TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 pp1–19
DOI: 10.26599/TST.2026.9010051
Volume x, Number x, xxxx xxxx

Research Article

Cross-App Keystroke Inference Through Malicious In-App Training on Keystroke-Induced Vibrations

Ruyi Wei†, Junhua Wu†, Huigui Rong, and Daibo Liu*

Abstract: Keystroke privacy is critical on mobile devices, as typed content often includes passwords and sensitive personal data. This paper revisits cross-Application (App) keystroke inference and shows that software sandboxing cannot prevent keystroke leakage at the physical layer because co-resident Apps inevitably share the same hardware. Our insight is twofold. (1) Keystroke-induced vibrations propagate through the touchscreen and internal components and are captured by zero-permission motion sensors, preserving spatial traits tied to key positions. (2) Typing habits enable cross-App generalization, as users typically rely on the same Virtual Keyboard (VK) layout across Apps, causing identical letters to yield similar vibration patterns. Based on these properties, we propose Cross-Apps Mirror Inference (CaMi), a practical cross-App keystroke inference attack. CaMi learns letter-level vibration signatures using labeled samples in any commonly used App and then covertly collects motion sensor data to infer keystrokes typed in a target App. We implemented CaMi on smartphones and tablets, and extensive experiments demonstrate high inference accuracy across devices. We also discuss potential software-level defenses.

Key words: smartphone; keystroke inference; cross-App attack; vibrations; side-channel

1 Introduction

In modern societies, mobile devices (e.g., smartphones and tablet computers), along with their software applications, have penetrated into people's daily life pervasively. Mobile devices are typically equipped with an on-screen Virtual Keyboard (VK) to perform textual and numeric data input. Keystroke privacy is critical to ensuring the security of mobile systems and the privacy of common users as what is being typed could be passwords or privacy-sensitive information, such as chat content, trade information, account information, or private emails. Due to its high privacy sensitivity, current mobile systems (e.g., iOS and Android) adopt data isolation techniques[1–3], namely virtual isolation, to prevent rogue mobile applications (Apps) from eavesdropping on the input content entered through the VK. In other words, the input content can only be captured by the App in use.

Although virtual isolation methods perform well in preventing information leakage among cohabited mobile Apps at software level, physical isolation is impossible because all Apps run in the same hardware modules, thus offering a shared medium and posing a potential channel for Apps running in background to perceive the keystroke-induced physical responses sourcing from the foreground App. If the response's physical features are unique to each VK-letter and the VK-layout is known to co-resident rogue Apps, the keystroke content can be mirrored by the Keystroke-

• Ruyi Wei, Huigui Rong, and Daibo Liu are with College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China. E-mail: ruyiwei@hnu.edu.cn; hgrong@hnu.edu.cn; dbliu@hnu.edu.cn.
• Junhua Wu is with Hunan University and Public Security Department of Hunan Province, Changsha 410082, China. E-mail: junhuaw@gmail.com.
† Ruyi Wei and Junhua Wu are the co-first authors.
* To whom correspondence should be addressed.
Manuscript received: 2025-12-23; revised: 2026-03-19; accepted: 2026-05-18

© The author(s) 2026. The articles published in this open access journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/).

Induced Vibrations (KiVi) signals and hence can easily be inferred by any commonly used App (if it chooses to go rogue). We refer to such privacy threat as Cross-Apps Mirror Inference (CaMi). This phenomenon motivates us to investigate the trustworthiness of keystroke privacy isolation between the App in use and potential rogue Apps in background, as well as the security implication on VKs involving every mobile user.

Previous attacks have demonstrated the feasibility of inferring keystrokes by accelerometer signals. For example, TapLogger[4] utilizes the accelerometer and gyroscope to infer password and Personal Identification Numbers (PINs), However it can only recognize the numeric keyboard, rather than 26 English letters. TapPrints[5] can recognize 26 English letters well on the tablet, but it cannot be deployed on mobile phones because of the smaller screen. These attacks hardly pose a threat to mobile phone users as most account and passwords consist of English letters. What is more, their preparation and attack phases are located on different devices and it is not possible to effectively label signal samples to achieve supervised training. Different from these attacks, our data collection and attack are both conducted on the same device, so we can better implement the attack.

In this paper, we identify a new class of system-level vulnerabilities that can be used by rogue Apps to collapse existing privacy isolation. As shown in Fig. 1, a rogue App can easily gain the access to zero-permission IMU sensors and stealthily collect accelerometer data when running in the background. In this way, it can detect potential keystroke actions generated from foreground App and record these keystroke-induced vibrations, referred to as KiVi. With

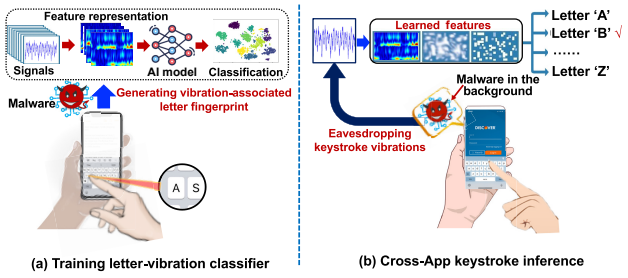


Fig. 1 Attack scenario of CaMi. When victim enters sensitive information in the foreground App with a VK, the rogue App running in the background can collect the KiVi signals from accelerometer sensor and infer the victim's input content by querying a pre-trained KiVi-letter dictionary.

such KiVi signals, the rogue App can further infer the entered sensitive information by querying a pre-trained KiVi-letter inference model. We call such attacks CaMi. The key principles behind CaMi are as follows: **(1) Key-position relevant KiVi.** For each keystroke on VK, the frequency distributions and harmonics of the produced KiVi are closely related to the internal structure of the vibrating source behind the key-position. Due to irregular interior structure, the KiVi signals retain unique traits dependent on the key-position. **(2) Non-isolatable physical vibrations.** Keystroke on a VK's key-position could stimulate the vibration of both touchscreen and components inside the device behind key-position. Such KiVi signals cannot be completely physically isolated and can be perceived by the inbuilt accelerometer sensor. **(3) Realistic assumption.** Most if not all mobile users are accustomed to using the same preferred on-screen VK for textual and numeric input among different Apps (see Fig. 2 and our investigation report in Table 1). The keyboard layout is dependent on the touchscreen attribute but irrelevant to Apps[6]. Thus to enter the same letter, KiVi signals created in different Apps source from the same key-position and possess similar traits. On this basis, a mobile App with the need for text input can effortlessly train a KiVi-letter classifier with supervised learning.

It is also worth noting that any commonly used App※ could be motivated to conduct CaMi attack if it can be easily done without leaving any trace, because creating a comprehensive user profile involving multidimensional privacy can bring huge benefits

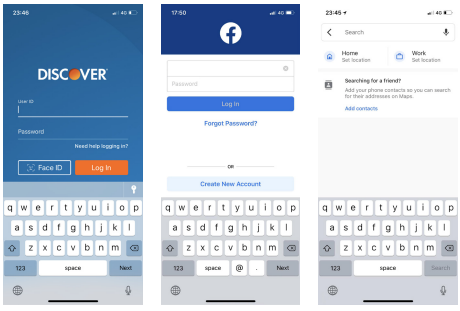


Fig. 2 Layout of the same on-screen VK, i.e., default iPhone input method, in three different smartphone Apps: Discover credit card, Facebook, and Google maps. The VK at different APPs is almost with the same layout, which depends on the screen attribute rather than Apps.

※Our goal is to reveal the potential privacy threat, rather than to point out which commonly used App is malware.

Table 1 User usage of VK.

	Fixed	Change occasionally	Change often
Male	34	9	3
Female	30	11	3
20–30	10	4	1
30–40	36	9	3
40–50	14	4	2
50+	14	3	0

through either providing satisfactory services or employing malicious means. Due to zero-permission, using IMU sensors in the background will not be suspected by the system. Moreover, despite rigorous process-killing strategies in current mobile OS, the plug-and-play feature makes CaMi effective to infer keystroke content without the need for continuously eavesdropping in the background, thus existing process management policies will have no real effect on the effectiveness of CaMi. However, to achieve this goal, we still need to answer the following research questions.

- **RQ 1:** How to characterize the KiVi traces?
- **RQ 2:** How to beat the impact of interaction behavior?
- **RQ 3:** How to train KiVi-letter classifier and reliably detect KiVi in a stealthy way?

To address **RQ 1**, we begin by exploring the generation and propagation of the vibration response through a series of experiments to investigate the signal properties specific to the key position (Section 5.2). Then, we propose a fine-grained feature extraction scheme to enhance the uniqueness of vibration features from different keystroke positions, and maintain consistency of the vibration features for the same keystroke position.

On that basis, we trained a neural network to distinguish features at different keystroke positions (Section 6.4).

To address **RQ 2** to generalize to different finger-screen interaction behaviors, CaMi aims to achieve a consistent feature space representation in different on-screen finger-screen interaction environments, by harnessing the power of supervised contrastive learning[7] to extract more abstract and universal features. This allows CaMi to focus more on the differences between the signals of different positions and thus better differentiate between the different keystrokes.

To address **RQ 3**, we developed a mobile application designed to collect and classify KiVi using the built-in sensors of smartphones. The App captures raw signal data and preprocesses it through noise reduction, normalization, and feature extraction techniques to ensure high-quality inputs. A lightweight machine learning model, trained either on-device or via cloud-based processing, is employed to classify the signal efficiently. To enhance stealth and reduce power consumption, the App uses optimized background services, allowing it to detect KiVi patterns intermittently without disrupting normal phone operations. This approach not only ensures reliable signal detection but also maintains user privacy by processing data locally and minimizing network transmissions.

We implemented a prototype of CaMi with off-the-shelf smartphones with Android systems. We conduct a suite of IRB-approved real-world experiments to validate the effectiveness of CaMi. Extensive experiments demonstrate the effectiveness of our attacks in a wide range of COTS mobile devices. The results show that CaMi achieves an average accuracy of 94.81% for inference on typed keys. Given that CaMi is based on supervised learning, we treat adjacent keys as potential candidates for inference, leveraging their proximity as a context for predicting key presses. In a nutshell, our main contributions are summarized as

- To the best of our knowledge, this is the first work to identify vulnerabilities in all modern mobile devices for Cross-Apps Keystroke Inference via bypassing existing virtual isolation.
- We propose CaMi, the first KiVi-based attack against keystroke privacy isolation among mobile Apps.
- We have considered a series of practical and effective solutions to defend against the CaMi attack in Section 8.

2 Related Work

Existing keystroke inference attacks leveraging side-channel information can be broadly classified into the following categories.

2.1 Side-channel attacks using motion sensors

Embedded motion sensors, such as accelerometers and gyroscopes, are widely utilized in mobile applications for motion tracking and command-based interaction. However, they also pose a potential risk of leaking sensitive user information. These sensors can detect

changes in motion and device orientation, which may result in privacy breaches[4, 5, 8, 9]. One of the prominent threats involves location tracking, where built-in motion sensors are used to track users' movements via inertial navigation models[10–12]. Narain et al.[10] highlighted privacy issues related to calculating a user's location and travel patterns using side-channel data. In addition, device fingerprinting attacks, which enable adversaries to track and identify a user's device over time, have been explored[11, 12]. Recent studies have shown how motion sensors can be employed for more complex attacks, such as inferring handwriting[13], factory floor secrets[14], and objects printed by nearby 3D printers[15]. Additionally, Chen et al.[16] introduced ViFin, a novel technique using finger movement-induced vibrations for tracking micro-level finger movements with a commodity smartwatch.

2.2 Keystroke inference attacks

Video-based attacks. Video-based attacks leverage computer vision techniques to analyze typing processes, recording either hand movements[17, 18], eye movements[19–21], plate back movements[22], or reflections from nearby objects (e.g., glasses and plastic bottles)[23–25]. GAZEploit[26] achieved over 80% accuracy in remote keystroke inference by exploiting gaze patterns, identifying vulnerabilities in over 15 top-ranked websites. Despite their feasibility, video-based attacks are constrained by the need for a clear line of sight and fail in poor lighting conditions.

Radio signal-based attacks. With the widespread deployment of wireless infrastructures, such as Wi-Fi and cellular towers, researchers have explored the possibility of eavesdropping on keystrokes via radio signals. Studies[27–31] have demonstrated the potential to capture keystrokes through Wi-Fi Channel State Information (CSI), while Ling et al.[32] showed how cellular signal reflections can reveal ATM PINs. Wang et al.[33] enabled multi-person keystroke inference using BFI-enabled Wi-Fi sensing. However, these attacks are highly sensitive to environmental changes and require stable conditions to be effective, limiting their practical application.

Electromagnetic (EM) emanation-based attacks. EM radiation unintentionally emitted by electronic devices has been explored as a side-channel for inferring keystrokes, particularly on physical keyboards[34–37]. Unlike traditional keyboards, soft keyboards rely on touch positions on the screen, making these eavesdropping methods less applicable in modern mobile environments.

Acoustic signal-based attacks. Acoustic cryptanalysis exploits sound emissions from devices to infer sensitive information, especially keystrokes. Some attacks focus on sound feedback from keypads, such as ATMs or door keypads, where each keystroke produces a distinct sound[38, 39]. Researchers have also examined using nearby smartphones to record keystrokes on external keyboards[40–43] or by analyzing reflected sounds from the user's fingers while typing[44]. Similarly, keystroke inference has been studied for smartphones[45, 46] and laptops[47] using external devices like smartwatches or cameras[48]. Feng et al.[49] and Narain et al.[50] used onboard sensors, including a microphone, for keystroke recognition. Shumailov et al.[51] employed Time Difference of Arrival (TDoA) measurements between microphones to recognize keystrokes. TapSnoop[52] and TapLeak[53] leveraged advanced signal processing techniques and deep learning to improve accuracy in keystroke recognition. Liu et al.[54] used an ear microphone for identity verification.

2.3 Motion sensor-based keystroke inference attacks

Keystroke inference using mobile device motion sensors remains a critical area of research. Earlier work primarily used accelerometer data to infer user passwords[8, 9], with gyroscopes later incorporated to enhance accuracy[4, 5]. Recent studies have extended these techniques to wearable devices, such as smartwatches, to monitor user input[55, 56]. However, these approaches fail to address key challenges, such as accurately recognizing complex keystrokes and mitigating the impact of user behavior.

2.4 Innovations in CaMi

CaMi is not the first to explore keystroke inference through keystroke-induced vibrations, but it offers significant improvements over prior work. Techniques such as TouchLogger[8], TapLogger[4], Accessory[9], TapPrints[5], and (sp)iPhone[57] utilized accelerometers and gyroscopes to infer keystrokes on physical or smartphone keyboards. These studies primarily focused on the importance of the typed information in the context of online activity[4, 8]. In contrast, CaMi introduces a novel contrastive learning network to effectively mitigate the influence of user behavior,

enabling accurate recognition of English-letter keystrokes and revealing privacy risks that were previously not detectable.

3 Background

3.1 Sandbox-based cross-app data isolation

Sandbox-based cross-App data isolation is fundamental to ensuring application security and user privacy in Android and iOS, effectively preventing unauthorized data access and inter-App interaction.

In **Android**, each App operates within an isolated sandbox, with its own directory and a unique User Identifier (UID), ensuring strict data and execution separation. Android's permission system requires Apps to explicitly declare resource access needs, with users granting permissions during installation or runtime. Security-Enhanced Linux (SELinux) further enforces fine-grained access control, strengthening isolation at the system level.

In **iOS**, all Apps run under a common "mobile" user identity but are confined to individual directories via a chroot jail. The App Sandbox model tightly restricts filesystem access and mandates user consent for external data access. Apps must obtain specific entitlements through a rigorous review process, and system call restrictions further limit unauthorized behaviors. Additionally, iOS encrypts App data using a key derived from the user's passcode, enhancing data protection.

Overall, both Android and iOS employ sandboxing to securely isolate applications, significantly mitigating the risks of data breaches and unauthorized access.

3.2 VK and user usages

VK. A VK is a software-emulated keyboard that operates independently of physical keys. It is a standard feature on modern touchscreen devices such as smartphones, tablets, smart TVs, and other devices lacking physical keyboards, offering a primary mode of user interaction[6]. When users input text via a VK, applications listen to the input through designated input fields and can retrieve the entered content in real time.

Protecting keystroke privacy is critical, as user inputs often include passwords and other sensitive information. Modern mobile operating systems implement isolation mechanisms, i.e., sandboxes, to safeguard user inputs, ensuring that only the active application can access input data. Consequently, background applications cannot intercept coordinate information or keystrokes generated by users interacting with a VK.

User usage. Users typically maintain a consistent preference for a specific virtual keyboard—commonly default keyboards provided by Google, Apple, or third-party vendors—when entering text and numerical data across applications. As shown in Table 1, a survey of 90 participants indicated that 64 consistently used a single preferred VK, while 20 occasionally switched to alternative keyboards.

The VK layout remains highly uniform across different applications, primarily dictated by the smartphone's screen dimensions, as illustrated in Fig. 2. Users interact with the VK by tapping virtual keys with their fingertips to input information, including highly sensitive data such as account credentials and passwords.

3.3 Motion sensors in modern mobile devices

Motion sensors are essential components of modern mobile devices, designed to measure and record motion-related physical properties. Contemporary smartphones are typically equipped with a three-axis accelerometer and a three-axis gyroscope, both based on Micro-Electro-Mechanical Systems (MEMS) technology and highly sensitive to device movement.

Taking the MEMS accelerometer as an example, it comprises three primary components: A seismic mass, flexible spring legs, and stationary electrodes, as illustrated in Fig. 3. The seismic mass is suspended above the substrate via spring legs. When acceleration is applied, the mass displaces, causing a change in capacitance between the fixed electrodes, which is measured to determine the magnitude and direction of the acceleration. A three-axis accelerometer integrates three independent sensor structures to capture acceleration along the orthogonal x, y, and z axes. It continuously monitors the device's instantaneous acceleration in three spatial dimensions, with outputs typically expressed in meters per second squared (m/s^2).

These motion sensors enable critical functionalities such as orientation detection, vibration sensing, and shock monitoring, and have been extensively adopted across diverse mobile applications. However, their open accessibility also introduces significant privacy risks. Accessing motion sensors generally does not require explicit user permission, allowing third-party applications to collect motion data without user

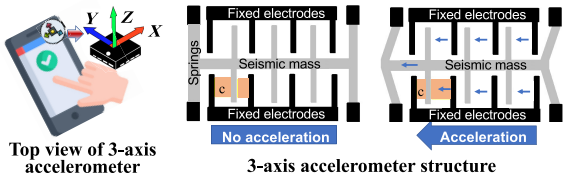


Fig. 3 Structure of MEMS 3-axis accelerometer.

authorization. This unrestricted access creates a potential side channel through which sensitive user activities could be inferred, posing a serious threat to privacy and security.

3.4 Availability of accelerometer running in the background

This section explores the feasibility of third-party applications collecting KiVi signals during background operation.

Unrestricted access to accelerometers during typing. Our research investigates a range of smartphone models and reveals that applications, given the appropriate permissions, can consistently access the device's accelerometer to capture KiVi signals while a user types on the VK. The study surveyed 16 smartphone models from various brands, including the Samsung S8, S9, S10, S20, Apple iPhone 8 through 14, OPPO R17, Huawei P20, Mate 20, Honor 20, Google Pixel 4, Nexus, OnePlus 8 and 9, and Xiaomi Redmi K40 and Mix 4. We found that neither smartphone manufacturers (such as Samsung, Apple, and OnePlus) nor the operating system platforms (Android and iOS) impose restrictions on the accelerometer's functionality during VK input.

Signal consistency across diverse applications. Further investigation was conducted to assess the stability of accelerometer signals across different applications during typing. As shown in Fig. 4, when typing the letter "H" in applications such as "Discover mobile", "Facebook", and "Google maps", the accelerometer signals exhibit remarkable consistency. This observation suggests that the context of the App does not significantly affect the accelerometer's output, indicating that KiVi signals can be reliably captured regardless of the specific application in use.

4 Threat Model

The adversary aims to infer the victim's keystroke input on a VK. When the victim begins typing on their mobile device, CaMi operates in the background, covertly capturing the KiVi signals generated by each keystroke on the touchscreen. These signals are transmitted to the adversary's cloud server, where they are analyzed to reconstruct the victim's input content.

To collect and transmit KiVi samples, the adversary must have access to the victim's smartphone IMU sensors. This requires installing malicious software on the victim's device with minimal permissions, such as internet connectivity. Importantly, adversaries in this scenario could be established commercial services like Facebook, TikTok, Tinder, Hinge, or other similar entities, which have a vested interest in constructing detailed user profiles. These services may have obtained the necessary permissions through legitimate means to fulfill their basic functional requirements, such as enhancing interactivity, health tracking, or voice messaging. Once these permissions are granted, the malicious App can precisely label the KiVi signal generated by each keystroke during user input within the App. By accumulating these labeled signals, the App can perform supervised training in the background, mapping KiVi signal features to specific characters.

To achieve this, we make the following assumptions.

- We assume that the attacker can run malicious software in the background of the victim's phone to collect KiVi samples and send them to a remote server. The malicious software can disguise itself as a legitimate App or game, distributed through the App store, and could even be a widely used commercial App. These applications already have the necessary permissions for their intended functionality.
- We assume that the victim uses a standard QWERTY virtual keyboard with a fixed layout across

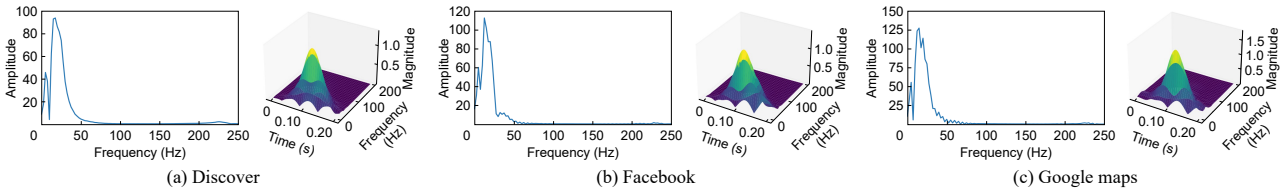


Fig. 4 Example of ASD and spectrogram of vibration signals generated by tapping the same character "H" in different Apps.

Apps. In this way, the spatial position of each key remains consistent, and the vibration patterns for the same character are comparable across different applications.

- We assume that the operating system's vibration feedback for each keystroke is disabled to ensure cleaner signal collection. In practice, this feature is rarely used by the majority of users.

5 Principle, Feasibility, and Challenge

In this section, we preliminarily introduce the basic principles of KiVi and the inherent characteristics that are relevant to clicking position on touchscreen.

5.1 Primer on KiVi

Keystroke on touchscreen for text-input. Modern smartphones are typically equipped with a three-axis accelerometer to monitor user movements. For text input, users generally rely on a preferred input method (e.g., Google or Apple input) with a fixed keyboard layout, tapping individual letters with their fingertips. Importantly, each letter is selected by tapping the same location on the screen. When typing, the action of tapping the screen generates vibrations within the device's internal components. To ensure accuracy and efficiency, users naturally lift their fingertips from the touchscreen after each keystroke. Consequently, the vibrations from the screen and its internal components, rather than the fingertip itself, become the primary source of KiVi. These vibrations propagate through the device and can be detected by the built-in accelerometer.

5.2 Can we distinguish KiVi by keystroke position?

Based on these observations, we conducted experimental validations to assess the feasibility of CaMi. For each keystroke, the vibration characteristics depend on the specific screen position, as well as the particular finger and device being used. These variations can be captured through Amplitude Spectral Density (ASD) profiles and spectrograms, which enable us to distinguish between different keystroke positions. Figure 5 illustrates the layout of Gboard (Google Keyboard). Figure 6 presents the ASD profiles and spectrograms for keystrokes on the letters "H", "J", and "P". As seen in Figs. 6a and 6d, the frequency spectra of "H" and "P", which are located at distant positions on the keyboard, differ significantly. While



Fig. 5 Layout of letters in Google input.

the adjacent letters "H" and "J" show less pronounced differences, there remains a detectable variation. Additionally, Figs. 6a and 6b present an example of a user typing the same letter ("H"). The ASD profiles and spectrograms from both instances exhibit clear similarities, supporting the consistency of this method.

However, we must also account for the influence of user behavior and external disturbances, which present significant challenges. Variations in user behavior and external interference can impact the results, as different factors contribute to distinct variations in the vibration characteristics.

5.3 Barrier to practicability

Overcoming the impact of user behavior and external disturbance. Unlike active vibrations, KiVi is passively triggered by finger clicks, and its pattern is influenced by several factors, including the internal structure related to the keystroke position, the characteristics of the keystroke finger and the hand holding the smartphone, as well as external disturbances. Variations in finger-screen interaction—such as user behavior, external interference, slight deviations in keystroke positions, changes in hand posture, and differences in keystroke strength—can significantly impact the classification of KiVi characteristics, as shown in Figs. 6e and 6f. One potential solution is to collect a large and diverse dataset that covers a wide range of scenarios for model training. However, this approach is time-consuming, inefficient, and requires user participation, which may not always be feasible. Thus, identifying intrinsic features that can consistently and accurately capture the unique characteristics of keystroke positions remains a critical and challenging task.

Accurately detecting KiVi. Since the rogue App operates in the background, the first crucial step in inferring the keystroke content represented by KiVi signals is reliable detection of KiVi in real-time. Furthermore, the attack must be efficient enough to conceal its purpose without raising suspicion.

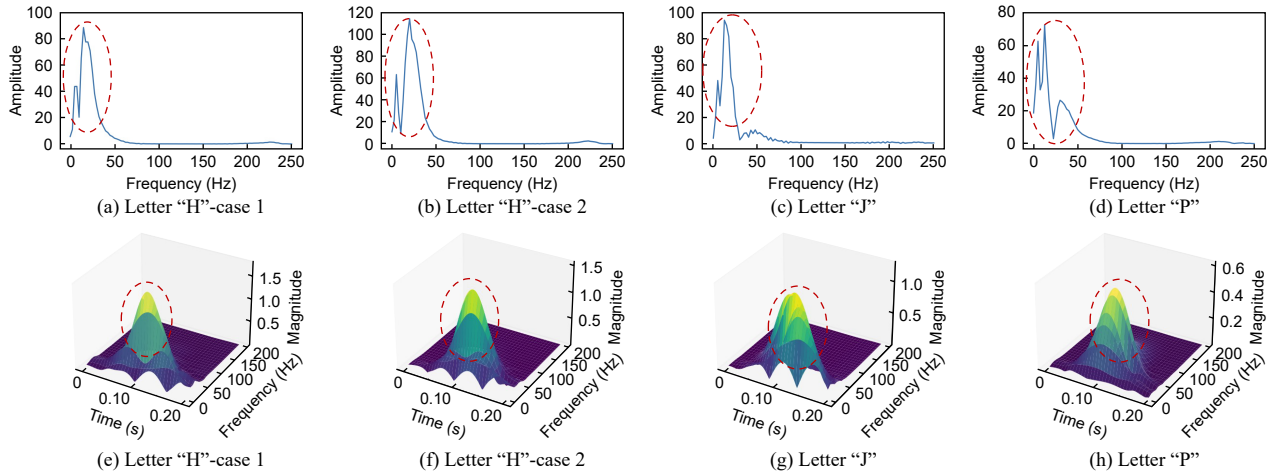


Fig. 6 ASD and spectrograms of KiVi signals for keystrokes on the same or different letters on the same smartphone.

6 CaMi Design

The system overview of CaMi consists of three main components: KiVi signal detection, KiVi preprocessing, and automatic inference on sensitive inputs. This section describes the mechanisms used for processing, segmenting, and detecting KiVi signals.

6.1 Signal preprocessing

The goal of signal preprocessing is to extract clean KiVi segments corresponding to individual keystrokes, even in the presence of ambient noise. However, KiVi signals are susceptible to distortion caused by user behavior and external factors (e.g., speaker vibrations). To address these challenges, we apply the following preprocessing steps to refine the raw KiVi signals.

Interpolation. Due to data instability, obtaining accurate frequency-domain characteristics can be challenging. To address this, CaMi employs linear interpolation to generate new data points at a 1000 Hz sampling rate, ensuring evenly spaced samples. The primary goal is to produce acceleration signals with a consistent sampling rate, as interpolation does not introduce any new information into the KiVi signal.

High-pass filtering. Human behavior generates interference signals primarily below 50 Hz, making a high-pass filter effective for removing such interference from the KiVi signal. KiVi signals typically occur above 20 Hz. Therefore, we apply high-pass filter with a cutoff frequency of 20 Hz to retain as much KiVi information as possible. While the filter removes a significant amount of noise, some residual noise remains, presenting an ongoing challenge.

Segmentation. Vibration signals are continuous in the time domain, so we need to segment them into KiVi segments for classification. The segmentation process follows these steps: First, we compute the absolute value of the signal and apply two rounds of exponential moving average smoothing, using sliding windows of 200 and 30, respectively. Next, we identify vibration segments by calculating a cut-off threshold as $0.35\times\text{Max} + 0.5\times\text{Min}$, where Max and Min are the maximum and minimum amplitudes from real keystroke samples as shown in Fig. 7. Segments are defined between the points where the signal intersects this threshold. To compensate for the narrowing effect of the moving average, we adjust the segment boundaries by shifting the points 100 units in both directions along the horizontal axis.

6.2 On-screen KiVi detection

The previous preprocessing step standardizes the data by filtering out continuous ambient noise caused by

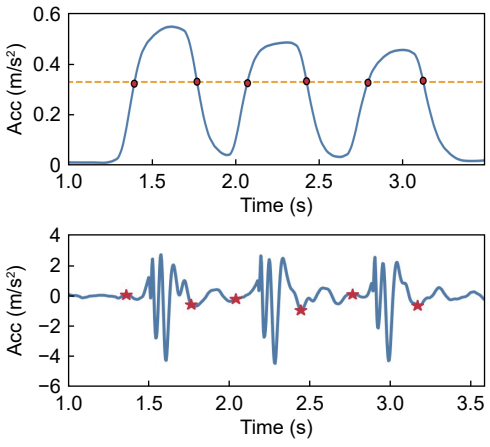


Fig. 7 Segmentation outcomes on the raw signal data.

human behaviors. However, it still retains noise from bursts of sound with dominant frequency components below 500 Hz, such as loudspeaker vibrations. Therefore, before proceeding to position-based KiVi classification, two key prerequisites must be met: (1) Reliable detection of KiVi samples at runtime, and (2) accurate identification of intentional on-screen inputs by eliminating other KiVi events, such as swiping or casual screen taps.

Time–frequency analysis. Although a high-pass filter effectively removes noise from human activities, high-frequency interference from sources such as ringtones or speaker vibrations often persists, typically above 100 Hz. As shown in Fig. 8, these sources can be differentiated by analyzing their temporal and spectral characteristics. A key observation is that KiVi samples consistently exhibit shorter and sharper durations in the time domain. Our research indicates that ringtones and speaker vibrations typically last at least 0.8 and 1 s, respectively, whereas over 99% of KiVi samples have durations of less than 0.2 s. Additionally, KiVi samples demonstrate smoother characteristics in the frequency domain. In contrast, motion-induced signals, as seen in the figure, contain primarily low-frequency components and have a duration that is roughly twice that of KiVi. Based on these observations, we leverage spectral smoothness characteristics[58] within each time window to quantify smoothness, and employ range characteristics to assess the difference between the highest and lowest amplitudes within a segment.

KiVi detection. Upon detecting onscreen activities, CaMi proceeds to the confirmation stage, aiming to distinguish KiVi from speaker vibrations. An unknown signal is classified as a KiVi sample only if it meets the criteria established by the preceding time–frequency analysis.

6.3 Feature extraction

Empirical studies suggest that both the time and frequency domain features of KiVi signals carry distinct characteristics of the keystroke position. Therefore, we can extract the spectrogram for KiVi features.

Spectrogram. To generate the spectrogram of the KiVi signal, we first divide the signal into multiple small segments with a fixed overlap. The overlap is set to 126 samples, and the window length is set to 128 samples. A sliding window then passes through the sequence, applying a Hamming window to each segment. The spectrogram is computed via Short-Time Fourier Transform (STFT), generating a series of complex coefficients for each grid. The signal along the z-axis is transformed into an STFT matrix that reflects the magnitude of the signal. The spectrogram is calculated as

$$\text{spectrogram } x(n)(m, w) = |\text{STFT}(x(n)(m, w)|^2 \quad (1)$$

where $x(n)$ is the z-axis acceleration signal, and $|\text{STFT}(x(n)(m, w)|$ is the magnitude of its corresponding STFT matrix.

To feed the spectrogram directly into a neural network used in computer vision tasks, we further convert the matrix into a tensor of dimensions $m \times n$, mapping the resulting values to integers between 0 and 255. Prior to this, we take the square root of each element in the spectrogram matrix to prevent data overflow, as most element values lie within the range $[10^{-7}, 10^{-5}]$.

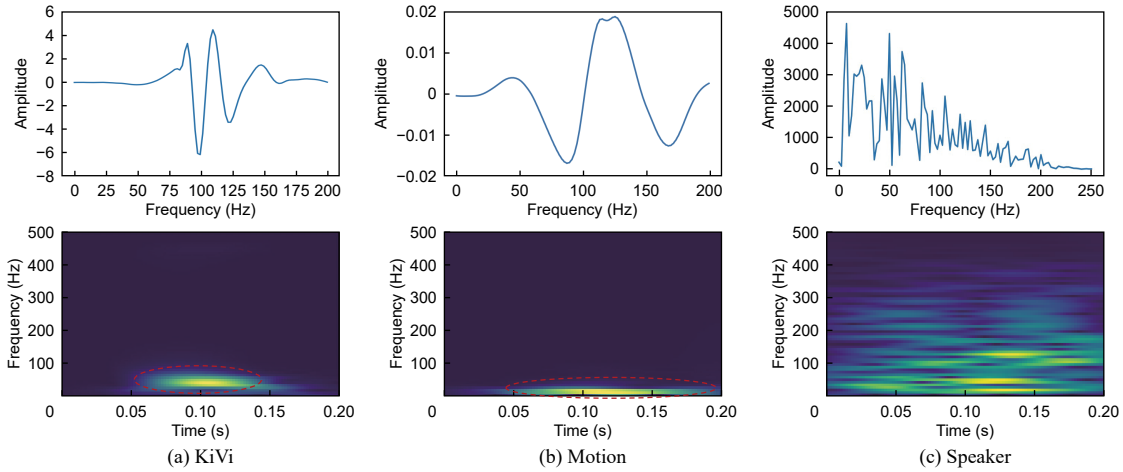


Fig. 8 ASD and spectrogram of vibration caused by KiVi, motion, and speaker.

6.4 Supervised training in malicious Apps

Acquire paired accelerometer–key data. When a user interacts with a malicious App (e.g., typing in a chat or search box, as discussed in Section 3.2), the App can capture each keystroke in real time. Simultaneously, the App can access accelerometer data without requiring explicit permissions. This enables the malicious App to easily obtain a paired dataset of keystrokes and their corresponding accelerometer signals.

Keystroke classifier training. In this section, we explain how contrastive learning is used to generalize across unseen domains, where "domains" refer to variations in user behavior and external disturbances.

Factors such as typing strength, speed, the user's condition, and the surrounding environment all influence keystrokes. A straightforward solution would be to gather a comprehensive dataset covering a wide range of scenarios for training. However, this approach is time-consuming, inefficient, and impractical, as it requires extensive user participation. Moreover, it is impossible to capture every possible scenario.

The variability in user behavior and external disturbances necessitates a robust method to ensure key inference's effectiveness across diverse conditions. To address this, we employ the concept of contrastive learning[59]. CaMi leverages supervised contrastive learning[7] to create a consistent feature space representation across different domains, helping to extract more abstract and generalizable features, as illustrated in Fig. 9.

During the training phase, we first prepare a dataset consisting of randomly paired KiVi signal spectrograms (x_1 and x_2), labeling them as $a = 0$ if the signals correspond to the same key, and $a = 1$ otherwise. Each spectrogram is passed through the feature extractor G_w, which generates feature vectors

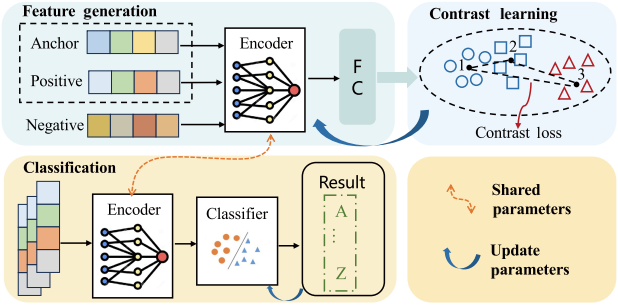


Fig. 9 Contrast learning network.

y_1 and y_2. We then compute the Euclidean distance D_w[60] between the two feature vectors as

$$D_w(x_1,\ x_2) = \|G_w(x_1) - G_w(x_2)\|_2 \tag{2}$$

where w is the weights of the feature extractor model. For simplicity, we refer to $D_w(x_1,\ x_2)$ as D_w. The loss function is defined as

$$\mathcal{L}(w,\ a,\ x_1,\ x_2) = (1-a)\frac{1}{2}(D_w)^2 + (a)\frac{1}{2}\{\max(0,\ m - D_w)\} \tag{3}$$

When the two KiVi signals correspond to the same key ($a = 0$), the loss function minimizes the distance between their feature representations, encouraging the feature extractor to bring similar inputs closer together. Conversely, if the signals correspond to different keys ($a = 1$), the loss function becomes zero if the distance exceeds the margin m, indicating that the model has successfully differentiated the signals. If the distance is smaller than m, the loss encourages the feature extractor to increase the separation between different keys.

In summary, the feature extractor minimizes the distance between identical samples and maximizes the distance between different ones, allowing the model to learn meaningful feature representations. This alignment of similar inputs in the feature space improves generalization, making the extracted features more abstract and universally applicable. These features are then used to train a linear classifier for keystroke classification.

6.5 Cross-App inference

After training on a large dataset of KiVi signals corresponding to each character (which encompasses various user behaviors and external factors) within a malicious App, the App can operate in the background, silently intercepting the user's keystrokes in other foreground applications. By leveraging the previously trained supervised learning model, the malicious App is able to perform cross-App keystroke inference. Specifically, both the malicious App and the target App run on the same device, and the malicious app extracts relevant features from the keystroke signals in the foreground App to infer the user's input.

6.5.1 Background signal collection and feature extraction

The malicious App exploits the device's built-in accelerometer to continuously capture keystroke signals while the user interacts with the target

application. Each keystroke generates a corresponding KiVi signal, reflecting mechanical vibration patterns associated with key presses. These signals are shaped by user-specific factors (e.g., typing style, speed, and finger placement) and external influences (such as phone handling and ambient noise).

After collecting the signals, the malicious App leverages a pretrained feature extractor to process the raw KiVi data, extracting pertinent features from each keystroke. This enables the App to filter out irrelevant noise and focus on the essential information needed for subsequent inference tasks.

6.5.2 Real-time keystroke inference

Once valid KiVi signals are collected and their features extracted, the malicious App applies the previously trained supervised learning model to perform real-time inference. By analyzing the extracted features, the App can accurately infer the user's keystroke inputs within the foreground application.

Since the model was trained using a diverse range of user behavior patterns (including typing strength, speed, and finger dynamics), the App is capable of making accurate inferences even when the signals originate from different applications. By recognizing feature patterns within the input signals and mapping them to the corresponding keys, the App successfully achieves cross-App keystroke inference.

6.5.3 Sensitive information inference

Once the malicious App successfully performs real-time keystroke inference, it can further utilize these inferences to deduce sensitive information such as passwords, PINs, or account credentials. By continuously analyzing the inferred keystrokes, the App identifies patterns in the input sequence, enabling inference of potential sensitive information.

For example, in the case of password entry, the malicious App can use the sequence of inferred keystrokes to approximate the correct password, based on the length of the input, the type of characters (e.g., alphanumeric and symbols), and any observed keystroke timing patterns. Since the App does not require the exact sequence of inputs but only needs to narrow down the candidate input space, it can generate multiple candidate predictions based on the learned model increasing the likelihood of successfully inferring sensitive information.

Additionally, the malicious App can exploit contextual clues such as commonly used password patterns, user behavior patterns, and application-specific input conventions to further improve inference accuracy. This multi-layered approach enables the malicious App to accurately infer sensitive information even when faced with incomplete or noisy signal data.

6.5.4 Continuous eavesdropping and concealment

A key advantage of the malicious App is its ability to continuously run in the background without alerting the user. By exploiting the device's permission mechanisms, the App can collect accelerometer data without significantly affecting the user experience. This allows the App to monitor and collect keystroke signals over long periods of time, even when the user switches between different applications.

Moreover, the App can be designed to operate with minimal resource usage, reducing CPU and memory consumption to avoid detection. Its background operation can mimic that of regular Apps, making it difficult for users or security software to identify its presence.

7 Evaluation Result

In this section, we conduct a series of experiments to evaluate the performance of CaMi and analyze the effects of various influencing factors.

7.1 Experimental setup

Implementation. We developed and implemented CaMi as a standalone application on commercially available smartphones. The system leverages the built-in accelerometer and utilizes existing Application Programming Interfaces (APIs) to capture motion data and detect vibrations caused by finger taps. To ensure accurate and timely data collection, we configured the smartphone to operate at its maximum sampling rate by adjusting the API delay settings.

Network choice. For our contrastive learning framework, we employ DenseNet as the Convolutional Neural Network (CNN) architecture. Unlike traditional deep networks, such as ResNet, DenseNet introduces direct connections between each layer and all preceding layers, resulting in a total of $(L+1)L/2$ connections for an L-layer network. This enhanced connectivity allows DenseNet to achieve performance comparable to ResNet with fewer parameters. Additionally, the improved flow of information and gradients throughout the network mitigates the vanishing gradient problem, making DenseNet easier to train. In our inference tasks, DenseNet demonstrated superior accuracy with fewer parameters (6.98 M for

DenseNet vs. 27.97 M for ResNet) and lower computational costs compared to ResNet.

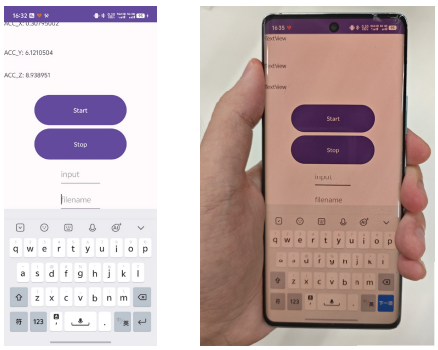
Experimental setup. Since CaMi does not require real-time processing, we decoupled KiVi signal acquisition from keystroke inference to reduce the smartphone's workload and minimize battery drain during the attack. The experiments were conducted on a vivo X100 as the primary device, with additional devices validated in the robustness analysis. An Android App simulating malicious behavior was employed to collect accelerometer signals in the background. This App also captures user input, enabling the collection of paired accelerometer–keystroke data for in-App training.

We recruited 10 participants (6 males and 4 females) aged between 20 and 30 with diverse natural typing habits. All participants were asked to hold the device and type as they normally would, covering a range of typing speeds and pressing strengths. This design ensures that the collected data reflect realistic user behavior rather than a single controlled typing style.

Participants first used the App to type each key 120 times over a week, generating the training set. The training data were primarily collected under a static hand-held posture to obtain high-quality labeled signals for model learning. Then, participants continued typing each key 80 times across different Apps while the malware App recorded accelerometer signals in the background, forming the testing set. The testing data cover multiple usage conditions, including both static use and more dynamic scenarios such as walking, which are further analyzed in the robustness evaluation. To ensure that the data were independently and evenly distributed, each key was collected three times, with an interval of at least 30 min between collections. The experimental setup is illustrated in Fig. 10.

All collected raw KiVi data were preprocessed as described in Section 6.1, including resampling at 500 Hz, normalization, and segmentation. The processed data were then used to train and evaluate the proposed CaMi model.

For training, we used categorical cross-entropy to compute the loss and the SGD optimizer to update the model parameters, resulting in a pre-trained model. The model was trained for 100 epochs with an initial learning rate of 0.005 and momentum of 0.8. Subsequently, the model was fine-tuned using contrastive learning, as introduced in Section 6.4, with a margin of 1. Performance was evaluated in terms of classification accuracy.



(a) App's interface (b) Handhold position

Fig. 10 Developed CaMi prototype, along with the experimental setup.

7.2 Performance of keystroke inference

In this section, we present a comprehensive analysis of the performance of the proposed CaMi system, evaluating it across multiple dimensions such as overall accuracy, per-key performance, and its ability to handle misidentifications using a novel metric, Around Accuracy (AR Acc).

Overall inference performance with different models. We compared the overall performance of CaMi with various state-of-the-art deep learning methods, as shown in Fig. 11. The results indicate that CaMi (DenseNet Contrast) outperforms all other models, achieving the highest accuracy. To assess model robustness, we evaluated performance across different environments and input behaviors, presenting the results with error bars for each model. As illustrated, the DenseNet-based contrastive learning approach not only achieves high accuracy but also demonstrates strong robustness against input variations and external factors, making it highly effective in real-world scenarios characterized by user behavior variability.

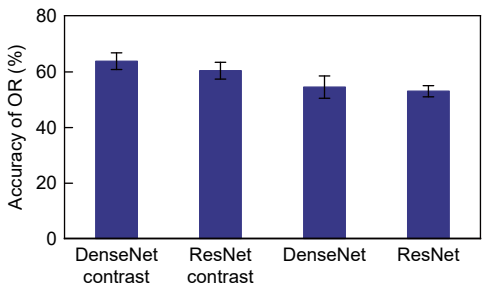


Fig. 11 Comparison of overall performance in keystroke inference across different deep learning models.

The superior performance of CaMi is largely due to its ability to model the complexity of keystroke dynamics using contrastive learning. This enables it to generalize effectively to new, unseen domains, highlighting the value of contrastive learning in overcoming the challenges posed by dynamic and highly variable user input patterns.

Per-key inference performance. Here, we explore the inference performance of individual keys to provide a more granular analysis of the system's accuracy. As shown in Fig. 12, the *x*-axis represents the individual keys,the accuracy for different keys fluctuated between 40% and 85%, with most keys achieving accuracy above 60%. While these fluctuations indicate variability in the system's performance, it is important to note that the majority of keys are still correctly identified with relatively high accuracy.

An interesting observation is that most of the incorrectly inferred keys are misidentified as keys in their immediate vicinity, as shown in the confusion matrix in Fig. 13. This pattern suggests that errors are likely to occur when the system encounters keys with similar mechanical characteristics or spatial proximity.

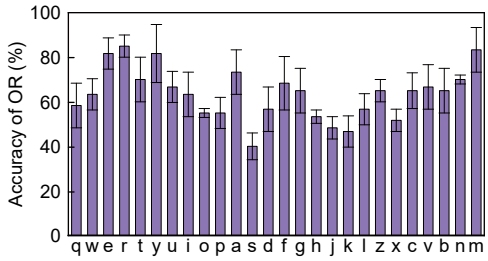


Fig. 12 Accuracy of per-key keystroke inference.

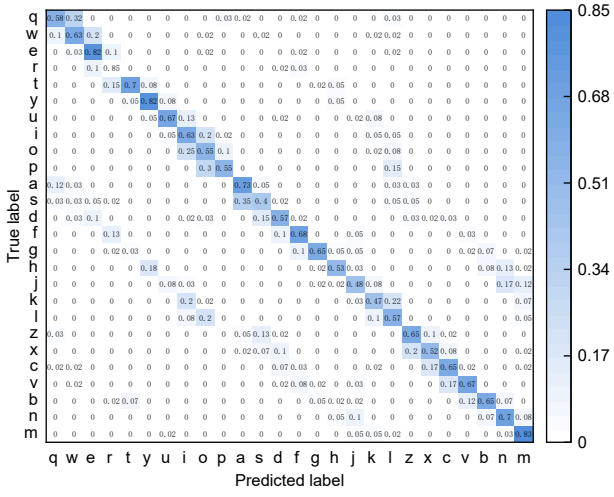


Fig. 13 Confusion matrix showing the accuracy of on-device keystroke detection.

While these misclassifications can be considered minor errors, they can be crucial in certain security-sensitive applications, such as password entry. Understanding and mitigating these misidentifications is essential for improving the system's overall robustness.

The fluctuations in accuracy between different keys also reflect the inherent difficulty in distinguishing between certain keys. For instance, keys with similar mechanical signatures (e.g., those located next to each other on the keyboard) may present more challenges for the model, leading to higher misclassification rates. This suggests the need for further optimization to handle subtle distinctions between keys more effectively.

AR Acc performance. To address the issue of misidentifications, particularly when keys are misclassified as neighboring keys, we introduce a novel metric called AR Acc. This metric considers misidentified keys as correctly classified if they fall within a specific neighborhood of the intended key. In other words, when a key is misidentified as a neighboring key, we treat this as a correct inference because such small errors can often be corrected in real-world scenarios using semantic context or multiple input attempts, such as when entering a password.

As shown in Fig. 14, the AR Acc metric achieves an impressive 94.81% accuracy. This substantial improvement over the Original Raw Accuracy (OR Acc) underscores the effectiveness of this approach in mitigating minor misclassifications. In practice, this metric is particularly useful because it accounts for the context in which keys are entered. In many cases, small errors are not critical, and being able to infer the correct input with a few attempts significantly increases the system's practical utility.

The high AR Acc value demonstrates that, while minor errors in individual key identification do occur, the system's ability to adjust based on semantic context or multiple attempts makes it highly effective in real-

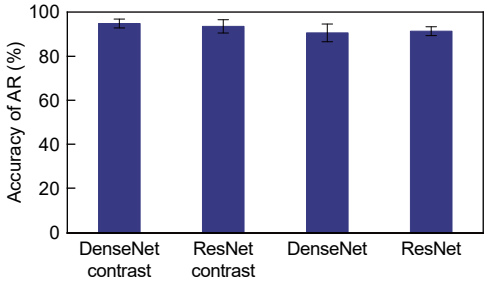


Fig. 14 Overall AR Acc of keystroke inference.

world scenarios, particularly for tasks like password entry or account authentication. This ability to tolerate small errors and still make accurate inferences is a key strength of the CaMi system.

7.3 Robustness of keystroke inference

In this section, we investigate the robustness of our keystroke inference model by evaluating its performance across various conditions, including environmental factors, long-term usage, and the effectiveness of contrastive learning. These factors are critical for assessing the model's ability to maintain accuracy and generalization over time and in diverse scenarios.

Impact of attack environments. To assess the generalization capability of our model in real-world conditions, we tested it in dynamic environments, specifically while walking. As shown in Fig. 15, although the training data were collected under controlled conditions with the phone in hand, our model demonstrates strong performance even in the walking scenario. In these conditions, the accuracy reaches 39.74%, and the AR Acc improves to 80%.

This result highlights two important aspects. First, the effectiveness of our denoising algorithm, which filters out motion-induced noise. Second, the model's inherent ability to generalize to environments not present in the training data. These findings suggest that our model can adapt to real-world variability, where users are often on the move and their input may be affected by factors such as walking or changing postures.

Long-term consistency of inference performance. Over extended periods, user behavior can evolve as people adjust to their devices. To evaluate the long-term stability of our model, we collected data at one-month intervals over three months, while the model continued to be trained and tested. As depicted in Fig. 16, although we observed fluctuations in the accuracy, these variations were within expected ranges. The diversity in user behavior across time contributed to this natural fluctuation.

Importantly, after a month, there was a noticeable improvement in accuracy, reinforcing the observation that the model's performance is not degrading over time. Rather, the fluctuations can be attributed to the inherent changes in user interaction patterns. These results demonstrate that the model remains effective even in the face of evolving user habits, further confirming its long-term robustness.

Impact of device type on performance. To assess device independence, we conducted cross-device experiments using smartphones from various brands and models. As shown in Fig. 17, the AR Acc exceeded 90% in all cases, demonstrating that our approach generalizes effectively across different mobile devices.

7.4 Impact of contrast learning on performance

In our previous experiments (Figs. 11 and 14), we observed significant improvements in accuracy when employing our contrastive learning model. To further validate its impact, we conducted an ablation study where we removed the contrastive learning component and compared the results.

The t-SNE visualizations of the features before and after applying contrastive learning are shown in Figs. 18a and 18b. These plots clearly demonstrate that,

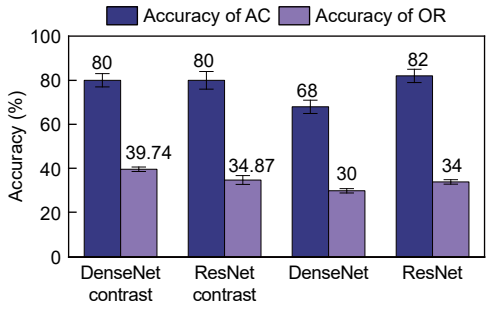


Fig. 15 Keystroke accuracy while walking.

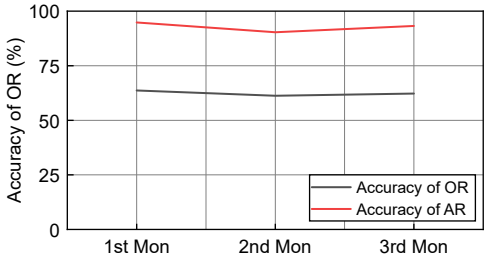


Fig. 16 Accuracy fluctuations over a three-month period.

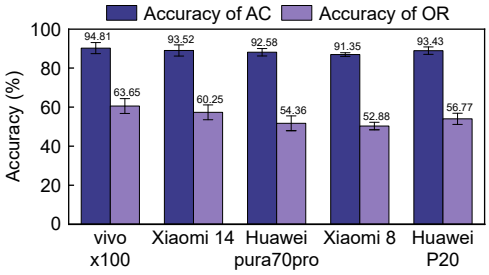
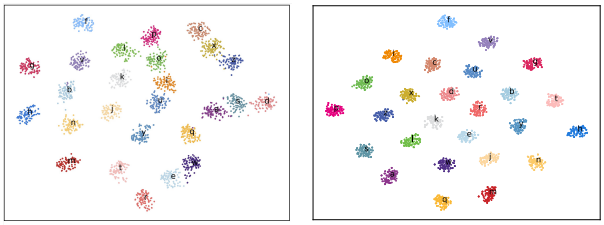


Fig. 17 Keystroke accuracy using different devices.



(a) Without contrast learning (b) With contrast learning

Fig. 18 Contrastive learning presents t-SNE results without and with its application.

with contrastive learning, the feature distributions of different keys are more distinct, indicating that the model can better differentiate between keystrokes. This enhanced separation improves the model's overall performance, especially in challenging real-world scenarios where subtle variations in keystroke dynamics can be difficult to capture.

In summary, contrastive learning not only boosts the model's accuracy but also enhances its ability to generalize across different environments, user behaviors, and input variations.

8 Defense

In this section, we explore potential defense strategies against the attack proposed in this work.

Access control for accelerometer permissions. A fundamental defense against malicious Apps like CaMi lies in controlling access to device sensors, particularly the accelerometer. Restricting or managing permissions for accelerometer access is crucial. Users should have the option to grant or deny permission for Apps to collect accelerometer data, especially at high sampling rates, which are often used in inference attacks. Implementing more stringent controls, such as prompting users for explicit permission to record accelerometer data at high frequencies, would mitigate the risk of unauthorized data collection.

Shuffled keyboards. Another potential defense against keystroke inference based on vibrations is the use of dynamically shuffled keyboard layouts. By randomly reordering the keys each time the user inputs credentials, this approach makes it significantly more challenging for attackers to map vibration signals to specific keystrokes. Even if the attacker successfully collects keystroke-induced vibration signals, without knowledge of the keyboard layout, it becomes nearly impossible to infer the actual input.

While this defense strategy substantially hinders keystroke inference, it introduces an additional cognitive load on the user. Users would need to locate keys in a new layout each time they input sensitive information, which could lead to increased search time and a higher likelihood of input errors. Furthermore, the inconvenience introduced by constantly shuffled layouts might deter users from using this defense in practice, highlighting a trade-off between security and user experience.

Shuffled keyboards can significantly reduce the effectiveness of keystroke inference attacks; however, the trade-off between security and usability must be carefully considered.

9 Discussion

This section summarizes the main limitations of our work and outlines directions for improvement.

9.1 Model performance and fine-grained accuracy

Although the overall around accuracy of our model reaches about 90%, the top-1 accuracy remains around 60%, showing that the model still struggles with fine-grained distinctions between similar keystrokes. Improving this performance will require more diverse training data and further optimization of the model architecture and feature extraction.

9.2 Adaptability to dynamic environments

Our model performs well in typical usage scenarios but its accuracy drops under sudden or vigorous movements, which commonly occur in real-world settings. Enhancing robustness in such dynamic environments may involve incorporating additional motion cues and more advanced denoising or multi-sensor fusion techniques.

9.3 Data diversity and long-term stability

User typing behaviors evolve over time, which can lead to natural fluctuations in accuracy and affect long-term stability. Adaptive learning techniques and longitudinal data collection could help the model continuously adjust to changing user patterns; as discussed in Section 7, using multiple participants with diverse, natural typing habits is a first step in this direction.

9.4 Impact of keyboard variations and feedback

Our study focuses on a standard QWERTY virtual keyboard with a fixed layout and key-click haptic and audio feedback disabled, so that KiVi signals mainly originate from fingertip impacts on the touchscreen.

When such feedback is enabled, additional vibration or acoustic components may alter the signal characteristics and interfere with our current pipeline, and extending the attack to these settings will require more advanced signal separation or denoising.

9.5 System-level limitations

Modern mobile operating systems restrict background sensor access, power consumption, and App lifecycles, which can hinder continuous high-rate KiVi collection. An attacker may disguise a malicious App as a legitimate foreground service (e.g., a fitness or navigation App) to obtain long-term sensor access, our work mainly illustrates an upper-bound threat under permissive conditions and highlights the importance of stronger system-level defenses and stricter sensor policies.

9.6 Trade-offs in user experience and security measures

Defense mechanisms such as dynamic keyboard layouts can mitigate keystroke inference attacks but may also increase users' cognitive load and harm usability. Future work should explore user-centric defenses, such as context-aware or adaptive keyboards, that provide stronger protection with minimal disruption to normal usage.

9.7 Future research directions

Future work will focus on improving fine-grained classification, enhancing robustness to motion and device heterogeneity, enabling long-term adaptability to evolving user behaviors, and designing practical, user-friendly defense mechanisms. Addressing these challenges will help make keystroke inference models both more accurate and more reflective of real-world deployment conditions.

10 Conclusion

This paper introduces a novel class of security threats in the form of cross-App keystroke inference attacks, which may represent an underexplored vulnerability. We present CaMi, a deep learning-based model that leverages "zero-permission" motion sensors to infer keystrokes, circumventing traditional permission-based restrictions. The model begins by applying a series of pre-processing techniques to raw acceleration data, followed by the use of a contrastive learning network to extract distinguishing features from the signals. In particular, we utilize two CNNs with shared parameters, where the loss function aims to minimize the distance between features of identical keystrokes while maximizing the distance between those of different keystrokes. These features are then classified through a fully connected network. Experimental results reveal that CaMi achieves strong performance in recognizing 26 English letters, with an overall accuracy of 94.81% and a top-1 accuracy of 64%. This work demonstrates the feasibility of cross-app keystroke inference, offering a new perspective on mobile security threats.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Application domain, Available: https://en.wikipedia.org/wiki/Application domain, 2023.

[2] Application sandbox of android, https://source.android.com/docs/security/app-san dbox, 2024.

[3] Application sandbox of iOS, https://support.apple.com/zh-cn/guide/security/sec15bfe098e/web, 2021.

[4] Z. Xu, K. Bai, and S. Zhu, TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors, in *Proc. Fifth ACM Conf. Security and Privacy in Wireless and Mobile Networks*, Tucson, Arizona, USA, 2012, pp. 113–124.

[5] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, Tapprints: Your finger taps have fingerprints, in *Proc. 10th Int. Conf. Mobile Systems, Applications, and Services*, Low Wood Bay Lake District, UK, 2012, pp. 323–336.

[6] Android input, https://developer.android.com/develop/ui/views/touch-and-input/creating-input-method, 2024.

[7] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, Supervised contrastive learning, *Adv. neural inf. process. syst.*, vol. 33, pp. 18661–18673, 2020.

[8] L. Cai and H. Chen, TouchLogger: Inferring keystrokes on touch screen from smartphone motion, *HotSec*, vol. 11, p. 9, 2011.

[9] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, ACCessory: Password inference using accelerometers on smartphones, in *Proc. Twelfth Workshop on Mobile Computing Systems & Applications - HotMobile '12*, San Diego, CA, USA, 2012, pp. 1–6.

[10] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, Inferring user routes and locations using zero-permission mobile sensors, in *Proc. IEEE Symp. on Security and Privacy* (*SP*), San Jose, CA, USA, 2016, pp. 397–413.

[11] A. Das, N. Borisov, and M. Caesar, Do you hear what I

hear? : Fingerprinting smart devices through embedded acoustic components, in *Proc. 2014 ACM SIGSAC Conf. Computer and Communications Security*, Scottsdale, AZ, USA, 2014, pp. 441–452.

[12] K. Joo, W. Choi, and D. H. Lee, Hold the door! Fingerprinting your car key to prevent keyless entry car theft, https://arxiv.org/abs/2003.13251, 2020.

[13] T. Yu, H. Jin, and K. Nahrstedt, WritingHacker: Audio based eavesdropping of handwriting *via* mobile devices, in *Proc. 2016 ACM Int. Joint Conf. Pervasive and Ubiquitous Computing*, Heidelberg, Germany, 2016, pp. 463–473.

[14] A. Hojjati, A. Adhikari, K. Struckmann, E. Chou, T. N. Tho Nguyen, K. Madan, M. S. Winslett, C. A. Gunter, and W. P. King, Leave your phone at the door: Side channels that reveal factory floor secrets, in *Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security*, Vienna, Austria, 2016, pp. 883–894.

[15] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3D printers, in *Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security*, Vienna, Austria, 2016, pp. 895–907.

[16] W. Chen, L. Chen, M. Ma, F. S. Parizi, S. Patel, and J. Stankovic, ViFin: Harness passive vibration to continuous micro finger writing with a commodity smartwatch, *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 1, pp. 1–25, 2021.

[17] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, Beware, your hands reveal your secrets!, in *Proc. 2014 ACM SIGSAC Conf. Computer and Communications Security*, Scottsdale, AZ, USA, 2014, pp. 904–917.

[18] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, Seeing double: Reconstructing obscured typed input from repeated compromising reflections, in *Proc. 2013 ACM SIGSAC Conf. Computer & Communications Security - CCS '13*, Berlin, Germany, 2013, pp. 1063–1074.

[19] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, EyeTell: Video-assisted touchscreen keystroke inference from eye movements, in *Proc. IEEE Symp. on Security and Privacy* (*SP*), San Francisco, CA, USA, 2018, pp. 144–160.

[20] Y. Wang, W. Cai, T. Gu, and W. Shao, Your eyes reveal your secrets: An eye movement based password inference on smartphone, *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2714–2730, 2020.

[21] Y. Wang, W. Cai, T. Gu, W. Shao, I. Khalil, and X. Xu, GazeRevealer: Inferring password using smartphone front camera, in *Proc. 15th EAI Int. Conf. Mobile and Ubiquitous Systems: Computing, Networking and Services*, New York, NY, USA, 2018, pp. 254–263.

[22] J. Sun, X. Jin, Y. Chen, J. Zhang, R. Zhang, and Y. Zhang, VISIBLE: Video-assisted keystroke inference from tablet backside motion, in *Proceedings 2016 Network and Distributed System Security Symp.*, San Diego, CA, USA, 2016.

[23] M. Backes, T. Chen, M. Duermuth, H. P. A. Lensch, and M. Welk, Tempest in a teapot: Compromising reflections revisited, in *Proc. 30th IEEE Symp. on Security and Privacy*, Oakland, CA, USA, 2009, pp. 315–327.

[24] M. Backes, M. Dürmuth, and D. Unruh, Compromising reflections—or—how to read LCD monitors around the corner, in *2008 IEEE Symposium on Security and Privacy* (*SP 2008*), Oakland, CA, USA, 2008, pp. 158–169.

[25] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J. M. Frahm, iSpy: Automatic reconstruction of typed input from compromising reflections, in *Proc. 18th ACM Conf. Computer and Communications Security*, Chicago, IL, USA, 2011, pp. 527–536.

[26] H. Wang, Z. Zhan, H. Shan, S. Dai, M. Panoff, and S. Wang, GAZEploit: Remote keystroke inference attack by gaze estimation from avatar views in VR/MR devices, in *Proc. 2024 on ACM SIGSAC Conf. Computer and Communications Security*, Salt Lake City, UT, USA, 2024, pp. 1731–1745.

[27] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, Keystroke recognition using WiFi signals, in *Proc. 21st Annual Int. Conf. Mobile Computing and Networking*, Paris, France, 2015, pp. 90–102.

[28] S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu, No training hurdles: Fast training-agnostic attacks to infer your typing, in *Proc. 2018 ACM SIGSAC Conf. Computer and Communications Security*, Toronto, Canada, 2018, pp. 1747–1760.

[29] E. Yang, Q. He, and S. Fang, WINK: Wireless inference of numerical keystrokes *via* zero-training spatiotemporal analysis, in *Proc. 2022 ACM SIGSAC Conf. Computer and Communications Security*, Los Angeles, CA, USA, 2022, pp. 3033–3047.

[30] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan, When CSI meets public WiFi: Inferring your mobile phone password *via* WiFi signals, in *Proc. 2016 ACM SIGSAC Conf. Computer and Communications Security*, Vienna, Austria, 2016, pp. 1068–1079.

[31] J. Zhang, X. Zheng, Z. Tang, T. Xing, X. Chen, D. Fang, R. Li, X. Gong, and F. Chen, Privacy leakage in mobile sensing: Your unlock passwords can be leaked through wireless hotspot functionality, *Mob. Inf. Syst.*, vol. 2016, p. 8793025, 2016.

[32] K. Ling, Y. Liu, K. Sun, W. Wang, L. Xie, and Q. Gu, SpiderMon: Towards using cell towers as illuminating sources for keystroke monitoring, in *Proc. IEEE INFOCOM 2020 - IEEE Conf. Computer Communications*, Toronto, Canada, 2020, pp. 666–675.

[33] H. Wang, J. Hu, T. Zheng, J. Hu, Z. Chen, H. Jiang, Y. Zheng, and J. Luo, MuKI-Fi: Multi-person keystroke inference with BFI-enabled Wi-Fi sensing, *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9835–9850, 2024.

[34] Y. L. Du, Y. H. Lu, and J. L. Zhang, Novel method to detect and recover the keystrokes of PS/2 keyboard, *Prog. Electromagn. Res. C*, vol. 41, pp. 151–161, 2013.

[35] M. Vuagnoux and S. Pasini, Compromising electromagnetic emanations of wired and wireless

keyboards, in *USENIX security symposium*, Montreal, Canada, 2009, pp. 1–16.

[36] L. Wang and B. Yu, Analysis and measurement on the electromagnetic compromising emanations of computer keyboards, in *Proc. Seventh Int. Conf. Computational Intelligence and Security*, Sanya, China, 2011, pp. 640–643.

[37] W. Jin, S. Murali, H. Zhu, and M. Li, Periscope: A keystroke inference attack using human coupled electromagnetic emanations, in *Proc. 2021 ACM SIGSAC Conf. Computer and Communications Security*, New York, NY, USA, 2021, pp. 700–714.

[38] D. Genkin, A. Shamir, and E. Tromer, RSA key extraction *via* low-bandwidth acoustic cryptanalysis, in *Advances in Cryptology – CRYPTO*, J. A. Garay, R. Gennaro, Eds. Berlin, Heidelberg: Springer, 2014, pp. 444–461

[39] D. Genkin, A. Shamir, and E. Tromer, Acoustic cryptanalysis, *J. Cryptol.*, vol. 30, no. 2, pp. 392–443, 2017.

[40] Y. Berger, A. Wool, and A. Yeredor, Dictionary attacks using keyboard acoustic emanations, in *Proc. 13th ACM Conf. Computer and Communications Security*, Alexandria, VA, USA, 2006, pp. 245–254.

[41] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser, Snooping keystrokes with mm-level audio ranging on a single phone, in *Proc. 21st Annual Int. Conf. Mobile Computing and Networking*, Paris, France, 2015, pp. 142–154.

[42] L. Zhuang, F. Zhou, and J. D. Tygar, Keyboard acoustic emanations revisited, *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, pp. 1–26, 2009.

[43] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, Context-free attacks using keyboard acoustic emanations, in *Proc. 2014 ACM SIGSAC Conf. Computer and Communications Security*, Scottsdale, AZ, USA, 2014, pp. 453–464.

[44] L. Lu, J. Yu, Y. Chen, Y. Zhu, X. Xu, G. Xue, and M. Li, KeyListener: Inferring keystrokes on QWERTY keyboard of touch screen through acoustic signals, in *Proc. IEEE INFOCOM 2019 - IEEE Conf. Computer Communications*, Paris, France, 2019, pp. 775–783.

[45] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, Friend or foe? : Your wearable devices reveal your personal PIN, in *Proc. 11th ACM on Asia Conf. Computer and Communications Security*, Xi'an, China, 2016, pp. 189–200.

[46] Y. Liu and Z. Li, aLeak: Privacy leakage through context - free wearable side-channel, in *Proc. IEEE INFOCOM 2018 - IEEE Conf. Computer Communications*, Honolulu, HI, USA, 2018, pp. 1232–1240.

[47] H. Wang, T. T. Lai, and R. Roy Choudhury, MoLe: Motion leaks through smartwatch sensors, in *Proc. 21st Annual Int. Conf. Mobile Computing and Networking*, Paris, France, 2015, pp. 155–166.

[48] G. Ye, Z. Tang, D. Fang, X. Chen, K. I. Kim, B. Taylor, and Z. Wang, Cracking Android pattern lock in five attempts, in *Proceedings 2017 Network and Distributed System Security Symp.*, San Diego, CA, USA, 2017.

[49] Y. Feng, D. Liu, W. Jin, and L. Gong, KeyPrint: Practical black-box keystroke inference attacks to mobile devices, *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 9, no. 2, pp. 1–30, 2025.

[50] S. Narain, A. Sanatinia, and G. Noubir, Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning, in *Proc. 2014 ACM Conf. Security and Privacy in Wireless & Mobile Networks*, Oxford, UK, 2014, pp. 201–212.

[51] I. Shumailov, L. Simon, J. Yan, and R. Anderson, Hearing your touch: A new acoustic side channel on smartphones, https://arxiv.org/abs/1903.11137, 2019.

[52] H. Kim, B. Joe, and Y. Liu, TapSnoop: Leveraging tap sounds to infer tapstrokes on touchscreen devices, *IEEE Access*, vol. 8, pp. 14737–14748, 2020.

[53] Z. Xiao, T. Chen, Y. Liu, and Z. Li, Mobile phones know your keystrokes through the sounds from finger's tapping on the screen, in *Proc. IEEE 40th Int. Conf. Distributed Computing Systems* (*ICDCS*), Singapore, 2020, pp. 965–975.

[54] D. Liu, H. Lu, X. Qi, H. Rong, H. Chen, and H. Jiang, Decoding air friction rhythms: Enabling user identification with out-ear microphones in COTS earphones, *IEEE Trans. Mobile Comput.*, vol. 25, no. 4, pp. 4648–4663, 2026.

[55] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, When good becomes evil: Keystroke inference with smartwatch, in *Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security*, Denver, CO, USA, 2015, pp. 1273–1285.

[56] A. Maiti, O. Armbruster, M. Jadliwala, and J. He, Smartwatch-based keystroke inference attacks and context-aware protection mechanisms, in *Proc. 11th ACM on Asia Conf. Computer and Communications Security*, Xi'an, China, 2016, pp. 795–806.

[57] P. Marquardt, A. Verma, H. Carter, and P. Traynor, (sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers, in *Proc. 18th ACM Conf. Computer and Communications Security*, Chicago, IL, USA, 2011, pp. 551–562.

[58] S. McAdams, Perspectives on the contribution of timbre to musical structure, *Comput. Music. J.*, vol. 23, no. 3, pp. 85–102, 1999.

[59] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, A simple framework for contrastive learning of visual representations, in *Proc. 37th Int. Conf. Mach. Learn.* (*ICML*), 2020, pp. 1597–1607.

[60] J. Tabak, Geometry: The Language of Space and Form, Facts on File Math Library. New York: Facts On File, 2014.



Ruyi Wei is currently a third-year MS student in College of Computer Science and Electronic Engineering at Hunan University, majoring in electronic information (computer technology). His research interests include mobile computing and security.



Junhua Wu received the BS degree in information and computing science from Xiangtan University, in 2000, and the MS degree in law from Hunan University, in 2014. She is currently a senior engineer with the Public Security Department of Hunan Province. Her major research area is network and cyberspace security.



Huigui Rong received the PhD degree in information management from Wuhan University, in 2010. He is currently a full associate professor of computer science and technology at Hunan University and was a visiting scholar in computer science department of Michigan State University (MSU) from 2014 to 2015. His major research areas include Internet of Things and edge computing. He has published more than 30 research papers in international conferences and journals, such as *IEEE IoT-J*, *IEEE-Transactions on Big Data (TBD)*, and Association for Computing Machinery (ACM)- International Conference on Information and Knowledge Management (CIKM). He is a member of IEEE, ACM, and China Computer Federation (CCF).



Daibo Liu received the PhD degree in computer science and engineering from the University of Electronic Science and Technology of China, in 2018. He is currently an associate professor in the College of Computer Science and Electronic Engineering, Hunan University. His research interests cover the broad areas of low-power wireless networks, mobile and pervasive computing, and system security. He is a member of the IEEE and ACM.