Research Article

Just Shake Them Together: Imitation-Resistant Secure Pairing of Smart Devices via Shaking

Congcong Shi ^{1,2}, Lei Xie ¹, Chuyu Wang ¹, Peicheng Yang ¹, Yubo Song ³,
and Sanglu Lu ¹

¹State Key Laboratory for Novel Software Technology, Nanjing University, China

²State Grid Key Laboratory of Information and Network Security, Global Energy Interconnection Research Institute Co. Ltd, China

³School of Cyber Science and Engineering, Southeast University, China

Correspondence should be addressed to Lei Xie; lxie@nju.edu.cn

Received 10 December 2020; Revised 2 February 2021; Accepted 23 March 2021; Published 5 April 2021

Academic Editor: Zhipeng Cai

Copyright © 2021 Congcong Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In traditional device-to-device (D2D) communication based on wireless channel, identity authentication and spontaneous secure connections between smart devices are essential requirements. In this paper, we propose an imitation-resistant secure pairing framework including authentication and key generation for smart devices, by shaking these devices together. Based on the data collected by multiple sensors of smart devices, these devices can authenticate each other and generate a unique and consistent symmetric key only when they are shaken together. We have conducted comprehensive experimental study on shaking various devices. Based on this study, we have listed several novel observations and extracted important clues for key generation. We propose a series of innovative technologies to generate highly unique and completely randomized symmetric keys among these devices, and the generation process is robust to noise and protects privacy. Our experimental results show that our system can accurately and efficiently generate keys and authenticate each other.

1. Introduction

More and more smart mobile devices have been used in our daily life. These devices usually communicate with each other via Bluetooth or WiFi Direct. However, data transmitted through these methods is not completely private since the attacker can intercept information by eavesdropping in the wireless channel. This leads to insecure transmission of private data such as an electronic business card. Therefore, it is essential for users to authenticate each other and establish spontaneous secure connections between devices. This is to achieve mutual authentication and key generation.

Existing solutions mainly utilize the password or patterns to authenticate each other and establish secure connections. However, it is not suitable to apply these solutions on mobile devices. These solutions have three major weaknesses. First, for some small devices like smart bracelets, they have no touch screens for users to enter the password or patterns. Sec-

ond, the password/patterns are vulnerable to shoulder surfing attacks because they are easily peeped in public places. Third, users prefer to choose shorter passwords or simpler patterns if they have to input passwords or patterns frequently; this leads to a higher risk of information leakage.

Recently, some emerging work tries to make use of the embedded sensors such as touch screens [1–5], accelerometer [6–15], gyroscope [16–19], and wireless signals [20–26] to capture user's behavior and realize mutual authentication and key generation. Specifically, the accelerometer is usually used to authenticate devices and generate keys by shaking. In such scenarios, the user first needs to put together the devices to be authenticated and then shake them simultaneously. The accelerometer can collect the movement features of these devices during shaking process. The detailed profile and movement pattern of each device can be extracted from the sensor data. The devices can detect if they are shaken together by comparing these information. These

devices will authenticate each other and generate the unique and consistent key only when they are shaken together. The basic idea is that users tend to produce very random but similar trajectories for multiple devices during shaking process. This behavior contains enough features to create a unique profile for these devices, and it is difficult for attackers to mimic this behavior.

However, these existing solutions have the following limitations: first, the key generation is inefficient since sensor data is temporally and spatially correlated. The user has to shake devices long enough to generate enough bits. Second, they envision the ideal application scenario, where the devices are all homogeneous. However, the devices we use to pair are not always the same. These devices may differ in size, measurement units, and sampling frequency. Third, the sensors they use can only make coarse-grained perceptions of user's behavior. This leads to weak defenses against shoulder surfing attacks since it is easy to be imitated.

In this paper, we propose iShake, an imitation-resistant secure pairing framework including authentication and key generation for smart devices, by shaking these devices together as shown in Figure 1. iShake solves the limitations of previous work. First, we eliminate spatiotemporal correlation by selective sampling of sensor data. Moreover, we utilize hash-based message authentication code (HMAC) to generate random bits. Second, we investigate the key generation in heterogeneous devices. We find the "device displacement" which is common in shaking heterogeneous devices. We effectively solve this problem through adaptive calibration and quantization. Third, we capture the user's behavior in a much finer granularity by utilizing both the accelerometer and gyroscope. Therefore, we can perceive the micro-movement of user's behavior. We adaptively sample sensor data from different dimensions to generate more differentiated bits according to the different sensitivities of different dimensions.

2. Related Work

Nowadays, many researchers focus on how to effectively conduct pairing or authentication in smart devices, by using equipped sensors like touch screens [1–5], accelerometer [6–14], gyroscope [16–19], and wireless signals [20–26]. The idea of associating two or more devices by shaking them together was first put forward in "Smart-Its Friends" [27]. It uses shared "context proximity" of dedicated devices to authenticate each other, which is created by explicit user-controlled action. They mainly leverage the collected sensor data to carry out pairing or simple authentication in smart devices, without further considering establishing secure connections. Besides, they all try to exchange the raw sensor data or explicit feature information to each other, which cannot guarantee confidentiality and lead to heavy transmission overhead.

Recent work considers to further establish spontaneous secure connections among the smart devices. Mayrhofer proposes a protocol for generating secret shared keys from similar sensor data streams [7]. Rene and Gellersen leverage traditional asymmetric encryption and the feature of raw

sensor data to authenticate dedicated devices and create shared keys for secure channel between shaking devices [8, 9]. However, these methods mostly leverage asymmetric encryptions to encrypt raw sensor data, which brings heavy computational cost and cannot be effectively implemented in current mobile devices like the smart bracelets. Xu et al. [28] propose a novel secret key generation protocol, Gait-Key, which can generate the same cryptographic key based on the unique gait pattern for two legitimate devices of the user. Moreover, Revadigar et al. [29] further leverage the accelerators of multiple devices attached on the human body and generate the common secret keys based on the unique walking style of the user. But these works focus on the gait pattern for key generation between multiple devices of the same user, instead of the devices of different users. The most closely related works are [30, 31]. In [30], Bichler et al. present a novel approach to establish a secure connection between two devices by shaking them together. Instead of distributing or exchanging a key, the devices independently generate a key from the measured acceleration data by appropriate signal processing methods. In [31], Bichler et al. further propose an algorithm to independently synchronize the shaken devices so that reliable key generation becomes possible. However, their method requires an additional training phase to extract the detailed profile and pattern of user behaviors on the specified devices, which severely limits its application for different devices and different users. Moreover, some recent works try to build a secure connection between two devices (e.g., smartwrist or smartwatch) based on the handshaking gestures [13, 32]. In this paper, we focus on shaking two smartphones together to generate the cryptographic keys, which can generate more stable and distinguish keys compared with the handshaking.

3. Understand Shaking via Empirical Study

To understand how to conduct efficient mutual authentication and key generation, while considering the shaking gestures, shaking frequency, and types of sensors, we conduct comprehensive experiments in real environment, illustrate several novel observations, and extract some important clues. In the following experiments, we sample the sensor data on Samsung Galaxy Nexus i9250 and use the accelerometer and gyroscope to continuously collect the sensor data during the shaking process. Both the accelerometer and gyroscope can collect the trace of user behavior in three dimensions (x -, y -, and z -axes). For brevity's sake, we, respectively, use A_x , A_y , A_z and G_x , G_y , G_z to denote the accelerometer data and gyroscope data in various dimensions. The sensor data from accelerometer and gyroscope are, respectively, measured in the units of meter per second squared and radians per second. We totally record 100 sets of shaking traces at 100 Hz, and each set consists of 2048 data points.

Without loss of generality, when sampling sensor data, we shake devices using the following patterns of shaking gestures: (1) *back and forth*: the user shakes the devices in a back-and-forth approach with a large movement range of arm; the linear acceleration should be fairly large; (2) *roll*: the user shakes the devices with some micromovement of

FIGURE 1: Application scenario of iShake.

the wrist; the angular acceleration should be fairly large; (3) *cocktail*: it is a combination of back and forth and roll, so both the linear acceleration and the angular acceleration should be fairly large.

3.1. Sensor Data in Time Domain and Frequency Domain.

The sensor data sampled when shaking together have very close patterns, while sensor data sampled when shaking independently always have difference in details. Specifically, the sensor data in the time domain can effectively verify the difference of user behavior during shaking, while the sensor data in the frequency domain can reveal some statistic regularities like the activity degrees of user behavior. We obtain the sensor data by trying three different patterns of shaking gestures (back and forth, roll, and cocktail). We, respectively, test the case of shaking together (traces A and B) and shaking independently (traces A and C) and try to make the shaking gesture as random as possible. We find that for all shaking patterns, the sensor data sampled when shaking together have very close patterns, while sensor data sampled when

shaking independently always have difference in details. This just provides us a chance to effectively differentiate multiple devices that are not shaking together. Figure 2(a) shows an example of detailed results from the accelerometer and gyroscope in the x -, y -, and z -axes in the time domain for cocktail. Figures 2(b)–2(d) further show the sensor data in the frequency domain, which is obtained from fast Fourier transformation (FFT) over the raw sensor data in the time domain. The sensor data in the frequency domain show some regularities in a statistical approach. Specifically, if the sensor data contain more components in higher frequency, it means that the user has more detailed activities during the shaking process. We find that the shaking frequency mainly lies in between 0 Hz and 10 Hz, and the sensor data generated from the roll and cocktail pattern have more signals in higher frequency than back and forth; this implies that these shaking patterns can generate more details during the shaking process, which would help generate more number of bits for the key generation. Obviously, cocktail is the most efficient pattern in generating randomized bits, since it has the most details in degree of activity.

FIGURE 2: Continued.

FIGURE 2: Analysis on the sensor data in various domains.

3.2. Similarity in Time Domain and Frequency Domain.

When we use a window to sample the raw sensor data from devices shaking together, we find that as the size of window increases, the similarity of the traces gradually decreases. It is essential to set an appropriate granularity for the window to avoid too many mismatched bits. For two traces generated by shaking together, we, respectively, divide the series of sensor data into a number of blocks with fixed size in regard to each dimension. Figures 2(e) and 2(f), respectively, show the similarity in all dimensions for various block sizes in regard to the time domain and frequency domain. As we increase the block size from 64 bits to 512 bits, we find that the average similarity in all dimensions gradually decreases. The reason is as follows: as the amount of raw sensor data increases with the block size, the amount of mismatched data between the two pairs increases; hence, the similarity of the two pairs of sensor data sequence decreases. This implies that, if we need to verify the consistency of the generated bits in key generation, we need to set an appropriate granularity for the window to avoid too many mismatched bits. Note that the similarity in the time domain is always lower than the frequency domain; this means that the sensor data in the time domain can more effectively help verify the difference of user behavior than frequency domain.

3.3. Sensitivity in Various Sensors and Dimensions. Sensors in different dimensions have different sensitivities to capture the user behavior during the shaking process; the gyroscope is more sensitive than the accelerometer in all three dimensions for depicting the user behavior. According to the results in Figures 2(b)–2(d), it is noted that the sensors have different values for different dimensions in the frequency domain; this means that the users have different degrees of activities in different dimensions. For example, in Figure 2(d), the accelerometer data in the y -axis is apparently smaller than the accelerometer data in the x -axis and z -axis. This is understandable since any kind of shaking pattern may have intense activities in some dimensions and weak activities in the other dimensions. Therefore, the sensor data in different dimensions have different sensitivities to capture the user behavior during the shaking process, and the sensor data in the frequency domain can rightly reveal these different sensitivities. Moreover, we find that the gyroscope is usually more sensitive than the accelerometer in all three dimensions; this happens in all three kinds of shaking patterns. We believe this is because the micromovement of the wrist is usually larger than the movement of the arm during the shaking process.

3.4. The Correlation in Space and Time Domain. The sensor data have correlations in both the space and time domains, especially when the devices are shaking in a regular approach. In order to study the correlation in the space and time domain, we shake the devices with the cocktail pattern as randomly as possible. In Figures 2(g) and 2(h), we, respectively, conduct correlation analysis of sensor data in both the space and time domains. We use the Pearson product-moment correlation coefficient to calculate the correlation: $\rho_{x,y} = (E[(x - \mu_x)(y - \mu_y)]) / (\sigma_x \sigma_y)$, where μ_x, μ_y are, respectively, the mean of x and y ; σ_x, σ_y are, respectively, the standard deviation of x and y ; and E is the expectation.

Figure 2(g) shows the correlation in the space domain; we compare the correlations of sensor data from the accelerometer and gyroscope in all three dimensions. We find that there indeed exist some correlations among these sensor data. For example, the correlation is close to a value of 0.4 for several dimensions in both accelerometer and gyroscope. The reason is that, when the user is shaking, the sensors like the accelerometer and gyroscope may capture the same activity simultaneously from all dimensions. Figure 2(h) shows the correlation in the time domain; we conduct crosscorrelation in the same trace for different dimensions. We find that, even when we shake the devices in a rather random approach, there still exist some correlations for the traces in the time domain. Therefore, in order to ensure the randomness in key generation, it is essential to remove the redundancy in both the space and time domains. Moreover, we find that as the sensor data are having more components in higher frequency, the value of autocorrelation is getting smaller. The reason is that the proportion of components in high frequency is positively relevant to the complexity of the raw sensor data; when the raw sensor data is becoming more complex, the value of autocorrelation is getting smaller. Therefore, we can sufficiently leverage the sensor data with more details in high frequency to improve the randomness.

3.5. Shaking with Heterogeneous Devices. While shaking heterogeneous devices, it is common to see the device displacement; i.e., the devices may have relative displacement during the shaking process due to the variation in sizes; this may cause these devices to be out of sync and have biases in various dimensions. In order to study the moving patterns while shaking with heterogeneous devices, we shake two heterogeneous devices (device A: Samsung Galaxy Nexus i9250, 4.65 inches; device B: MI 2S, 4.3 inches) together with the cocktail pattern. Figure 2(i) shows the raw sensor data in the time domain. Different from the previous observations, we find that the traces may have some clear differences in amplitude with respect to some dimensions, especially for the sensor data from the accelerometer. The traces in the gyroscope only have some slight difference in amplitude as compared to the accelerometer. The reason is as follows: while shaking heterogeneous devices, it is common to see that the devices may have relative displacement due to the variation in sizes. Hence, there might be some differences in regard to the sensor data in all dimensions. As the displacement issue mainly impacts the parameters in linear

acceleration, so it mainly leads to disturbance in the sensor data from the accelerometer. Nevertheless, as these devices are shaking in the same trajectory, the general outline of the moving trace remains unchanged; there still exists much similarity with respect to the sensor data in variation trend.

4. iShake Framework

4.1. Design Goal and Challenges. The key goal is to conduct mutual authentication and generate symmetric key over insecure channel via shaking smart devices. Each device can independently generate a common key by exchanging a limited number of feature data instead of the whole series of raw sensor data, so that each device can establish a privacy-preserving channel with this common key between each other. According to the experimental study, we find that there are many disturbing factors that hinder devices to generate a unique and consistent key. We still need to effectively tackle the following challenging problems: (1) *generate distinctive bits*: in order to be imitation-resistant, the device should generate more distinctive bits to differentiate the legal users and the attacker. The generated key should be highly distinctive by exploring the characteristic details from the user's moving gesture. (2) *Robust to the noise*: due to the natural deviations and the device displacement in real applications, we cannot guarantee that any two traces have identical trajectories, even if the devices are shaking together. There are always some differences in details of the traces. The common features should be effectively extracted to generate a consistent key. (3) *Reduce the correlations*: there exist some correlations for the raw sensor data in both the time and space domains; the generated key should be made fully randomized to increase entropy by sufficiently reducing the relevant correlations and redundancies.

4.2. Threat Model. We assume attackers cannot hack into the target mobile devices to obtain the raw sensor data for user behaviors during the shaking process. Nevertheless, we assume attackers have the following three capabilities. First, attackers can launch eavesdropping attack by intercepting the transmitted messages in the open environment. Second, attackers can launch man-in-the-middle attack between two devices in interaction. Third, attackers can conduct shoulder surfing by privately imitating the legal user's behaviors in handshaking.

4.3. Mutual Authentication and Key Generation Framework. Figure 3 shows the diagram of the mutual authentication and key generation framework. The framework mainly includes the following steps: *data calibration*, *quantization*, and *key extraction*. The user first shakes the devices together in an arbitrary approach. Once detecting the event of shaking, each device starts to sample the user's behavior via the accelerometer and gyroscope. In the step of *data calibration*, the devices adaptively conduct *trace synchronization* and *data interpolation* on the sampled raw sensor data. In this way, the sensor data can be effectively synchronized and smoothed to mitigate the impact of out-of-sync and device displacement. In the step of *quantization*, each device independently generates

FIGURE 3: The diagram of mutual authentication and key generation.

bit series from the calibrated sensor data, according to the sensitivity of user behavior in different dimensions. In the step of *key extraction*, the devices exchange a limited number of encrypted messages among each other, so as to verify the consistency for mutual authentication and selectively use the consistent bits for key generation. In this way, these devices can generate a unique and consistent key via shaking.

4.4. Data Calibration

4.4.1. Trace Synchronization. In order to generate a consistent key among the shaking devices, it is essential to first synchronize the locally generated raw sensor data traces among each other, so as to make it possible to generate identical bit series for candidate key usage.

Conventionally, the synchronization is realized by broadcasting a *beacon* message from a specified device to the other devices. However, due to the existence of the attacker, it is impossible for a device to differentiate between the legal device and an illegal device, since the legal devices do not share a secret key in advance. An illegal device can thus send multiple *beacon* messages to interfere the synchronization among those legal devices. Therefore, we can only rely on the inherent *beacons* inside the raw sensor data traces to devise efficient synchronization method for shaking devices.

Figure 4 shows an example of trace synchronization. Note that before the user shakes the devices, the sensor data in all dimensions remains very close to 0. Once the user starts to shake the devices, the sensor data in all dimensions start to increase almost simultaneously. This provides us an opportunity to use it as an *inherent beacon* to locate an identical start point for synchronization. However, since we learn from the experimental study that the sensor data in different dimensions have different sensitivities to the user behavior, we need to comprehensively investigate the sensor data in all dimensions. In Algorithm 1, we propose an efficient solution for trace synchronization. In this algorithm, for each dimension, we, respectively, use a sliding window to simultaneously evaluate the sensor data; once we detect that the average value inside the window is beyond a certain threshold, we record the start point of the window. Realizing the different sensitivities in different dimensions, we use the average value of the first k points as the start point of the sensor data sequence. In this way, the traces from different devices can be effectively synchronized.

In order to conduct accurate trace synchronization, it is important to set an appropriate window size for the sliding window. If the window size is too large, then the sliding window cannot be very sensitive to the sudden increase of the sensor data; if the window size is too small, then the sliding window cannot tolerate any unpredictable turbulence or noise from the outside. According to the experimental study, we note that the user's shaking frequency is usually below 10 Hz; we use f_{sk} to denote the average frequency, e.g., $f_{sk} = 5$ Hz. We then use f_{sp} to denote the sampling frequency, e.g., $f_{sp} = 100$ Hz; thus, an appropriate value for window size should be $w = f_{sp}/f_{sk} = 20$. This implies that the window can effectively capture a complete action for average cases.

4.4.2. Data Interpolation. After the *trace synchronization*, the generated raw sensor data can be effectively synchronized. However, according to the experimental study, it is noted that due to the natural deviations and device displacement, any two traces are not necessary to have identical trajectories, even if the devices are shaking together. There are always some differences in details of the traces. Therefore, in order to make it robust to the noises, it is essential to conduct *data interpolation* to retain the broad outline while dropping away those details.

A key challenging problem is how to effectively differentiate between the outline and the detail. Since the user may shake the devices in either a rapid or slow approach, the major components of the raw sensor data may either in high frequency or in low frequency. Hence, we need to conduct data interpolation by efficiently differentiating between the outline and the detail based on the user's actual behavior.

As the user's shaking behavior has some locality in the time domain, e.g., the user's shaking frequency may not change too much within a limited time interval, we first break the whole data sequence S into multiple subsequences D ; then, we apply data interpolation to each subsequence D . For each subsequence D , we maintain a sliding window with a window size w to smooth the sensor data. The window size w can be adaptively adjusted to fit the actual situation of the trace. The process of data interpolation is shown in Algorithm 2. While using the sliding window to smooth the sensor data, we adaptively adjust the window size w by iteratively evaluating the crosscorrelation coefficient θ against a certain threshold t , e.g., 90%. Once the

FIGURE 4: Detect an identical start point for synchronization.

crosscorrelation coefficient is approaching the threshold, the smoothing process is terminated.

Figure 5 shows an example of the data interpolation. For two traces A and B obtained by shaking together, it can be found that before interpolation, the two traces have a number of differences in detail, which hinders them from generating identical bit series. We thus set the threshold $t = 0.9$ and adaptively adjust the window according to the trace; finally, it converges to a window size of 16. We find that, after the interpolation, the different details can be effectively smoothed and the major outline can be better retained as compared to the case with a window size of 32.

4.5. Data Filtering. Data filtering is a technique for improving the quality of motion sensor data. This technique selects the appropriate frequency band for fine-grained data filtering to reduce noise in the motion sensor data. This method can be applied in the training phase to determine the most efficient filtering method for noise reduction.

The proposed data filtering technique selects bands of different lengths rather than a fixed low-pass or high-pass band. This technique divides the full frequency band into several subbands whose granularity is determined by cutting the entire frequency band and reconnecting according to certain rules. These subband samples are used to filter the raw sensor data and to screen out the best performing subbands.

The most commonly used band division method is the $1/n$ Octave method, which divides the entire frequency band into two halves, then divides the low-frequency half into two halves and iterates according to this law. Finally, the method divides each subband equally into n new subbands. The $1/n$ Octave method is characterized by dividing the low-frequency band into finer-grained subbands that facilitate the processing of low-frequency dominant raw data. The $1/2$ Octave method is used to divide the entire frequency band into 16 subbands. Since the first three low-frequency subbands are smaller in size, they were combined into one subband.

After obtaining the 14 subbands divided by the Octave method, the 14 subbands are spliced and cascaded into differ-

ent granularities. Finally, a total of 105 frequency band samples of different lengths are obtained by this method. Then, these samples are applied to an infinite impulse response filter and evaluate the performance of these subbands to filter the motion sensor data.

4.6. Quantization. After the data calibration, we need to quantize the calibrated sensor data sequence into bit series for key generation. In order to boost the bit generation rate, we consider to quantize the amplitude of the sensor data across different sensors and dimensions. According to the experimental study, it is noted that different sensors have different sensitivities to the user's behavior in different dimensions. Therefore, in order to generate more distinctive bits based on user's profile, it is essential to generate bits from those dimensions which is more sensitive than the others.

Then, it raises a new challenge for designing the quantization algorithm: how to effectively evaluate the sensitivity to user's behavior for various dimensions? As a matter of fact, according to the experimental study, the sensor data in the frequency domain can reveal the activity degree of user behavior in different dimensions. Therefore, we can conduct *Fourier analysis* over the calibrated data to obtain the activity distribution in frequency.

Algorithm 3 shows the details of the quantization. When we get the calibrated sensor data from different dimensions, we first conduct fast Fourier transformation (FFT) over them for each dimension i . Then, we compute the top- k principal components in the frequency domain and get the highest frequency of these components as f_i . This implies that the user's behavior is most active within the band $[0, f_i]$. According to the Shannon theorem, we use the sampling frequency $2f_i$ to quantize the data sequence. In this way, the sensor data in those sensitive dimensions can effectively assist in generating more distinctive bits based on the user's behavior. We then normalize the sampled data such that the gap between two traces by shaking together can be effectively reduced; we then apply the level crossing method to generate the bit series according to the specified quantization levels.

After that, for each dimension i , we, respectively, break the generated bit series S_i into blocks with variable size according to the specified time line; the exact size right depends on the number of generated bits for each dimension. We then merge these blocks from multiple dimensions to bit series S in time order. In this way, we can generate a candidate bit series for the following key extraction. Figure 6 shows an example for quantization from different dimensions based on different sensitivities.

4.7. Mutual Authentication and Key Extraction. After the step of quantization, we can obtain a bit series according to the calibrated sensor data. However, these bit series cannot be directly used as the final key, because of the following reasons: (1) nonrandomness: due to the correlations in the space and time domain, there exist redundancy and correlation for a substantial part of the bit series. This causes that the generated key does not have good statistical randomness and high entropy. (2) Inconsistency: due to the issues like device displacement and noises, there still exist mismatched bits in

Require: 1) The raw sensor data sequence: D . 2) The threshold for accelerometer and gyroscope: α_a and α_g .

Ensure: The start point of the sensor data sequence: P .

- 1: Initialize a sliding window W_i for each dimension i , set the window size to $w = f_{sp}/f_{sk}$. Initialize the threshold t_i for each dimension i according to the sensor type.
- 2: **for** each dimension i **do**
- 3: Forward the sliding window W_i along the raw sensor data sequence D , calculate the average value inside the window W_i as v_i .
- 4: **if** $v_i > t_i$ **then**
- 5: Set the start point of the window as P_i . Stop forwarding the window W_i .
- 6: Find the first k points among the point set $\{P_i\}$ for all dimensions. Set the average value of the k points as P .

ALGORITHM 1: Trace synchronization.

Require: 1) The synchronized raw sensor data sequence: D . 2) The threshold of the cross correlation coefficient: t .

Ensure: The interpolated sensor data sequence: D_f .

- 1: Set the window size $w = 1$. Set the index of the sliding window $i = 1$. Set the cross correlation coefficient $\theta = 1$.
- 2: **while** $\theta > t$ **do**
- 3: Set the window size $w = w \cdot 2$.
- 4: **for** $i \leq \text{the length of } D$ **do**
- 5: Smooth the value D_i to $D_{f,i}$ with the average value $\bar{D}$ in the sliding window.
- 6: Forward the sliding window, set $i = i + 1$.
- 7: Conduct cross correlation between D and D_f , get the coefficient θ .
- 8: Set the window size $w = w/2$. Smooth the sequence D to D_f with the sliding window.

ALGORITHM 2: Data interpolation.

FIGURE 5: Interpolate the raw sensor data sequence.

the bit series, although we already take measures to reduce the possibility of mismatched bits in the previous steps. Therefore, we need an additional step to conduct mutual authentication and generate randomized and consistent keys.

Algorithm 4 shows the detailed design of mutual authentication and key extraction. In order to verify the consistency while preserving the privacy, it is essential for the peer devices to exchange a kind of signature for the generated bits to each other. Here, we leverage the hash-based message

Require: 1) The calibrated sensor data sequence with multiple dimension A_x, A_y, A_z and G_x, G_y, G_z ; D_f ; 2) The length of sensor data sequence: l .

Ensure: The bit series: S ;

1: **for** each dimension ido

2: Conduct *Fast Fourier Transformation* over $D_{f,i}$ for dimension i . Compute the top- k principal components in the frequency domain, obtain the highest frequency of these components as f_i .

3: Sample the sensor data $D_{f,i}$ for every $l/2f_i$ points, use a sliding window W_i to get the maximum value and the minimum value, normalize the sampled data into the interval $[-1, +1]$.

4: Use the level crossing method to quantize the data sequence $D_{f,i}$, obtain the generated bit series as S_i .

5: For each dimension, break the generated bit series S_i into blocks with variable size according to the specified time line. Merge these blocks from multiple dimensions to bit series S in time order.

ALGORITHM 3: Quantization.

FIGURE 6: Quantization from different dimensions based on different sensitivities.

Require: 1) The bit series: S , its length is l . 2) The number of bits required in key generation: k . 3) The threshold for mutual authentication: t .

Ensure: The extracted key: K .

1: Initialize a sliding window W of size w along the bit series S . Set the index $i = 1$.

2: **while** $i \leq l$ **do**

3: Obtain the bit string inside window W as M_i , generate a random number r_i , and call $\text{HMAC}(r_i, M_i)$ to obtain a bit string B_i .

4: Forward the window along the bit series S with 1 bit.

5: Send the series $X = \{r_i || B_i\} (1 \leq i \leq l)$ with length l_X to the peer device via broadcasting.

6: Receive the series $Y = \{r'_i || B'_i\} (1 \leq i \leq l)$ with length l_Y from the peer device.

7: Compute the edit distance d between X and Y , compute $\alpha = d / \max \{l_X, l_Y\}$, find the matching components of B_i and B'_i from X and Y accordingly.

8: **if** $\alpha \leq t$ **then**

9: Authenticate the peer device as an authorized device.

10: Find the first matching components of B_i and B'_i from X and Y , verify the consistency between the locally generated bits S with the bits S' from the peer device based on the following matching results.

11: Compute the edit distance between S and S' , find the matching bits and divide them into multiple blocks, calculate the hash code M' for each block M , i.e., $M' = \text{HMAC}(M)$, assemble them as a bit series, and extract the first k bits into the key K .

ALGORITHM 4: Mutual authentication and key extraction.

FIGURE 7: Decoding the exact bits for the peer device.

authentication code (HMAC) as the signature. Each device forwards a sliding window along the bit series S and calculates its HMAC on the bit series inside the window with a random number. For every step of 1 bit, each device calculates a new HMAC for the window. After that, each device exchanges the series of HMAC to the peer device. After receiving the HMAC series from the peer device, each device compares the *edit distance* [33] between the locally generated HMAC series and the received HMAC series. If the edit distance is below a specified threshold, then the peer device is authenticated as an authorized device; the mutual authentication is finished.

After that, according to the first matched HMAC and the following HMACs in the HMAC series, each device can locally figure out the exact bits of the following generated bits from the peer device. Figure 7 shows an example of decoding the exact bits for the peer device. Note that once device B gets the random number r_1 from device A, it calculates $\text{HMAC}(r_1, M'_1)$ over the bits M'_1 inside its local window. Since it obtains the same value 0010 as A, it implies that the two bit series M_1 and M'_1 are equal. Then, it forwards the window with 1 bit and further verifies the HMAC with A; this time, M_2 is not equal to M'_2 since they do not have the same HMAC. Based on the previous result, it can be implied that the only difference lies in the exact 1 bit; device B can thus infer the exact bits in M_2 . In this way, all bits from the peer device that follow the first matched bits can be effectively figured out. Note that, while exchanging the HMAC series between the peer devices, the attacker may intercept the message and try to solve M_i from $\text{HMAC}(r_i, M_i)$ using a brute force searching. However, as long as we use a large enough window size w , e.g., w is larger than 64 bits, the user may have little chance to crack it.

At last, each device can calculate the *edit distance* [33] between the locally generated bits S and the bits S' from the peer device. Based on this, the series of matching bits can be effectively identified, even if the two matching bit series have some time lag in bit generation. This can effectively solve the out-of-sync problem during the shaking process, since one device may generate the same bits with a little time delay after the other. In order to reduce the correlations among the bit series, we further divide the matching bits into multiple blocks, compute HMAC for each block, and assem-

ble them together as the final key. In this way, since the generated bits from each block all reflect the details of the user behavior, they always have some difference in both the time and space domains; as each bit in the output of HMAC is dependent on all bits in the input bit series, the generated bits can be fully randomized.

4.8. Security Analysis. In *eavesdropping attack*, attackers can launch eavesdropping attack by intercepting the transmitted messages in the open environment. Since the symmetric keys are generated independently among various devices and the messages broadcasted in key generation phase are only digital signatures used for verifying consistency, the attackers can never capture the actual value of the generated symmetric key.

In *man-in-the-middle attack*, attackers can launch man-in-the-middle attack between two devices in interaction. As all participating devices are communicating in the open wireless channel, any man-in-the-middle attack will be overheard and detected by the legal devices; these devices can then actively reject the replayed fake messages.

In *shoulder surfing*, attackers can conduct shoulder surfing by privately imitating the legal user's behaviors in handshaking. However, as long as any two devices are not shaking together, there will exist differences in the detailed profile and pattern of user behaviors. Our solution can accurately identify the slight difference to prevent generating the same symmetric keys.

5. Performance Evaluation

5.1. Experiment Settings. We implemented our authentication architecture on an Android 4.0 operating system. We let 14 users (6 males and 8 females), respectively, shake two homogeneous and heterogeneous devices 10 times using three patterns (back and forth, roll, cocktail). So we get $14 \times 10 \times 3$ sets of sensor data. Besides, we also let one user imitate shaking behavior of another user using homogeneous/heterogeneous devices. There are in total 20 pairs of users that will have to do the experiments, and each pair of users does the same experiment 10 times. So we get another $20 \times 10 \times 3$ sets of sensor data. We adopt WiFi as the insecure wireless channel. In the start of authentication, each device sends the beacon to scan the adjacent devices and connects to each other.

5.2. Evaluate the Accuracy of Trace Synchronization. In data calibration, each device independently detects the start point of sampling. In order to show the performance of the trace synchronization algorithm, each device sends a beacon to the server as long as it detects the start point. Once receiving two beacons, the server calculates the time difference. Figure 8(a) shows the time difference of three patterns. The time difference of three patterns are all lower than 14 ms, and the average is about 8 ms. As the sampling frequency of devices is about 50-100 Hz, so the data on two shaking devices has just few point offsets, which can be removed in data interpolation. Besides, the time difference of homogeneous smart device is smaller than the one of the

(a) Detect the start point

(b) The similarity value

(c) Self coefficient value

(d) Bit generation rate

(e) Matching ratio

(f) Key generation rate

FIGURE 8: Continued.

FIGURE 8: Performance evaluation.

heterogeneous device. This difference is caused by the sensor accuracy of the heterogeneous device. Overall, our algorithm could accurately detect the start sampling point.

5.3. Evaluate the Performance of Data Interpolation. In data interpolation, each device dynamically determines the smoothing window size w with the crosscorrelation coefficient t . Figure 8(b) shows the similarity of two sets of sensor data sampled from two shaking devices with different w . We can notice that as w increases, the similarity value increases, which means data interpolation effectively reduces the difference between sensor data by smoothing. Figure 8(c) shows the coefficient value of the sensor data with its own raw data with different w . We note that as w increases, the coefficient decreases. But the larger smoothing window size may remove the detail of the raw data. Fortunately, there is always a rapid decline of the coefficient value as w increases. So we can set t to a proper value, e.g., 0.85, to detect this change and this method works fine.

5.4. Evaluate the Ability of Key Generation. Our authentication architecture adopts six-dimensional sensor data instead of the amplitude of the sensor data. In order to compare the performance of those two methods, we use three metrics to evaluate the performance of our authentication architecture: *bit generation rate*, *matching ratio*, and *key generation rate*. Bit generation rate refers to the number of bits generated per second in the phase of quantization. Matching ratio refers to the percentage of the equal bits generated from two sets of sensor data in the phase of quantization. Key generation rate refers to the number of keys generated per second after key extraction. For simplicity, we represent the method of adopting six-dimensional sensor data as multidimensional method and the method of adopting the amplitude of the sensor data as one-dimensional method. Figure 8(d) shows the bit generation rate of two methods. We can see that the bit generation rate of the multidimensional method is far greater than the one of one-dimensional method. Besides,

the bit generation rate of data sampled by homogeneous device is always greater than the case of heterogeneous device. Figure 8(e) shows the matching ratio of two methods. Those two methods have the similar matching ratio, which means that the multidimensional method could generate more bits than the one-dimensional method using an equal set of sensor data. That is because the multidimensional method effectively utilizes each dimension of the data while removing the differences. Figure 8(f) shows the comparison between the key generation rate of two methods. Like the previous case, the multidimensional method generates more keys than the one-dimensional method.

5.5. Evaluate the Imitation Resistance. In order to show the antiattack capability of our authentication architecture, we let one user imitate the shaking behavior of another user. We set the quantization level as 2. Figure 8(g) shows the matching rate of bits separately generated by the data sampled from the devices shaken together and independently using the multidimensional method. Figure 8(h) shows the same value using the one-dimensional method. We notice that even when the devices are not shaken together, both two devices could generate a certain amount of equal bits. However, the success rate by shaking devices together is all greater than the success rate by imitating. So each device could preset a threshold value, e.g., 0.7, to remove the false-positive situation. Besides, comparing these two figures, we can find that the discrimination between shaking together and shaking independently by the multidimensional method is more larger than the one by the one-dimensional method. So our authentication architecture can effectively detect the imitation behavior with homogeneous and heterogeneous devices.

5.6. Evaluate the Randomness via NIST Test. In order to verify the randomness of the final key, we perform the NIST tests on the extracted keys. We concatenate the generated keys together as the input of the NIST test. Among the 15

TABLE 1: NIST statistical test suite result.

NIST test	Homogeneous devices	Heterogeneous devices
Freq.	0.5004	0.4339
Block Freq.	0.8843	0.8596
Cum. sums (Fwd)	0.4506	0.42485
Cum. sums (Rev)	0.7511	0.5945
Runs	0.3426	0.4605
Longest run os ls	0.7187	0.6342
FFT	0.5629	0.5856
Approx. entropy	0.9365	0.9201
Serial	0.9702/0.9991	0.8053/0.7230

tests in the NIST tool, we run 8 tests. As shown in Table 1, the extracted bits can pass these tests as all the values are greater than 0.01. So the key generated from our authentication architecture is secure.

6. Limitations and Discussions

6.1. Robustness to Tackle the Interference from Ambient Environment. The issues like the interference from ambient environment, e.g., inconsistent movements among the shaking devices, could lead to the generation of mismatched bits in the bit series among the peer devices in secure paring. To tackle these issues, first, we take measures to reduce the possibility of mismatched bits in the data calibration, filtering, and quantization. Moreover, we leverage the hash-based message authentication code (HMAC) as the signature to generate consistent keys. After exchange of the HMAC series among the peer devices, each device can locally figure out the exact bits in the generated key in a consistent manner.

6.2. Generalization to Tackle the Differences for Various Sensors. iShake uses the sensor readings of the accelerometer and gyroscope in 3 dimensions to generate the randomized key. If there exist any differences for various sensors and dimensions, in regard to the sensitivity, sampling rate, and similarity, it could lead to the performance degradation in mutual authentication and key generation. To tackle these issues, we actually already considered these differences in the data calibration, filtering, and quantization and take measures to mitigate the impact from these differences. In the final key generation phase, we can further use the HMAC signature to remove the inconsistent bits caused by the differences for various sensors.

6.3. Cost versus Benefit among Different Secure Pairing Solutions. iShake uses the inertial measurements to perform secure pairing, including mutual authentication and key generation among peer devices. Since inertial measurement unit (IMU) has been brought to standard configuration for most state-of-art smart devices, it is both low in cost and pervasive in use. Compared to the more advanced secure pairing solutions like the near-field communication (NFC), the IMU-based solution in iShake achieves high cost performance ratio and pervasive usage.

7. Conclusion

In this paper, we propose iShake, a mutual authentication and key generation framework for smart devices, by shaking these devices together. As compared to previous work, iShake generates highly distinctive and fully randomized key between devices, and the whole process is more resistant to imitation. The main contributions are as follows: (1) we conduct comprehensive experiments in real environment, illustrate novel observations, and extract important clues for efficient key generation; (2) we propose a series of novel techniques to make the generated key highly distinctive and fully randomized; and (3) we are the first to consider key generation among heterogeneous devices and address new challenging issues in generating unique and consistent keys.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

An earlier version of the manuscript has been presented at the IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS) in 2019. This work is an extended version of the earlier manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant Nos. 61872174, 61832008, and 61902175 and Jiangsu Natural Science Foundation under Grant No. BK20190293. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

- [1] L. Li, X. Zhao, and G. Xue, "A lightweight system to authenticate smartphones in the near field without NFC chips," in *2013 IEEE International Conference on Communications (ICC)*, pp. 6391–6395, Budapest, Hungary, 2013.
- [2] L. Li, X. Zhao, and G. Xue, "Near field authentication for smart devices," in *2013 Proceedings IEEE INFOCOM*, pp. 375–379, Turin, Italy, 2013.
- [3] M. Sethi, M. Antikainen, and T. Aura, "Commitment-based device pairing with synchronized drawing," in *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 181–189, Budapest, Hungary, 2014.
- [4] M. Shahzad, A. X. Liu, and A. Samuel, "Behavior based human authentication on touch screen devices using gestures and signatures," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2726–2741, 2016.

- [5] Y. Song, Z. Cai, and Z.-L. Zhang, "Multi-touch authentication using hand geometry and behavioral information," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 357–372, San Jose, CA, USA, 2017.
- [6] J. Lester, B. Hannaford, and G. Borriello, "Are you with me?-using accelerometers to determine if two devices are carried by the same person," in *International Conference on Pervasive Computing*, pp. 33–50, Springer, 2004.
- [7] R. Mayrhofer, "The candidate key protocol for generating secret shared keys from similar sensor data streams," in *European Workshop on Security in Ad-hoc and Sensor Networks*, pp. 1–15, Springer, 2007.
- [8] R. Mayrhofer and H. Gellersen, "Shake well before use: authentication based on accelerometer data," in *International Conference on Pervasive Computing*, pp. 144–161, Springer, 2007.
- [9] R. Mayrhofer and H. Gellersen, "Shake well before use: intuitive and secure pairing of mobile devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, 2009.
- [10] R. Marin-Perianu, M. Marin-Perianu, P. Havinga, and H. Scholten, "Movement-based group awareness with wireless sensor networks," in *International Conference on Pervasive Computing*, pp. 298–315, Springer, 2007.
- [11] K. Hinckley, "Synchronous gestures for multiple persons and computers," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pp. 149–158, Vancouver, Canada, 2003.
- [12] Y. Shen, F. Yang, B. Du, W. Xu, C. Luo, and H. Wen, "Shake-n-shack: enabling secure data exchange between smart wearables via handshakes," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, Athens, Greece, 2018.
- [13] Q. Jiang, X. Huang, N. Zhang, K. Zhang, X. Ma, and J. Ma, "Shake to communicate: secure handshake acceleration-based pairing mechanism for wrist worn devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5618–5630, 2019.
- [14] R. D. Findling, M. Muaz, D. Hintze, and R. Mayrhofer, "Shakeunlock: securely transfer authentication states between mobile devices," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 1163–1175, 2016.
- [15] S. Muhammad, A. X. Liu, and A. Samuel, "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 - MobiCom '13*, pp. 39–50, Miami, Florida, 2013.
- [16] W. Xu, G. Revadigar, C. Luo, N. Bergmann, and W. Hu, "Walkie-talkie: motion-assisted automatic key generation for secure on-body device communication," in *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 1–12, Vienna, Austria, 2016.
- [17] S. Mare, R. Rawassizadeh, R. Peterson, and D. Kotz, "Continuous smartphone authentication using wristbands," in *Proceedings of the Workshop on Usable Security*, pp. 1–12, Stockholm, Sweden, 2019.
- [18] H.-M. C. Leung, C.-W. Fu, and P.-A. Heng, "TwistIn," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 2, pp. 1–24, 2018.
- [19] S. Mare, R. Rawassizadeh, R. Peterson, and D. Kotz, "SAW: wristband-based authentication for desktop computers," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–29, 2018.
- [20] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the effectiveness of secret key extraction from wireless signal strength in real environments," in *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, pp. 321–332, Beijing, China, 2009.
- [21] X. Zhu, F. Xu, E. Novak, C. C. Tan, Q. Li, and G. Chen, "Extracting secret key from wireless link dynamics in vehicular environments," in *2013 Proceedings IEEE INFOCOM*, pp. 2283–2291, Turin, Italy, 2013.
- [22] H. Liu, J. Yang, Y. Wang, and Y. Chen, "Collaborative secret key extraction leveraging received signal strength in mobile wireless networks," in *2012 Proceedings IEEE INFOCOM*, pp. 927–935, Orlando, FL, USA, 2012.
- [23] H. Liu, Y. Wang, J. Yang, and Y. Chen, "Fast and practical secret key extraction by exploiting channel response," in *2013 Proceedings IEEE INFOCOM*, pp. 3048–3056, Turin, Italy, 2013.
- [24] C. Castelluccia and P. Muta, "Shake them up!: a movement-based pairing protocol for CPU-constrained devices," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pp. 51–64, Seattle, Washington, 2005.
- [25] T. Zhang, X. Yi, R. Wang et al., "Tap-to-pair: associating wireless devices with synchronous tapping," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, pp. 1–21, 2018.
- [26] S. Mei, Z. Liu, Y. Zeng, L. Yang, and J. F. Ma, "Listen!: audio-based smart IoT device pairing protocol," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, pp. 391–397, Xi'an, China, 2019.
- [27] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen, "Smart-its friends: a technique for users to easily establish connections between smart artefacts," in *International Conference on Ubiquitous Computing*, G. D. Abowd, B. Brumitt, and S. Shafer, Eds., pp. 116–122, Springer, 2001.
- [28] W. Xu, C. Javali, G. Revadigar, C. Luo, N. Bergmann, and W. Hu, "Gait-Key: a gait-based shared secret key generation protocol for wearable devices," *ACM Transactions on Sensor Networks*, vol. 13, no. 1, pp. 1–27, 2017.
- [29] G. Revadigar, C. Javali, W. Xu, A. V. Vasilakos, W. Hu, and S. Jha, "Accelerometer and fuzzy vault-based secure group key generation and sharing protocol for smart wearables," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2467–2482, 2017.
- [30] D. Bichler, G. Stromberg, M. Huemer, and M. Low, "Key generation based on acceleration data of shaking processes," in *International Conference on Ubiquitous Computing*, pp. 304–317, Springer, 2007.
- [31] D. Bichler, G. Stromberg, and M. Huemer, "Synchronizing shaking sequences for generating symmetric keys," in *2009 2nd International Workshop on Nonlinear Dynamics and Synchronization*, pp. 75–80, Klagenfurt, Austria, 2009.
- [32] Y. Shen, B. Du, W. Xu et al., "Securing cyber-physical social interactions on wrist-worn devices," *ACM Transactions on Sensor Networks*, vol. 16, no. 2, pp. 1–22, 2020.
- [33] E. Ukkonen, "On approximate string matching," in *Foundations of Computation Theory*, pp. 487–495, Springer, 1983.