = \frac{\sum_{i=k_1}^{i=k_2} v_i}{n'_k}$, where $k_1 = \sum_{j=1}^{j=k-1} n'_j + 1$ and $k_2 = k_1 + n'_k$. In regard to the v_i, it is calculated with Eq. (5).

$$v_i = \begin{cases} \dfrac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{t_i - t_{i-1}} & ,i \in [2, n] \\ 0 & ,i = 1 \end{cases} \tag{5}$$

- Direction: We use $\bar{\theta}_k, k \in [1, w]$ to represent the mean moving direction in each sub-gesture. Here, $\bar{\theta}_k = \frac{\sum_{i=k_1}^{i=k_2} \theta_i}{n'_k}$, where $k_1 = \sum_{j=1}^{j=k-1} n'_j + 1$ and $k_2 = k_1 + n'_k$. In regard to θ_i, it is calculated with Eq. (6), where the vector $\vec{v_i} = (x_i - x_{i-1}, y_i - y_{i_1})$ and the vector $\vec{v_0} = (0, -1)$.

$$\theta_i = \begin{cases} \arccos \dfrac{\vec{v_i} \cdot \vec{v_0}}{|\vec{v_i}| \cdot |\vec{v_0}|} & ,x_i - x_{i-1} \leq 0, \\ 360 - \arccos \dfrac{\vec{v_i} \cdot \vec{v_0}}{|\vec{v_i}| \cdot |\vec{v_0}|} & ,x_i - x_{i-1} > 0. \end{cases} \tag{6}$$

Candidate motion feature set: As described in Section 3, the device's motion is mainly caused by the pressure of fingertip touching on the screen, thus the inertial sensor data is also used for selecting features related to the device's motion. Specifically, we select six types of motion features for each sub-gesture, as described below.

- Rotation angle in x-axis, y-axis and z-axis: We use $r_{x_k}, r_{y_k}, r_{z_k}, k \in [1, w]$ to represent the rotation angle in each sub-gesture along the x-axis, y-axis and z-axis, respectively. In the kth sub-gesture, $r_{x_k} = \sum_{i=k_1}^{i=k_2} g_{x_i} \delta t$. Here, $k_1 = \sum_{j=1}^{j=k-1} n'_j + 1$, $k_2 = k_1 + n'_k$, $\delta t = \frac{1}{r_s}$, while r_s is the synchronized sampling rate, i.e., 100 Hz. Similarly, we can also calculate r_{y_k} and r_{z_k}.
- Moving speed in x-axis, y-axis and z-axis: We use $s_{x_k}, s_{y_k}, s_{z_k}, k \in [1, w]$ to represent the final moving speed in each sub-gesture along the x-axis, y-axis and z-axis, respectively. In the kth sub-gesture, $s_{x_k} = \sum_{i=k_1}^{i=k_2} a_{x_i} \delta t$. Here, $k_1 = \sum_{j=1}^{j=k-1} n'_j + 1$, $k_2 = k_1 + n'_k$, $\delta t = \frac{1}{r_s}$, while r_s is the synchronized sampling rate, i.e., 100 Hz. Similarly, we can also calculate s_{y_k} and s_{z_k}.

It is worth noting that we use the integration of linear acceleration and gyroscope data to select features, while not using the linear acceleration and angular velocity directly. This is because the micro movement of device caused by the fingertip will introduce the uncertainty of sensor data at a time, e.g., the small sensor data is easy to change between positive or negative. In addition, we apply integration on the collected sensor data without coordinate system transformation, this is mainly because the duration of a sub-gesture in touch gestures is usually short, i.e., about 100 ms. During a sub-gesture, the orientation of the mobile device changes little, thus we calculate the rotation angle and moving speed by integration in the device's coordinate system.

4.2 Feature Analysis and Determination

To select effective features of a touch gesture for user authentication, we should consider the intra-class stability of the same user and inter-class discriminability of different users at the same time. In TouchID, we utilize the Fisher Score technique [18, 53] to select effective features. For each feature, we can calculate a Fisher Score, then we select the features with high Fisher Scores. Specifically, Fisher Score consists of two metrics: inter-user metric S_b and intra-user metric S_w. For the kth feature, $S_b(k)$ represents the inter-user discriminability while $S_w(k)$ represents the intra-user stability. Then the Fisher Score Fisher(k) for the kth feature is calculated with Eq. (7).

$$Fisher(k) = \frac{S_b(k)}{S_w(k)} = \frac{\sum_{i=1}^{c} p_i(\mu_i - \mu)^2}{\sum_{i=1}^{c} p_i \sigma_i^2} \tag{7}$$

where c means the number of users in the data set, p_i denotes the number of samples of the i-th user, μ_i and σ_i^2 denote the mean and variance value of k-th feature element for the i-th user, μ denotes the mean value of the k-th feature element for all users. For each feature, the larger Fisher Score means this feature has better intra-user stability and inter-user discriminability. To select effective features with larger Fisher Scores, we collect 1640 samples from 41 volunteers for a gesture (i.e., gestures corresponding to a same graphic pattern), and then set the threshold $\epsilon_f = 1.0$ for the Fisher Score empirically. That is to say, the feature whose Fisher Score is larger than 1.0 will be chosen for user authentication.

4.3 Adaptive Feature Selection for Gestures with Different Topological Structures

According to Section 4.2, the Fisher Score can be used to select effective features for user authentication. However, in real scenarios, the gestures corresponding to different graphic patterns (e.g., 'L' and 'M') have a different number of sub-gestures, thus the number of candidate features is different. Thus the features selected for 'L' may not be suitable for 'M'. This means that we need to select features for the touch gesture corresponding to a different graphic pattern separately. Take the 3×3 grid shown in Fig. 2 as an example, the number of different graphic patterns on the lock screen can be very large. Consequently, selecting features for gestures corresponding

to each graphic pattern is unpractical. Therefore, we propose a four-part feature selection method to solve the above problems, as described below.

Sub-gestures are classified into four parts: Although the topological structures of gestures can be different, we find that each gesture can be described with four parts, i.e., the start node, the end node, the turning node(s), the smooth paths. In a touch gesture, the first sub-gesture is the start node, the last sub-gesture is the end node. In regard to the turning node, we use $O_i(x_{o_i}, y_{o_i})$, $O_j(x_{o_j}, y_{o_j})$, $O_k(x_{o_k}, y_{o_k})$ to represent the centers of three consecutive nodes in the moving trajectory, if O_i, O_j and O_k satisfy Eq. (8), the node containing O_j is the turning node. Here, $\vec{v}_{ji} = (x_{o_i} - x_{o_j}, y_{o_i} - y_{o_j})$, $\vec{v}_{jk} = (x_{o_k} - x_{o_j}, y_{o_k} - y_{o_j})$, $\epsilon_r = 0^\circ$.

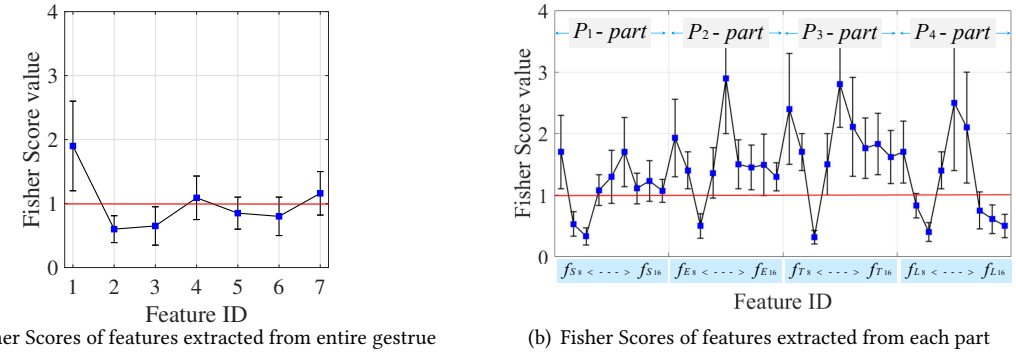
$$|\arccos \frac{\vec{v}_{ji} \cdot \vec{v}_{jk}}{|\vec{v}_{ji}| \cdot |\vec{v}_{jk}|} - \pi| > \epsilon_r \tag{8}$$

After eliminating the start node, the end node, the turning node(s), the remaining parts of the touch gesture are the smooth paths. Take the gesture in Fig. 7(b) as an example, N_1 is the start node, N_7 is the end node, N_3 is the turning node, while the remaining parts of the moving trajectory are treated as smooth paths. For a specific smooth path, its two endpoints can be a start node and a turning node, a turning node and an end node, or two turning nodes. Besides, each smooth path can contain one or more sub-gestures. It is worth noting that there can be more than one turning node or smooth path. For illustration, we use P_1, P_2, P_3, and P_4 to represent the start node, the end node, the turning node set, and the smooth path set, respectively.

Feature selection in each part: In each part P_z, $z \in [1, 4]$, we select effective features based on the Fisher Score. For the start node P_1 and the end node P_2, each of them corresponds to only one sub-gesture, we first select the candidate feature set based on Section 4.1, and then use the Fisher Score to select the effective features for P_1 and P_2, respectively. However, for the turning node set P_3 and the smooth path set P_4, due to the difference of topological structures in touch gestures, i.e., the graphic patterns are different, the number of sub-gestures in P_3 and that in P_4 can be different. Therefore, we need to select a fixed feature set from the variable number of sub-gestures. To solve this problem, for the part P_3 (or P_4), we first select candidate features mentioned in Section 4.1 and calculate Fisher Scores of features in each sub-gesture of P_3 (or P_4), then we average the Fisher Scores corresponding to the same type of feature for feature selection. For example, in gesture 'L', P_4 consists of six sub-gestures: the connection line $C_{1,4}$, the node N_4, the connection line $C_{4,7}$, the connection line $C_{7,8}$, the node N_8, the connection line $C_{8,9}$. We use $f_{i_{(j)}}$ to represent the ith candidate feature in the jth ($j \in [1, 6]$) sub-gesture, while using $F_{i_{(j)}}$ to represent the Fisher Score of the candidate feature $f_{i_{(j)}}$. Then, we average the six Fisher Scores of ith candidate feature from six sub-gestures in P_4 as $F_i = \frac{\sum_{j=1}^{j=6} F_{i_{(j)}}}{6}$. In this way, the number of average Fisher Scores F_i in P_4 is equal to the number of candidate features in a sub-gesture, no matter how many sub-gestures are in P_4. After that, we use the average Fisher Score F_i to determine whether to select the ith candidate feature. Therefore, whatever the graphic pattern is, the selected feature set (i.e., types of features) in a part P_z, $z \in [1, 4]$ is fixed.

However, it is worth noting that the fixed set of features in P_z does not mean the fixed number of features in P_z. For example, in gesture 'L', there are six sub-gestures in P_4, if a type of feature (e.g., pth feature) is selected, it means all (i.e., six) features of the same type , i.e., the pth feature f_{p_j}, $j \in [1, 6]$ from each sub-gesture, are selected for P_4. While in gesture 'M', there are eight sub-gestures in P_4, if a type of feature is selected, eight features of the same type from each sub-gesture are selected for P_4. Nevertheless, each part has its fixed feature set and each sub-gesture in the same part selects the same features. In this way, we can obtain the features of gestures corresponding to different graphic patterns conveniently.

Final features for each gesture: Until now, we have selected effective features for each part. Specifically, based on extensive experiments, we use ϵ_f to represent the threshold for Fisher Scores, only the feature whose Fisher Score is larger than ϵ_f will be selected. Besides the features in segmented parts, we also add the features



(a) Fisher Scores of features extracted from entire gestrue

(b) Fisher Scores of features extracted from each part

Fig. 10. Fisher Scores of all features

Table 1. Selected features in TouchID using Fisher Score

Feature Type	Description	Feature Identifer	Selected Features
Duration	Duration time during one touch gesture	f_1	f_1
Position	X and Y coordinates of the first sampling point	f_2, f_3	none
Position	X and Y coordinates of the last sampling point	f_4, f_5	f_4
Displacement	The distance between starting and ending points	f_6	none
Distance	The trajectory length of one gesture	f_7	f_7
Touch size	Mean of touch area size series during each part	$f_{S_8}, f_{E_8}, f_{T_8}, f_{L_8}$	$f_{S_8}, f_{E_8}, f_{T_8}, f_{L_8}$
Velocity	Mean of touch velocity series during each part	$f_{S_9}, f_{E_9}, f_{T_9}, f_{L_9}$	f_{E_8}, f_{T_8}
Direction	Mean of direction angel series during each part	$f_{S_{10}}, f_{E_{10}}, f_{T_{10}}, f_{L_{10}}$	none
X-Rotation change	Rotation change in x-axis during each part	$f_{S_{11}}, f_{E_{11}}, f_{T_{11}}, f_{L_{11}}$	$f_{S_{11}}, f_{E_{11}}, f_{T_{11}}, f_{L_{11}}$
Y-Rotation change	Rotation change in y-axis during each part	$f_{S_{12}}, f_{E_{12}}, f_{T_{12}}, f_{L_{12}}$	$f_{S_{12}}, f_{E_{12}}, f_{T_{12}}, f_{L_{12}}$
Z-Rotation change	Rotation change in z-axis during each part	$f_{S_{13}}, f_{E_{13}}, f_{T_{13}}, f_{L_{13}}$	$f_{S_{13}}, f_{E_{13}}, f_{T_{13}}, f_{L_{13}}$
X-Speed change	Speed change in x-axis during each part	$f_{S_{14}}, f_{E_{14}}, f_{T_{14}}, f_{L_{14}}$	$f_{S_{14}}, f_{E_{14}}, f_{T_{14}}$
Y-Speed change	Speed change in y-axis during each part	$f_{S_{15}}, f_{E_{15}}, f_{T_{15}}, f_{L_{15}}$	$f_{S_{14}}, f_{E_{14}}, f_{T_{14}}$
Z-Speed change	Speed change in z-axis during each part	$f_{S_{16}}, f_{E_{16}}, f_{T_{16}}, f_{L_{16}}$	$f_{S_{16}}, f_{E_{16}}, f_{T_{16}}$

(f_{S_8} means the feature extracted from the start node, f_{E_8} means the feature extracted from the end node, f_{T_8} means the feature(s) extracted from the turning node(s), f_{L_8} means the features extracted from the smooth paths)

related to the whole gestures, i.e., duration, position, displacement, distance mentioned in section 4.1, and then use the Fisher Score to select the effective features from them, as described in Section 4.2. In Fig. 10, we show the Fisher Scores of all the candidate features, where Fig. 10(a) and Fig. 10(b) show the Fisher Scores of features selected from the whole touch gesture and each part respectively. The special meaning of each feature is shown in Table 1. Fig. 10 indicates that different features have different performances in the stability and the discriminability. Based on the threshold of Fisher Scores, the final selected features for each gesture are shown in Table 1. In this way, we can select fixed types of features for gestures with different graphic patterns for user authentication.

5 SYSTEM DESIGN

The key idea of TouchID is to authenticate users based on stable and unique behavioral features during a touch gesture. Fig. 11 depicts the system architecture of TouchID, which consists of four components: *data collection and pre-processing*, *modeling of touch gestures*, *training process*, and *authentication process.*

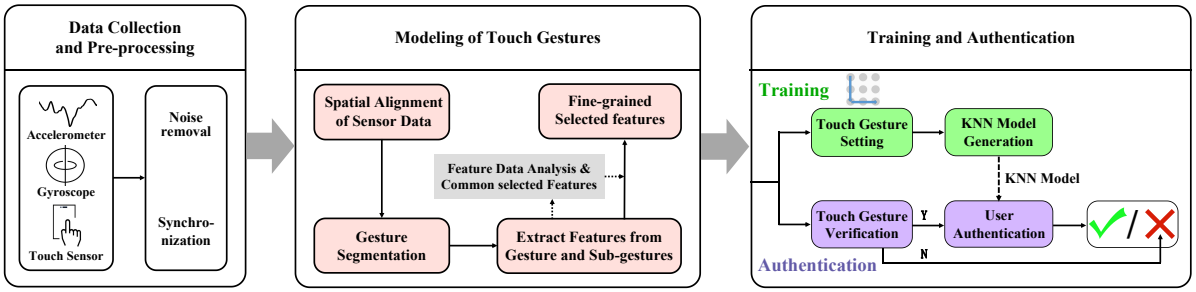


Fig. 11. System architecture

5.1 Data Collection and Pre-processing

In TouchID, the sensor data of a touch gesture is collected using a mobile device (i.e., smartphone) equipped with the touch sensor and inertial sensor. We collect coordinate and touch area size from the touch sensor. At the same time, we collect the linear acceleration from the accelerometer and the angular velocity from the gyroscope. The sampling rates of the touch sensor, the accelerometer, gyroscope are set to 60 Hz, 100 Hz, 100Hz, respectively. After data collection, we first use a moving average filter to remove the high-frequency noise in the sensor data. Then, considering the touch sensor and inertial sensor have different sampling rates, we synchronize the inertial sensor data and touch sensor data, to make them achieve the same sampling rate, i.e., 100 Hz, as described in Section 3. The synchronized sensor data will be used in the following process.

5.2 Modeling Touch Gestures

After data pre-processing, we aim to extract effective features from synchronized data to represent a touch gesture. To achieve this goal, there are three main steps to be followed: spatial alignment of sensor data, gesture segmentation, data analysis and feature selection. Given synchronized sensor data of a touch gesture corresponding to a graphic pattern, as described in Section 3, we first use coordinates to align the sensor data, then we segment the sensor data of the entire touch gesture into four parts, where a part contains one or more sub-gestures. After that, we analyze the sensor data both in the whole touch gesture and each part, and then select effective features from the sensor data based on the Fisher Score, as described in Section 4. Finally, we organize the selected features into a feature vector for the following training or authentication.

5.3 Training Process

In the training process, we first detect the graphic pattern (e.g., the node sequence "1-4-7-8-9" represents the graphic pattern of 'L' in Fig. 7(a)) of a touch gesture in the spatial alignment step, and then perform the training process for the gestures corresponding to the same graphic pattern. We use the selected feature vectors to train a one-class classifier for user authentication, since we can only obtain the training data from the legitimate user. Specifically, we get N feature vectors corresponding to N training samples and the one-class k-Nearest Neighbors (kNN) classifier. For one-class kNN, it utilizes the density [7] as the metric to classify the legitimate user and attackers. As shown in Eq. (9), the density M is the average Manhattan distance between any two feature vectors, where d_{ij} means the Manhattan distance between the ith feature vector and the jth feature vector.

$$M = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}}{C_N^2} \tag{9}$$

To avoid that the feature element having large data dominate the d_{ij}, we normalized all the feature vectors before calculating the density M, i.e., the value of each feature element ranges in $[0, 1]$. Based on extensive experiments, the parameter 'k' for kNN is set to 4, i.e., one-class 4NN classifier.

5.4 Authentication Process

In the authentication process, we first detect the graphic pattern of a testing sample in the spatial alignment step. If the graphic pattern of this testing sample is different from that of gesture to be authenticated, our system will directly reject this authentication request without performing further operations. Otherwise, given a testing sample of a touch gesture, TouchID performs the process of modeling the gesture and selecting features, and then normalizes the features to form a feature vector $\vec{f}'$. After that, we use the feature vector $\vec{f}'$ and the trained one-class kNN classifier corresponding to the same graphic pattern for user authentication. Specifically, we calculate the Manhattan distance between the feature vector $\vec{f}'$ and each feature vector in the training samples. Then we get k smallest Manhattan distances and calculate the mean of these k smallest distances as m. Finally, the one-class KNN classifier determines the identity of this testing sample as follows:

$$Identity = \begin{cases} legitimate\ user, & m \leq \lambda \times M \\ attacker, & m > \lambda \times M \end{cases} \tag{10}$$

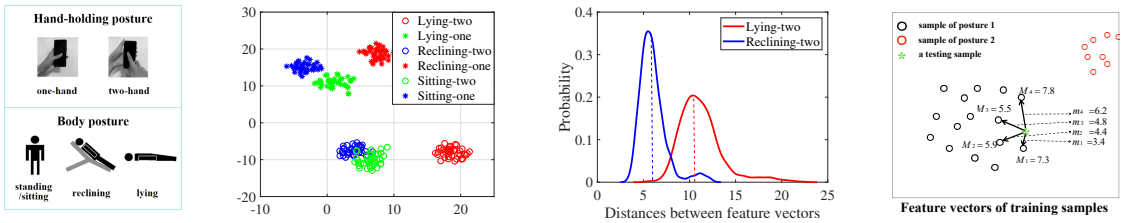
where λ is the tradeoff factor between usability and security, it can be set to a larger value for better usability but worse security, and vice verse.

6 AUTHENTICATING USERS UNDER MULTI-POSTURE SCENARIOS

In the actual environment, the postures of a user performing the same touch gesture may change, mainly including the change of hand-holding posture and body posture. In order to ensure the scalability of our system, it is necessary to model touch gestures under different postures without sacrificing the authentication performance. As shown in Fig. 12(a), for hand-holding postures, we consider two most common cases: one-hand posture and two-hand posture. For body postures, we consider three common body postures in daily life: standing/sitting, reclining, and lying down. By combining the body postures and hand postures, we use the format 'X-Y' to represent a unique posture, where the body posture is 'X' and the hand-holding posture is 'Y', e.g., 'Lying-one'. Note that the distance between feature vectors in the following description refers to the Manhattan distance.

6.1 Touch Gesture Inconsistencies Caused by Different Body Postures and Hand Postures

The user behaviors can be inconsistent in different postures. For example, in the two-hand posture, the user will use the index finger to swipe gestures, while in the one-hand posture, the user tends to use the thumb to swipe gestures. To illustrate this inconsistency, we collected 240 samples of gesture 'L' from the same user in 2 different hand-holding postures and 3 different body postures, there are a total of six (2×3) postures, each with 40 training samples. Then we extract the feature vector from each gesture sample and visualize the feature vectors using the t-SNE technique [50], which can reduce the dimension of data. As shown in Fig. 12(b), the data under the same 'X-Y' posture is clustered together and has an obvious distance from the data under other postures. Besides, the feature vectors under different postures have different classification thresholds (i.e., the density shown in Eq. (9)), which mean the average distance between the feature vectors in a cluster. For example, the classification threshold of 'Lying-two' posture is 6.01 while that of 'Reclining-two' posture is 10.32, as shown in Fig. 12(c). If we use a traditional KNN classifier to model touch gestures under multi-posture scenarios, only a single classification threshold M will be calculated based on all training samples, while ignoring the differences of postures in samples. It will fail to represent the classification thresholds under several different postures, thus leading to the decrease of user authentication performance under different postures.



(a) Typical postures (b) Sample distributions of different postures using the t-SNE technique (c) Distance distribution for two postures (d) Multi-threshold kNN

Fig. 12. Illustration of multi-posture scenarios

6.2 Multi-Posture Modeling Based on Multi-threshold KNN

To solve the above problem, we propose a multi-threshold kNN, which adaptively selects a different classification threshold for a different posture. Specifically, **in the training phase**, we calculate the classification threshold (i.e, the density) for each training sample, instead of calculating only one average threshold (i.e., Eq. (9)) for all training samples. To calculate the classification threshold for the training sample, we calculate the Manhattan distance between the feature vector of this training sample and the other feature vectors. Then, we select N_m smallest distances and average the distances as its classification threshold, where N_m is smaller than the number of training samples of a touch gesture under a unique posture. **In the authentication phase**, for the feature vector of a testing sample, we calculate the Manhattan distance between the testing feature vector and each training feature vector, and select k smallest distances $m_i, i \in [1, k]$. Suppose the parameter k for kNN is 4, then the four smallest distances are m_1, m_2, m_3, m_4, as shown in Fig. 12(d). In regard to the training feature vectors corresponding to m_1, m_2, m_3, and m_4, the classification thresholds of them are represented as M_1, M_2, M_3, and M_4. Then, we calculate the distance between the testing feature vector and the training feature vectors as $\bar{m} = \frac{\sum_{i=1}^{i=4} m_i}{4}$, while calculating the classification threshold of the classifier as $\bar{M} = \frac{\sum_{i=1}^{i=4} M_i}{4}$. Finally, we use Eq. (10) to compare $\bar{m}$ and $\bar{M}$ to authenticate a user. In this way, we can perform user authentication under different postures.

6.3 Incremental Authentication under New Scenarios

In the previous subsections, we mainly focus on several common postures in performing touch gestures. In fact, the user can perform touch gestures under arbitrary postures, which can be different from the typical postures in Fig. 12(a). Therefore, TouchID is expected to flexibly adapt to new postures. To address this issue, we propose an incremental authentication scheme, based on the multi-threshold kNN described in Section 6.2. Specifically, for a new posture, in the training phase, TouchID adds the training data of the touch gesture under the new posture and calculates the classification threshold for each new training sample. While in the authentication phase, TouchID calculates the distances between the testing sample and training samples, and then compares the distances with the classification thresholds for user authentication, as described in Section 6.2. It is noteworthy that TouchID only needs to calculate the classification thresholds for new training samples, while not retraining the classifier with all training samples. Therefore, in this way, TouchID can incrementally add the training samples under a new posture, and authenticate users under new postures. It is worth noting that incremental authentication can not only benefit the authentication under new postures but also benefit the authentication in a long time. Specifically, if there exists a certain variation of user behaviors in a long time, it is possible to add the near-term training samples of gestures to update the classification thresholds for multi-threshold kNN classifier, which will be used for user authentication.

Fig. 13. Touch gestures used in our experiment

7 EVALUATION

We implement TouchID on an Android-powered smartphone, i.e., Samsung Galaxy S9 with a 5.8-inch screen, and TouchID works in an online way. To evaluate the performance of TouchID in user authentication, we introduce the experiment setting and evaluation metrics, and then conduct a lot of experiments. Firstly, we show the overall performance of TouchID. Secondly, we evaluate each component of TouchID and observe the impact of parameters, to analyze the efficiency of the proposed solution. Thirdly, we evaluate the performance of TouchID under complex scenarios. Finally, we test the performance of TouchID under attacks and compare TouchID with the existing research work.

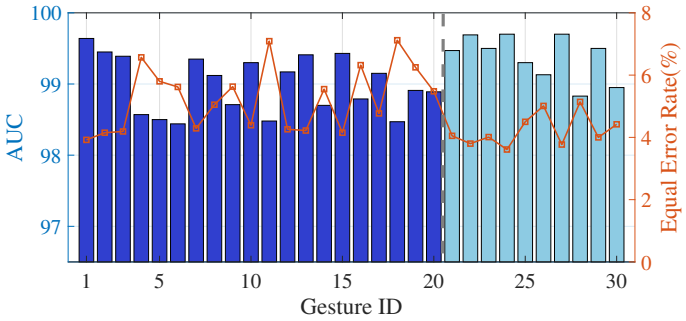
7.1 Experiment Setting

On commercial Android smartphones, we implement an application to collect sensor data, train classifiers, and authenticate users, as shown in Fig. 1. The smartphone we used is Samsung Galaxy S9 with a 5.8-inch screen, 2.8 GHz quad-core processor and 4 GB RAM. In our experiment, the sampling rates of the touch sensor and the inertial sensor were set to 60 Hz and 100 Hz respectively. During the process of data collection, we recruit 41 volunteers (28 males and 13 females) with the age ranging from 19 to 38 to perform gestures for evaluation, where all volunteers have more than one year experience in using smartphones. In regard to the unlock gestures (i.e., graphic patterns), 20 kinds of gestures are directly selected or slightly changed from the most commonly used unlock gestures reported on the website [3], and 10 kinds of gestures are customized gestures which are generated by users, as shown in Fig. 13. We collect 40 samples of each gesture from every user, the entire data collection process lasts more than two months. Finally, we collect a total of 49, 200 samples of all 30 touch gestures.

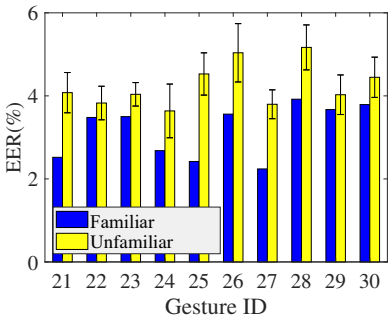
After data collection, we conduct the training and testing processes based on the collected data to evaluate system performance. For all users, we choose one user as the legitimate user and treat the remaining users as attackers. In the training phase, we randomly select a part of samples from the legitimate user, and train the one-class kNN classifier. Then in the testing phase, we use the remaining samples from legitimate user as the testing samples for the legitimate user and use the samples from attackers as the testing samples for attackers. During the experiments, we exchange the legitimate user so that every user acts as the legitimate user once. Furthermore, we also repeat the evaluation process for each legitimate user 10 times, then we calculate the mean value of evaluation metrics in all cases as the reported result.

7.2 Evaluation Metrics

We utilize the False Negative Rate (FNR) and False Positive Rate (FPR) as the main evaluation criteria for our experiment. Given the true positive (TP), false positive (FP), true negative (TN), and false negative (FN), where TP and TN mean positive and negative examples classified correctly, FP and FN mean positive and negative



(a) EER and AUC results of 30 gestures



(b) EER results of 10 user-generated gestures performed by the users familiar or unfamiliar with gestures

Fig. 14. Overall authentication performance

examples misclassified, we can calculate the FNR and FPR as below:

$$FNR = \frac{FN}{TP + FN} \tag{11}$$

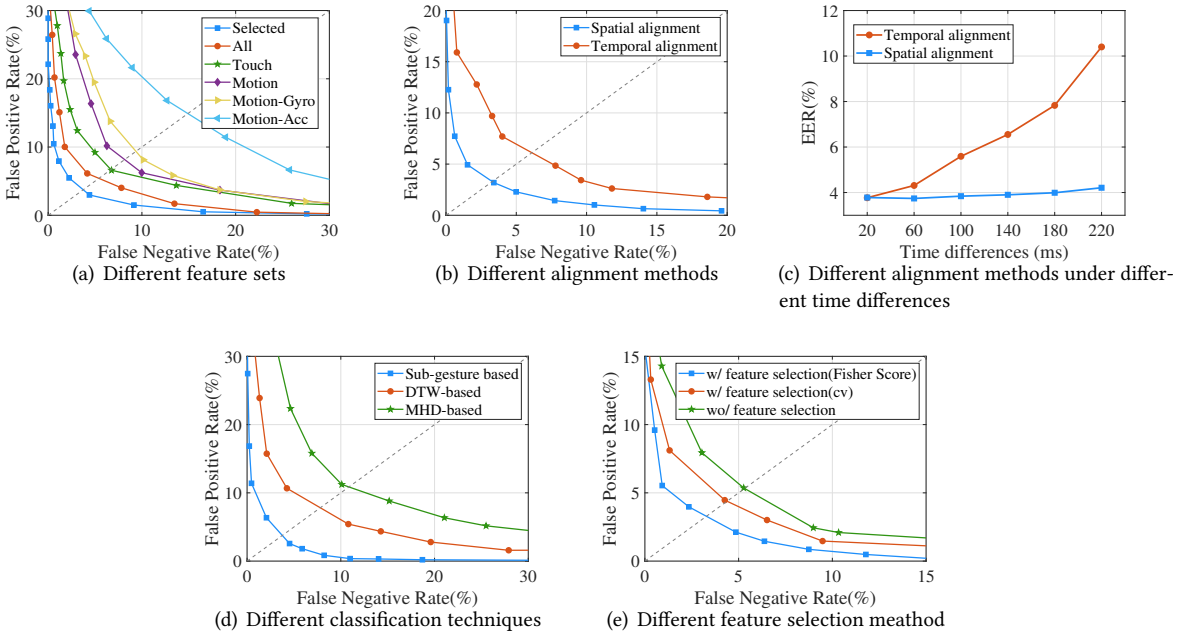
$$FPR = \frac{FP}{TN + FP} \tag{12}$$

Consequently, FNR is the rate that the legitimate user is misclassified as an attacker, and FPR is the rate that an attacker is misclassified as the legitimate user. When considering the FNR and FPR at the same time, we introduce the Equal Error Rate (EER) where FNR equals FPR. In addition, we also calculate the area under the receiver operating characteristic curve (AUC), which measures the system's effectiveness of distinguishing the negative and positive samples.

7.3 Performance of User Authentication

In this subsection, we evaluate the overall performance of our system, based on the collected data of 30 gestures from 41 users. Unless otherwise specified, the features used for training and testing are the feature subset after feature selection, as shown in Table 1. The number of training samples for each touch gesture is set to 15. Fig. 14(a) shows the EER and AUC results of 30 gestures, where the performance results are the average value over 41 users. For the first 20 commonly used gestures, the average EER is 5.24%, while for the last 10 user-generated gestures, the EER is 4.23%. Overall, our system has a good performance of user authentication and the average EER for all 30 gestures is 4.90%. Besides, we can adjust EER (i.e., FNR and FPR pairs) as needed, by varying the value of λ, which is the parameter of one-class kNN classifier as described in Section 5.4. A higher λ indicates that our system rejects fewer requests from legitimate user, which corresponds to lower FNR but higher FPR and means better usability but worse security, and vice versa. When considering the security in actual scenarios, we set λ to 0.94 when the average FPR for all gestures is less than 4%.

In addition to adjusting λ, we find that the familiarity of performing a touch gesture can also affect the authentication performance. In the experiments, we compare the authentication performance of the user who generates the customized unlock gesture and other users who are unfamiliar with this gesture. Specifically, for the ten users who generate the 10 customized gestures in Fig. 13, each user is given one week to repeatedly unlock the smartphone 20 times a day with her/his generated unlock gesture. After that, for each customized gesture in Fig. 13, we collect 40 samples from each user. As shown in Fig. 14(b), the EER of the user familiar with the gesture, i.e., who generates the gesture, is lower than that of other people unfamiliar with the gesture. Take the 21st unlock gesture as an example, the EER of the user familiar with gesture in only 2.52%, while the average



(a) Different feature sets (b) Different alignment methods (c) Different alignment methods under different time differences

(d) Different classification techniques (e) Different feature selection meathod

Fig. 15. Effectiveness of each system component

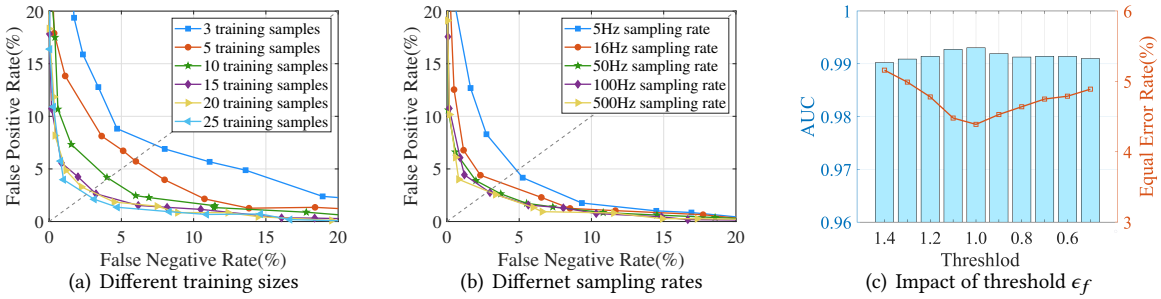
EER of the other 40 users unfamiliar with the gesture is 4.08%. It indicates that when the user is proficient to perform the gesture, e.g., training with one week, she/he will form her/his unique and stable habits, which are difficult to be mimicked. The uniqueness and stability in touch gestures can be compared to that of signatures. Thus the familiarity of touch gestures after practice can improve authentication performance.

7.4 Effectiveness of Each System Component

In this experiment, we evaluate the effectiveness of four major system components, including sensor combination, spatial alignment, gesture segmentation, and feature selection. To evaluate each system component, we use the samples of the 10th gesture in Fig. 13 from 20 users. Due to a smaller data set, the EER results in this subsection and later subsections can not be directly compared to the results reported in Section 7.3.

Sensor Combination. To explore the impact of different feature set on authentication performance, we considered five different feature set extracted from three sensors used in our system: (1)Touch: extracted from the touch sensor; (2)Motion: extracted from the inertial sensor; (3)Motion-Gyro: extracted from the gyroscope; (4)Motion-Acc: extracted from the accelerometer; (5)All: combining all the features in (1) and (2); (6)Selected: feature subset selected by a Fisher Score from (5). As shown in Fig. 15(a), feature set (5) combining the features of the touch sensor and inertial sensor brings performance improvements when compared with performances using feature subset (1), (2), (3), and (4). This indicates that our method combining touch information with inertial sensor information can better model the touching behavior. In regard to the inertial sensor, the EER of features extracted from the gyroscope (9.54%) is smaller than that extracted from the accelerometer (14.81%), this may be due to the fact that the rotation of the device can more stably represent the user-specific grip posture and the touching behavior, and the features extracted from the gyroscope are more discriminative and stable.

Spatial Alignment of Sensor Data. To evaluate the spatial alignment method, Fig 15(b) shows the EER when using the spatial and the temporal alignment method, i.e., 3.29% and 6.04% respectively. It indicates the efficiency



(a) Different training sizes (b) Differnet sampling rates (c) Impact of threshold ϵ_f

Fig. 16. Impact of parameters in our system

of our spatial alignment method. In addition, considering the time differences (i.e., duration difference) among gestures can affect the data alignment result, we group the gestures based on the difference between the duration of this gesture and the average duration of all gestures. As shown in Fig. 15(c), as the time difference increases, the EER of temporal alignment method increases, while that of our spatial alignment method keeps about 4%. It indicates that spatial alignment can effectively mitigate the influence of time difference in sensor data and keep the stability of touch gestures, resulting in the performance improvement of our system.

Gesture Segmentation. To evaluate the gesture segmentation method, Fig. 15(d) shows EERs of 3.61%, 7.94%, and 11.21% for TouchID when adopting our sub-gesture based model, DTW matching [51] and MHD matching [30], respectively. It indicates that our sub-gesture based method achieves better performance than DTW matching and MHD matching. This is because our model can highlight sub-gestures which contribute to authentication performance, and weaken sub-gestures which contain noise and other identity-independent information. The sub-gesture based model method outperforms the whole waveform matching methods in our system.

Feature Selection Based on Fisher Score. To evaluate the feature selection method, Fig. 15(e) shows EERs of 3.31%, 4.45% and 5.31% for the feature set selected based on Fisher Score, coefficient of variation (cv) [42] and feature set without feature selection. Firstly, it indicates that feature selection method can effectively choose features that are more effective at representing the user's touching characteristics. Secondly, the selected feature set based on Fisher Score achieves better EER performance than the selected feature set based on cv. This is because selected feature set based on cv just focuses on extracting stable features, while ignoring discriminative features that can improve the system performance.

7.5 Impact of Parameters

To evaluate the impact of parameters, including the training size, the sampling rate of inertial sensors, and the threshold for Fisher Score, we use the samples of 10th gesture in Fig. 13 from 20 users.

Effect of Training Set Size. In this experiment, we explore the impact of different training sample sizes on EER value. Fig. 16(a) shows the average EERs over all users, where the number of training samples is 3, 5, 10, 15, 20, and 25 respectively. It is obvious that the authentication performances increases as the training set size increases. Nevertheless, the average EER with 15 training samples is comparable to those with 20 and 25 training samples. Therefore, 15 training samples are enough for training, and the small-size samples can reduce the cost of data collection and classifier training.

Effect of Sampling Rate of Inertial Sensors. For most Android phones, the sampling rate of inertial sensors can be adjusted while the sampling rate of the touch sensor is usually fixed (e.g., 60 Hz). Therefore, we explore how the sampling rates of inertial sensors affect the authentication performance. In this experiment, the sampling rates of inertial sensors were set to 500 Hz, then we obtain new sensor data at different sampling rates by downsampling the raw sensor data. For example, to simulate a 100 Hz sampling rate, we choose every five data

series from the original data series. After getting the downsampled data from inertial sensors, we evaluate the performance of our system at sampling rates of 500 Hz, 100 Hz, 50 Hz, 16 Hz, and 5 Hz, respectively. Fig. 16(b) shows that the EER of 50 Hz is comparable to those of 100Hz and 500Hz, while the EERs for 16Hz and 5Hz are worse than 50 Hz. Besides, our measurements indicate that 99% sub-gestures last more than 20 ms, therefore we can capture enough information when the sampling rate is equal to or larger than 50Hz. As the sampling rate of 50 Hz is available on most commercial smartphones for inertial sensors, we can choose 50 Hz as the sampling rate of inertial sensors in TouchID, which can reduce the processing time and the energy overhead of user authentication.

Effect of Threshold for Fisher Score. The threshold of the Fisher Score will affect the performance of authentication. On the one hand, if the threshold is too large, we can only get very few features with the best stability and discriminability, and discard lots of effective features, which makes it difficult for us to fully depict the user's touch behavior. On the other hand, if the threshold is too small, some features with poor stability and discriminability may be selected, which leads to a reduction in the accuracy of authentication. Fig. 16(c) shows the average EERs with respect to different thresholds of Fisher Score. We observe that when the EER is minimum and the AUC is maximum, the corresponding threshold value of Fisher Score is 1.0, which is adopted in TouchID.

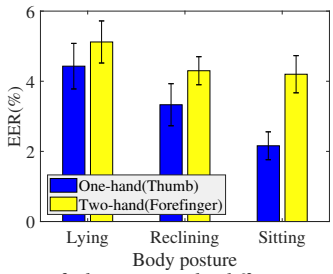
7.6 Authentication Performances under Complex Scenarios

In this subsection, we evaluate the performance of our system under complex scenarios. Specifically, we will test the performance of TouchID under different postures, on different screen sizes, with different hand sizes, and in a long time. Note that in the following experiments, we collect 40 samples from each user under each condition.

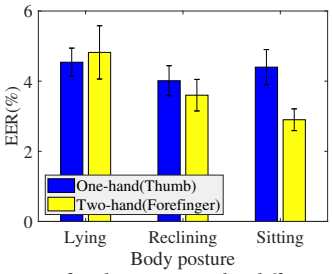
Single-posture Scenario. To evaluate the performance of our system under different postures, we choose four different gestures and six common postures, and invite 10 users to perform each gesture under each posture. The EER results of the 9th, 10th, 19th, and 26th gestures are shown in Fig. 17(a), Fig. 17(b), Fig. 17(c), and Fig. 17(d), respectively. Take the 9th gesture as an example, Fig. 17(a) indicates that the average EERs are 4.43%, 5.12%, 3.33%, 4.30%, 2.16%, and 4.20% under the 'Lying-one', 'Lying-two', 'Reclining-one', 'Reclining-two', 'Sitting-one', and 'Sitting-two', respectively. Our system achieves better performances under sitting and reclining postures than the lying posture while using two hands, this is mainly due to the unstable patterns of finger movements and device motions when the user performs touch gestures under the lying posture, especially when using one hand. Nevertheless, the EERs of the four gestures under different postures are all lower than 5.50%, indicating that our system can achieve a comparable EER performance under each single-posture scenario.

Multi-posture Scenario. In this case, we evaluate the performance of our system under multi-posture scenario based on the 9th gesture. We first compute EERs under six single postures as a benchmark, where the training and the testing data are collected under each single posture. Then we compute EERs under the multi-posture scenario, where the training and the testing data are collected under six different postures. Fig. 17(e) shows the EERs results tested on single-threshold kNN, the EERs under six single postures are about 5% while the EER under multi-posture scenario is larger than 30%. Fig. 17(f) shows the EERs results tested on our proposed multi-threshold kNN, the EERs under six single postures are less than 5%, and the EER under the multi-posture scenario is only about 5%, which is much smaller than the EER of 30% using the single-threshold kNN. Based on the multi-threshold kNN, TouchID can authenticate users under multi-posture scenarios without sacrificing the EER, thus ensuring usability in practice.

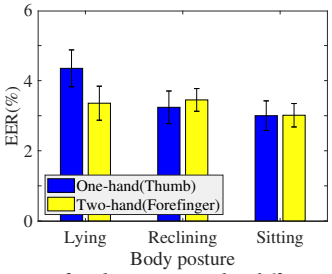
Different screen sizes. To evaluate how screen sizes affect the performance of TouchID, we introduce five smartphones with different screen sizes, i.e., Huawei nova with a 5.0-inch screen, Xiaomi Note3 with a 5.5-inch screen, Galaxy S9 with a 5.8-inch screen, Galaxy Note8 with a 6.3-inch screen, and Honor 8X Max with a 7.12-inch screen. This range from 5 inches to 7.12 inches can cover the screen sizes of more than 96% smartphones [28]. Then, we invite 10 users to perform the 10th gesture in Fig. 13 on each smartphone. As shown in Fig. 17(g), the



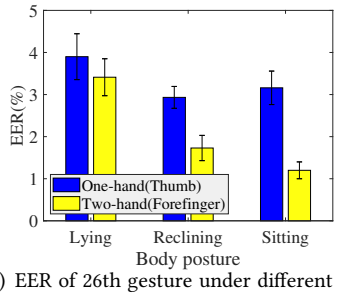
(a) EER of 9th gesture under different postures



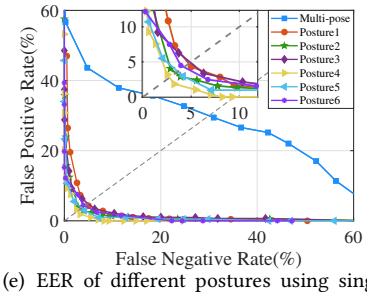
(b) EER of 10th gesture under different postures



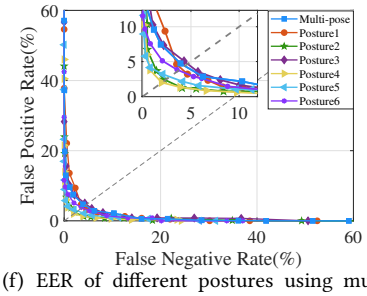
(c) EER of 19th gesture under different postures



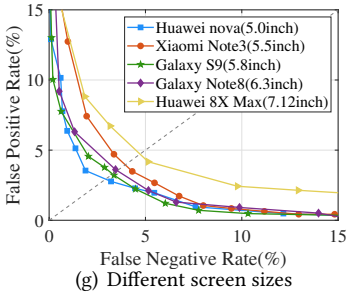
(d) EER of 26th gesture under different postures



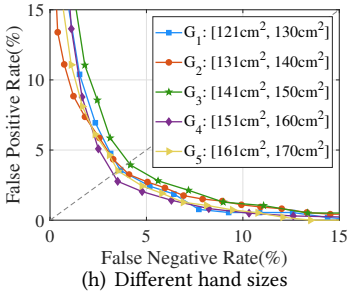
(e) EER of different postures using single-threshold kNN



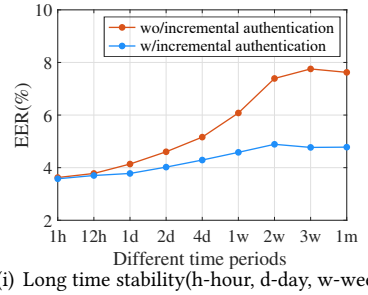
(f) EER of different postures using multi-threshold kNN



(g) Different screen sizes



(h) Different hand sizes



(i) Long time stability(h-hour, d-day, w-week, m-month)

Fig. 17. Authentication performance under complex scenarios

EER on the 5.0-inch screen, 5.5-inch screen, 5.8-inch screen, 6.3-inch screen, and 7.12-inch screen is 3.20%, 4.02%, 3.31%, 3.53%, and 4.64%, respectively. It indicates that the screen size has little effect on the user authentication performance of TouchID. This may be because people can adapt to the smartphones with common screen sizes and perform touch gestures on the screens well.

Different hand sizes. To evaluate how hand sizes affect the authentication performance of TouchID, we invite 20 users and each user performs the 10th gesture in Fig. 13. In regard to the hand size, it is represented with the product of the hand length and hand breadth. The hand length is measured from the tip of the longest finger to the crease under the palm, while the hand breadth is measured across the widest area where the fingers join the palm [22]. Considering the nonuniform distribution of hand sizes among the users, we deliberately select 20 users so that the users can be equally divided into five groups based on hand sizes. Specifically, $G_1 : (120cm^2, 130cm^2]$, $G_2 : (130cm^2, 140cm^2]$, $G_3 : (140cm^2, 150cm^2]$, $G_4 : (150cm^2, 160cm^2]$, and $G_5 : (160cm^2, 170cm^2]$ represent the

five groups. In the experiment, each user will be chosen as the legitimate user in turn, while all other users will be treated as attackers, and then we calculate the average EER of users in each group. As shown in Fig. 17(h), when the group is G_1, G_2, G_3, G_4, and G_5, the corresponding EER is 3.60%, 3.71%, 4.05%, 3.49% and 3.52%, respectively. It indicates that TouchID can efficiently reject the attackers with similar (i.e., users in the same group) or different (i.e., users in different groups) hand sizes. This implies that TouchID can authenticate users efficiently with different hand sizes.

Long-time stability. To evaluate the stability of TouchID over time, we invite 10 users to perform the 10th gesture in Fig. 13. Specifically, we first train the system using the samples collected at the initialization time. Then we use the initial trained classifier to test the samples of the same user collected after an hour, twelve hours, one day, two days, one week, two weeks, three weeks, and one month, respectively. As shown in Fig. 17(i), as time goes by, the user authentication performance of TouchID decreases, i.e., the EER increases a little. This is because of the slight variation of user behaviors in a long time. However, despite the performance decreases, the EER after one month is still less than 8%. In addition, when using the incremental authentication scheme described in Section 6.3, the EER almost keeps unchanged, i.e., the EER is low and only changes from 3.57% to 4.7% during a month. It indicates that TouchID can provide the long-time stability in terms of user authentication, and it can also adapt to the slight variation of gestures from the same user by incremental authentication.

7.7 Latency and Power Consumption

For each authentication process, TouchID achieves a slight latency of 4.20 *ms* for user authentication with 30 training samples. We measured the processing time of our system on Samsung Galaxy S9 with 2.8GHz quad-core processor. The sampling rate of the touch sensor and the inertial sensor were set to 60 Hz and 100 Hz respectively. For a touch gesture, our system takes an average of 1.41 *ms* to filter and synchronize the sensor data, 0.31 *ms* to spatially align the sensor data and segment the gesture, and 2.48 *ms* to extract features and authenticate a user. Therefore, our system can authentication users without noticeable latency.

By using BatteryHistorian [23] for power measurement, the average power consumption is 490.2 ± 32.1 mW in the authentication process of our system. To measure the power consumption, we first measure the power consumption of P_1 when only keeping the screen on, then we measure the average power consumption P_2 when running our system for user authentication. After that, we measure the power consumption with $P_2 - P_1$. We randomly choose 5 different gestures while each gesture was performed 50 times, and we calculated the average power consumption of all samples on Samsung Galaxy S9.

7.8 System Security Analysis

In this experiment, we consider five attacks that may threaten our authentication system. We selected 10 of total 41 volunteers as legitimate users, and the training process of these 10 legitimate users was recorded as videos. Then we recruited 10 different volunteers as attackers to conduct smudge attack, shoulder surfing attack, mimicking attack, and mimicking attack with incentive. Each attacker randomly attacks three gestures of each legitimate user, and each type of attack for each gesture is performed 10 times.

Gesture-aware attack: We assume that the attacker knows nothing but the graphic pattern to perform. We remove the data of 10 legitimate users in the data set, and use the remaining data as attackers' data. We use the evaluation result in this case as a benchmark for comparison, as shown in Table 2.

Smudge attack [47]: In this case, we show the gesture trajectory of the legitimate user on the touchscreen, then we let the attacker perform imitation attacks of each gesture 10 times based on the gesture trajectory. Table 2 shows that the EER of smudge attack only has a small increase compared with the baseline. This is because the attacker only obtains limited trajectory information while the sliding rhythm of finger is invisible. In addition, it is basically impossible to mimic the pattern of device motions only from the limited trajectory information.

Shoulder surfing attack [42]: In this case, we use videos to record the process of performing all gestures by legitimate users, then we let the attacker watch the video and perform imitation attacks of each gesture 10 times. In Table 2, the EER of shoulder surfing attack has only 1.73% increase compared with the baseline, this shows that it is difficult for attackers to imitate both explicit finger movements and implicit device motions at the same time.

Mimicking attack [34]: In this case, we not only provide the video from the legitimate user but also provide the handbook to the attacker. The handbook states the holding posture of smartphone, the finger touching the screen, the rhythm of speed, and the trajectory of whole gesture. Then, we let the attacker to perform imitation attacks of each gesture 10 times. As shown in Table 2, the EER of mimicking attack is 6.55%, which is larger than the previous three attacks. This is because the richer information provided to the attackers in the mimicking attack. Nevertheless, the EER of 6.65% is low. It indicates that it is difficult for attackers to have the same hand size, finger size, and user habits with the legitimate users.

Mimicking attack (with incentive) [34]: In this case, the attackers can obtain all the information given in mimicking attack. Besides, the attackers can also get real-time feedbacks. That is to say, they can get the real-time authentication result (i.e., rejection or acceptance) and obtain material rewards for successful attacks. As shown in Table 2, the authentication performance decreases a little when compared with mimicking attack, but is still a good performance, i.e., the EER is 6.73%. It further demonstrates the difficulty of mimicking other people's behaviors, which are generated by specific hands, fingers and user habits. In conclusion, these results show that our authentication system is resilient to the five different attacks.

Table 2. Average EERs for five attacks on our system

Type of Attack	Gesture-aware	Smudge	Shoulder surfing	Mimicking	Mimicking (with incentive)
EER result	3.19% (baseline)	4.04%	4.92%	6.55%	6.73%

7.9 Comparison with Previous Work

To compare TouchID with the state-of-the-art methods: Shahzad et al. [42], Shen et al. [44], Wang et al. [51], and Jain et al. [30] , we first analyze the difference between the existing work and TouchID from the aspects of sensor source, gesture type, feature selection method, and classifier. For sensor sources, all the existing work adopts the touch sensor [30, 42, 44, 51] to capture the on-screen gestures. However, to further capture the device motions caused by touch gestures, we also introduce the accelerometer and gyroscope. Thus we use the same sensors with Wang et. al [51]. For gesture types, all the existing work focuses on touch-related gestures, including the single-touch swiping gestures [30, 44], user-defined single-touch gestures [42] and multi-touch gestures [42, 51]. Usually, these customized gestures are rarely adopted for user authentication on COTS devices. In this paper, we focus on the single-touch gestures based on graphic patterns, which have been integrated in many COTS devices. However, without the relationship between a series of customized gestures and the relations between fingertips in multi-touch gestures, authenticating users in TouchID can be more challenging. For feature selection methods, statistical features [44] of the whole touch gesture are used in much work for authentication. Besides, the Fisher Score [44] and the coefficient of variation [42] are often adopted to select effective features. Different from these methods, we split the touch gestures by key nodes and select effective features both from each sub-gesture and the whole gesture based on Fisher Score, to represent both the stability and discriminability among user behaviors with features. For classifier, the waveform matching method [30, 51] and common classifiers like SVM [42], Random Forest [44], and KNN are often used. Different from these classifiers, we propose a multi-threshold kNN classifier, which supports incremental user authentication under new postures, in a long time, and so on.

In the experiments, to make a fair comparison, we use the original features and methods proposed in these papers, and adopt the same evaluation settings and datasets for each method. Specifically, we implement the methods in these four papers and perform the evaluation experiment based on our dataset. Table 3 shows the

average EERs over all gestures of each method. In Table 3, our method achieves the smallest EER among all methods, indicating that our method has a better authentication performance. This may be because that TouchID utilizes both touch sensor and inertial sensor to capture on-screen gestures and device's motions, adopts the spatial alignment method to reduce the intra-class difference among gestures, segments gestures into sub-gestures for better feature selection, introduces Fisher Scores to select effective features, and so on.

Table 3. Comparison with existing methods on our dataset

Method	Sensor Source∗	Gesture Type	Feature Selection	Classifier	EER
Our work	touch & acc & gyro	single-touch pattern gesture	Fisher Score & key node	KNN	**4.90%**
Shahzad et al. [42]	touch & acc	multi-touch & single-touch	coefficient of variation	SVM	9.67%
Shen et al. [44]	touch	single-touch swiping gesture	Fisher Score	Random Forest	13.09%
Wang et al. [51]	touch & acc & gyro	multi-touch gesture	-	DTW & Correlation	8.31%
Jain et al. [30]	touch & acc & ori	single-touch swiping gesture	-	MHD	11.54%

(∗touch-touch sensor, acc-acclerometer, gyro-gyroscope, ori-orientation)

8 DISCUSSION AND FUTURE WORK

In this paper, we propose a user authentication scheme on mobile devices via inertial-touch gesture analysis. Considering the easy-to-use unlock operation, low equal error rate and the support by a larger number of devices, TouchID has a good practicality in real-world scenarios. However, when considering the effect of application scenarios, advanced sensors, countless postures, and unexpected motions, TouchID still bears some limitations and we will try our best to deal with these issues in future work.

Touch gestures: TouchID focuses on the graphic pattern based touch gesture, which has the layout constraint. However, in some application scenarios, the touch gesture may have no constraints with graphic patterns. For example, some existing work has researched the swiping [30, 44] and free-form [45] touch gestures. For this kind of gestures, we may calculate the radius of curvature of touch gesture to detect the key points, which are like the nodes in graphic patterns, and then adopt the similar techniques in TouchID for user authentication. In addition, TouchID focuses on the single-touch gesture, i.e., using only one finger. However, when using multiple fingers to perform touch gestures, e.g., the 2-finger, 3-finger, and 4-finger swiping gestures in the existing work [42, 47], it is possible to get more information about the physiological features and user behaviors among multiple fingers. In future work, we will try to extend TouchID and make it support the single-touch gesture and multi-touch gesture based user authentication.

Sensor Data: In this paper, we use the touch sensor and inertial sensor for gesture analysis. The touch sensor can only provide the coarse-grained pressure data, i.e., 0 or 1. Nowadays, the touch sensor in some smartphones can provide more fine-grained pressure data [29], which can better describe the user behaviors on the screen. Besides, the fine-grained pressure data is unseen, thus can improve the difficult for attackers to snoop or imitate the legitimate user's touch behavior. In future, we will try to introduce the fine-grained touch sensor data to further improve the security and authentication performance of TouchID. In addition, due to the development of advanced sensors, the physiological features like fingerprints and faces are used for authentication. In the future, we will try to combine the physiological features ("what" the user has) and behavioral features ("how" the user performs) to provide both the static and the dynamic authentication, to further improve the authentication performance.

Arbitrary Postures: As described in the paper, the holding postures of smartphones affect the authentication performance. Besides, the arbitrary holding postures are countless. The cost of adding the training data under each posture for authentication is non-negligible. In future work, we will try to introduce adversarial learning to reduce the effect from postures and make TouchID work in different postures.

Interference Motions: In our system, we leverage the inertial sensor to depict the unique patterns of device motions, while the user is expected to keep the body relatively static when performing touch gestures. Since the inertial sensor is sensitive to body movements, our system might not reliably extract effective features related to the device's motion when the volunteer is running or walking. In the future, we will try to introduce the filter to remove the periodic movements or the motions with a greater range, to reduce the effect of body motions and make TouchID work well.

9 CONCLUSION

In this paper, we propose a single-touch gesture based user authentication scheme TouchID based on inertial-touch gesture analysis. When the user performs the touch gesture on graphic patterns, TouchID can align the sensor data in space domain, segment the touch gesture into sub-gestures, introduce Fisher Scores for feature selection, and use the multi-threshold kNN for user authentication. We implement TouchID on the Android powered smartphone and evaluate the performance of TouchID with a lot of experiments. The experiment results show that TouchID can authentication users efficiently with a low EER and outperforms the existing work.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China under Grant No. 2018AAA0102302; National Natural Science Foundation of China under Grant Nos. 61802169, 61872174, 61832008, 61906085, 61902175; JiangSu Natural Science Foundation under Grant Nos. BK20180325, BK20190293. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Yafeng Yin is the corresponding author.

REFERENCES

[1] Margit Antal and László Zsolt Szabó. 2016. Biometric authentication based on touchscreen swipe patterns. *Procedia Technology* 22 (2016), 862–869.

[2] Adam J Aviv, Katherine L Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. 2010. Smudge Attacks on Smartphone Touch Screens. *Woot* 10 (2010), 1–7.

[3] Bloggers. 2013. Common Lock Patterns of mobiles phones Screen Lock. http://mytrickytricks.blogspot.com/2013/07/commonlockpattern.html.

[4] Alipay by Ant Financial Inc. 2020. https://apps.apple.com/us/app/alipay-simplify-your-life/id333206289.

[5] Yaron Caspi and Michal Irani. 2002. Spatio-temporal alignment of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 11 (2002), 1409–1424.

[6] Tianfeng Chai and Roland R Draxler. 2014. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development* 7, 3 (2014), 1247–1250.

[7] Wenqiang Chen, Lin Chen, Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, and Kaishun Wu. 2019. Taprint: Secure Text Input for Commodity Smart Wristbands. In *The 25th Annual International Conference on Mobile Computing and Networking*. ACM, 1–16.

[8] Yimin Chen, Jingchao Sun, Rui Zhang, and Yanchao Zhang. 2015. Your song your way: Rhythm-based two-factor authentication for multi-touch mobile devices. In *2015 IEEE conference on computer communications (INFOCOM)*. IEEE, 2686–2694.

[9] Philip J Davis. 1975. *Interpolation and approximation*. Courier Corporation.

[10] Atul Dhingra, Amioy Kumar, Madasu Hanmandlu, and Bijaya Ketan Panigrahi. 2013. Biometric based personal authentication using eye movement tracking. In *International Conference on Swarm, Evolutionary, and Memetic Computing*. Springer, 248–256.

[11] Alaa El Masri, Harry Wechsler, Peter Likarish, Christopher Grayson, Calton Pu, Dalal Al-Arayed, and Brent ByungHoon Kang. 2015. Active authentication using scrolling behaviors. In *2015 6th International Conference on Information and Communication Systems (ICICS)*. IEEE, 257–262.

[12] Nesli Erdogmus and Sebastien Marcel. 2014. Spoofing face recognition with 3D masks. *IEEE transactions on information forensics and security* 9, 7 (2014), 1084–1097.

[13] Mohammed E Fathy, Vishal M Patel, and Rama Chellappa. 2015. Face-based active authentication on mobile devices. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1687–1691.

[14] Luca Fiorani, Mario Armenante, Roberta Capobianco, Nicola Spinelli, and Xuan Wang. 1998. Self-aligning lidar for the continuous monitoring of the atmosphere. *Applied optics* 37, 21 (1998), 4758–4764.

[15] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. 2012. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE transactions on information forensics and security* 8, 1 (2012), 136–148.

[16] Chiara Galdi, Michele Nappi, Daniel Riccio, and Harry Wechsler. 2016. Eye movement analysis for human authentication: a critical survey. *Pattern Recognition Letters* 84 (2016), 272–283.

[17] Nuria González-Prelcic, Roi Méndez-Rial, and Robert W Heath. 2016. Radar aided beam alignment in mmwave V2I communications supporting antenna diversity. In *2016 Information Theory and Applications Workshop (ITA)*. IEEE, 1–7.

[18] Quanquan Gu, Zhenhui Li, and Jiawei Han. 2011. Linear discriminant dimensionality reduction. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 549–564.

[19] Javier Guerra-Casanova, Carmen Sánchez-Ávila, Gonzalo Bailador, and Alberto de Santos Sierra. 2012. Authentication in mobile devices through hand gesture recognition. *International Journal of Information Security* 11, 2 (2012), 65–83.

[20] Guodong Guo, Lingyun Wen, and Shuicheng Yan. 2013. Face authentication with makeup changes. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 5 (2013), 814–825.

[21] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking access control and authentication for the home internet of things (IoT). In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 255–272.

[22] healthline Inc. 2019. What's the Average Hand Size for Men, Women, and Children? https://www.healthline.com/health/average-hand-size. (2019).

[23] Battery historian by Google Inc. 2016. https://github.com/google/battery-historian.

[24] Chenyu Huang, Huangxun Chen, Lin Yang, and Qian Zhang. 2018. BreathLive: Liveness detection for heart sound authentication with deep breathing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–25.

[25] Cisco Systems Inc. 2020. Fingerprint cloning: Myth or reality? https://blog.talosintelligence.com/2020/04/fingerprint-research.html. (2020).

[26] Google Inc. 2020. android.view.MotionEvent. https://developer.android.com/reference/android/view/MotionEvent. (2020).

[27] IDC Inc. 2020. Smartphone Challenges Continue in 2019. https://www.idc.com/getdoc.jsp?containerId=prUS45487719. (2020).

[28] Statista Inc. 2020. Smartphone unit shipments worldwide by screen size from 2018 to 2022. https://www.statista.com/statistics/684294/global-smartphone-shipments-by-screen-size/. (2020).

[29] WaveTouch Inc. 2020. Touchscreens with the 3rd dimension. https://www.wavetouch.com/technology/pressure-detection/. (2020).

[30] Ankita Jain and Vivek Kanhangad. 2015. Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures. *Pattern recognition letters* 68 (2015), 351–360.

[31] Andrew Jonathan Jakab. 2001. *Quality monitoring of GPS signals*. University of Calgary.

[32] Bing Jian and Baba C Vemuri. 2010. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence* 33, 8 (2010), 1633–1645.

[33] Sarah Martina Kolly, Roger Wattenhofer, and Samuel Welten. 2012. A personal touch: Recognizing users based on touch screen behavior. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*. 1–5.

[34] Xiaopeng Li, Fengyao Yan, Fei Zuo, Qiang Zeng, and Lannan Luo. 2019. Touch Well Before Use: Intuitive and Secure Authentication for IoT Devices. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–17.

[35] Jian Liu, Cong Shi, Yingying Chen, Hongbo Liu, and Marco Gruteser. 2019. Cardiocam: Leveraging camera on mobile devices to verify users while their heart is pumping. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 249–261.

[36] Li Lu, Jiadi Yu, Yingying Chen, Hongbo Liu, Yanmin Zhu, Linghe Kong, and Minglu Li. 2019. Lip reading-based user authentication through acoustic sensing on smartphones. *IEEE/ACM Transactions on Networking* 27, 1 (2019), 447–460.

[37] Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada, and Satoshi Hoshino. 2002. Impact of artificial" gummy" fingers on fingerprint systems. In *Optical Security and Counterfeit Deterrence Techniques IV*, Vol. 4677. International Society for Optics and Photonics, 275–289.

[38] Andriy Myronenko and Xubo Song. 2010. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence* 32, 12 (2010), 2262–2275.

[39] Nalini K Ratha, Ruud M Bolle, Vinayaka D Pandit, and Vaibhav Vaish. 2000. Robust fingerprint authentication using local structural similarity. In *Proceedings Fifth IEEE Workshop on Applications of Computer Vision*. IEEE, 29–34.

[40] Aditi Roy, Nasir Memon, and Arun Ross. 2017. Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems. *IEEE Transactions on Information Forensics and Security* 12, 9 (2017), 2013–2025.

[41] Nabilah Shabrina, Tsuyoshi Isshiki, and Hiroaki Kunieda. 2016. Fingerprint authentication on touch sensor using Phase-Only Correlation method. In *2016 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*. IEEE, 85–89.

[42] Muhammad Shahzad, Alex X Liu, and Arjmand Samuel. 2013. Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. 39–50.

[43] Chao Shen, Yuanxun Li, Yufei Chen, Xiaohong Guan, and Roy A Maxion. 2017. Performance analysis of multi-motion sensor behavior for active smartphone authentication. *IEEE Transactions on Information Forensics and Security* 13, 1 (2017), 48–62.

[44] Chao Shen, Yong Zhang, Xiaohong Guan, and Roy A Maxion. 2015. Performance analysis of touch-interaction behavior for active smartphone authentication. *IEEE Transactions on Information Forensics and Security* 11, 3 (2015), 498–513.
[45] Michael Sherman, Gradeigh Clark, Yulong Yang, Shridatt Sugrim, Arttu Modig, Janne Lindqvist, Antti Oulasvirta, and Teemu Roos. 2014. User-generated free-form gestures for authentication: Security and memorability. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. 176–189.
[46] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran S Balagani. 2015. HMOG: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security* 11, 5 (2015), 877–892.
[47] Yunpeng Song, Zhongmin Cai, and Zhi-Li Zhang. 2017. Multi-touch authentication using hand geometry and behavioral information. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 357–372.
[48] Jiayao Tan, Xiaoliang Wang, Cam-Tu Nguyen, and Yu Shi. 2018. SilentKey: A new authentication framework through ultrasonic-based lip reading. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–18.
[49] Furkan Tari, Ant Ozok, and Stephen H Holden. 2006. A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords. In *Proceedings of the second symposium on Usable privacy and security*. ACM, 56–66.
[50] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html
[51] He Wang, Dimitrios Lymberopoulos, and Jie Liu. 2015. Sensor-based user authentication. In *European Conference on Wireless Sensor Networks*. Springer, 168–185.
[52] Lei Wang, Kang Huang, Ke Sun, Wei Wang, Chen Tian, Lei Xie, and Qing Gu. 2018. Unlock with your heart: Heartbeat-based authentication on commercial mobile phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–22.
[53] Sa Wang, Cheng-Lin Liu, and Lian Zheng. 2007. Feature selection by combining Fisher criterion and principal feature analysis. In *2007 International Conference on Machine Learning and Cybernetics*, Vol. 2. IEEE, 1149–1154.
[54] Susan Wiedenbeck, Jim Waters, Leonardo Sobrado, and Jean-Camille Birget. 2006. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 177–184.
[55] Hui Xu, Yangfan Zhou, and Michael R Lyu. 2014. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*. 187–198.
[56] Zhi Xu, Kun Bai, and Sencun Zhu. 2012. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. 113–124.
[57] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zheng Wang. 2017. Cracking Android pattern lock in five attempts. In *Proceedings of the 2017 Network and Distributed System Security Symposium 2017 (NDSS 17)*. Internet Society.
[58] Safe Vault Hide your photos by FT Apps Inc. 2020. https://apps.apple.com/us/app/safe-vault-hide-your-photos/id1200107476.
[59] Heng Zhang, Vishal M Patel, Mohammed Fathy, and Rama Chellappa. 2015. Touch gesture-based active user authentication using dictionaries. In *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 207–214.
[60] Xi Zhao, Tao Feng, and Weidong Shi. 2013. Continuous mobile authentication using a novel graphic touch gesture feature. In *2013 IEEE sixth international conference on biometrics: theory, applications and systems (BTAS)*. IEEE, 1–6.
[61] Nan Zheng, Kun Bai, Hai Huang, and Haining Wang. 2014. You are how you touch: User verification on smartphones via tapping behaviors. In *2014 IEEE 22nd International Conference on Network Protocols*. IEEE, 221–232.
[62] Man Zhou, Qian Wang, Jingxiao Yang, Qi Li, Feng Xiao, Zhibo Wang, and Xiaofen Chen. 2018. Patternlistener: Cracking android pattern lock using acoustic signals. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1775–1787.