Friend or Foe? Your Wearable Devices Reveal Your Personal PIN

Chen Wang
Department of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
cwang42@stevens.edu

Xiaonan Guo
Department of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
xguo6@stevens.edu

Yan Wang
Department of CS
Binghamton University
Binghamton, NY, USA
yanwang@binghamton.edu

Yingying Chen *
Department of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
yingying.chen@stevens.edu

Bo Liu
Department of ECE
Stevens Institute of Technology
Hoboken, NJ, USA
bliu11@stevens.edu

ABSTRACT

The proliferation of wearable devices, e.g., smartwatches and activity trackers, with embedded sensors has already shown its great potential on monitoring and inferring human daily activities. This paper reveals a serious security breach of wearable devices in the context of divulging secret information (i.e., key entries) while people accessing key-based security systems. Existing methods of obtaining such secret information relies on installations of dedicated hardware (e.g., video camera or fake keypad), or training with labeled data from body sensors, which restrict use cases in practical adversary scenarios. In this work, we show that a wearable device can be exploited to discriminate mm-level distances and directions of the user's fine-grained hand movements, which enable attackers to reproduce the trajectories of the user's hand and further to recover the secret key entries. In particular, our system confirms the possibility of using embedded sensors in wearable devices, i.e., accelerometers, gyroscopes, and magnetometers, to derive the moving distance of the user's hand between consecutive key entries regardless of the pose of the hand. Our Backward PIN-Sequence Inference algorithm exploits the inherent physical constraints between key entries to infer the complete user key entry sequence. Extensive experiments are conducted with over 5000 key entry traces collected from 20 adults for key-based security systems (i.e. ATM keypads and regular keyboards) through testing on different kinds of wearables. Results demonstrate that such a technique can achieve 80% accuracy with only one try and more than 90% accuracy with three tries, which to our knowledge, is the first technique that reveals personal PINs leveraging wearable devices without the need for labeled training data and contextual information.

*Yingying Chen is the corresponding author.

ASIA CCS '16, May 30-June 03, 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2897845.2897847

Keywords

Privacy leakage; Wearable devices; Leakage of PIN; hand movement trajectory recovery; PIN sequence inference

1. INTRODUCTION

The convenience of wearable devices, such as smartwatches and fitness bands (e.g., Fitbit and Jawbone), has greatly stimulated the growth of the market of mobile devices in recent years; market researchers estimated that 72.1 million wearable devices will be shipped in 2015, which will be about 173% from the 26.4 million wearable devices shipped in 2014 [4]. Such increasing popularity of wearable devices has enabled a broad range of useful applications, including fitness tracking, falling detection, gesture control and user authentication. Since such wearable devices have the ability to capture users' hand movements and derive human dynamics directly, a major concern arises on whether a user's sensitive information could be leaked and obtained by adversaries including the user's PIN sequence when accessing an ATM machine or using debit cards for payment.

In this work, we demonstrate that a user's personal PIN sequence could be leaked through his wearable devices (e.g, smartwatch or fitness tracker), when accessing a key-based security system. Such systems are very common in daily lives. Examples include accessing ATM cash machines, electronic door locks, and keypad-controlled enterprise servers. A key-based security system requires people to enter personal key combinations on the keypad for identity verification. With people tending to wear wearable devices around-the-clock, the movements of their wrists during the key entry process to a security system (i.e., clicking keys and moving between clicks) are captured by the sensors on wearable devices. As such, wearables could cause a new way of sensitive information leakage when a user accesses the key-based security systems. In particular, adversaries can obtain sensor readings of wearables via sniffing Bluetooth communications [16, 19] or installing malwares [3] on the devices, and further infer the user's PIN sequence (e.g., ATM PIN sequences or key sequences on access control panels) for his own use.

There has been active study on sensitive information leakage when using key-based security systems. Traditional attacks rely on either shoulder surfing or hidden cameras [6, 11]. Such attacks require direct visual contact to key entry actions and additional instal-

lation efforts. Furthermore, Shukla *et al.* propose a side-channel attack utilizing a camera-based method to recover smartphone lock PINs from the user's spatial-temporal hand dynamics without directly seeing the keypad on screen [18]. The proposed method has a low inference accuracy and requires cameras to capture the user's hand and the back side of the touch screen. Two recent work [10, 20] propose to utilize sensors in smartwatches to infer user's typed words or passwords. The MoLe [20] system relies on a linguistic model to infer user's typed words, which is difficult to work with non-contextual inputs. Liu *et al.* [10] devise a system that requires training of the sensor data to classify user inputs.

In contrast to these prior studies, we develop a training-free, context-free technique to reveal a user's private PIN sequence (to a key-based security system) when a wrist-worn wearable device is employed. The wrist-worn wearable devices could be either smartwatches or fitness trackers. While the digital smartwatch is designed to be worn on either hand, the user can wear it on the right hand without the concern on traditional watch designed to adjust time easily when wearing it on the left hand. Additionally, many people tend to wear fitness tracker on the right hand while keeping wearing traditional watch on the left hand. The basic idea is to exploit embedded sensors in wearable devices to capture dynamics of key entry activities and derive fine-grained hand movement trajectories traversing secret key entries. While wearable devices have equipped with various sensors, it is challenging to accurately recover such fine-grained hand-movement trajectories that exhibit only mm-level difference in distance between keys via low-fidelity sensors. In addition, due to hand vibrations and rotations, the coordinate system of a wearable device is not always aligned with a fixed reference, which makes it hard to track the hand movements by using sensor readings directly. Additionally, in order to obtain a person's key entries without user cooperation or drawing any attention, the adversary has to achieve the PIN sequence with no training or contextual information.

To address these challenges, our approach examines the inherent physics phenomenon extracted from the user's key entry activities via wearable sensors and develops distance calculation and direction derivation schemes to produce mm-level accuracy when estimating the moving distance and angle between two consecutive key entries. To obtain the complete PIN sequence, our backward PIN-sequence inference algorithm exploits the physical constraints of distance between keys and temporal sequence of key entry activities to construct a tree of candidate key entries for determining the PIN sequence in a reversed manner, because in many practical cases, the "Enter" key is the last key after the user enters his/her PIN sequence. The mm-level precision of estimating the fine-grained moving distance and direction between two keys and the backward PIN-sequence inference algorithm enable our system to obtain the user's PIN sequence without training and contextual information. Such a technique can also be extended to support password recovery when people type on keyboards while wearing wearables.

We summarize our main contributions as follows:

- We demonstrate that a single wrist-worn wearable device can reveal a user's PIN sequence to key-based security systems. We develop a training-free approach by exploiting the inherent physics meaning extracted from sensor readings on wearables. Such an approach does not require contextual information, allowing it to recover random key entries.

- We develop the distance estimation and direction derivation schemes that capture the fine-grained hand movements at mm-level precision.

- We show that it is possible to infer a complete user's PIN number via a backward PIN-sequence inference algorithm. By exploiting spatial and temporal constraints of PIN entries and the fine-grained hand movement analysis, our approach can accurately pin-point the location of each PIN entry with the right sequence.

- We conduct extensive experiments with 20 participants wearing two types of smartwatch and a prototype of wearable on key-based security systems such as ATM keypads and keyboards over an eleven-month period. We show that our system can achieve 80% accuracy of inferring PIN sequences with only one try and over 90% accuracy with three tries without training and contextual information.

The rest of the paper is organized as follows. We first put our work in the context of related studies in Section 2. In section 3, we investigate the feasibility of using wearables to obtain a user's PIN sequence of key-based services. We then describe the design of our PIN-sequence inference framework in Section 4. Next, we present two schemes of distance estimation and direction derivation to capture fine-grained hand movements via sensors on wearables in Section 5. The backward PIN-sequence inference algorithm to recover the complete user PIN sequence is described in Section 6. We present the detailed implementation of our framework in terms of pre-processing of the sensor data and coordinate alignment in Section 7. In Section 8, we perform extensive evaluation of our approach involving real key-based security systems. Finally, we discuss the relative issues and conclude our work in Sections 9 and 10 respectively.

2. RELATED WORK

Recent studies show that embedded sensors on mobile devices, such as accelerometers and touch screens, can capture users' motion and leak their sensitive information [13, 15, 17]. Recently, wearable devices, such as smartwatches and fitness bands, extend the sensing capability to limbs and enable many useful applications [9, 14, 22]. These existing studies have shown the sensing capabilities of up-to-date mobile devices, which inspire us to explore the potential of using wrist-mounted wearables to recover fine-grained hand movements, and study to what extent the user's sensitive information could be leaked from their fingers.

Toward this end, we explore the possibility of recovering people's private PIN sequences through their wrist-worn mobile devices when they enter PINs on key-based security systems. Traditionally, key-based security systems could be breached by several methods, such as hidden cameras and skimmers. For example, some ATM machines are attached by a hidden camera, which was used to record PIN sequences or body movements of entering PINs [11]. An adversary may also put a skimmer into the ATM machine card slot. When the customer slides their card, it will go through the skimmer first and then into the machine. A chip inside the skimmer device records information about the account without the knowledge of the customer [1]. These existing methods largely depend on installing dedicated devices in the restricted area.

In addition, researchers show that it is possible to recognize users' keystrokes by using acoustic approaches. Berger *et al.* [7] demonstrate that by using linguistic models and recorded typing sound on a keyboard, an attacker can successfully reconstruct the typed words. Zhu *et al.* [23] present a context-free and geometry-based approach to recover keystrokes by using multiple smartphones to record acoustic emanations from the keystrokes. Wang *et al.* [21] develop a system that extracts and optimizes the location-dependent multipath fading features from the audio signals and leverages the signal diversity resulted from the dual-microphone interface in a

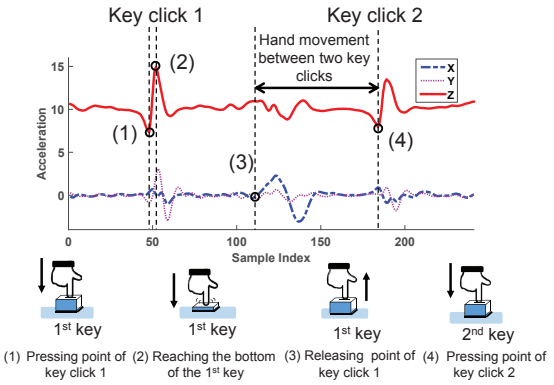
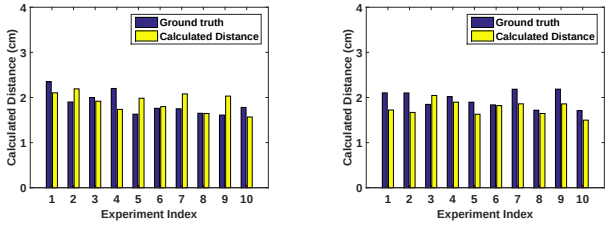


Figure 1: Acceleration patterns inherited from key entry activities, shown in the readings of a 3-axis accelerometer on IMU.



(a) Distance estimation between keys 4 and 5.

(b) Distance estimation between keys 8 and 5.

Figure 2: Distance estimation of the number pad on the Dell keyboard based on IMU.

mobile device to identify key entries typed on a keyboard. Along this line, Jian *et al.* [8] demonstrate that mobile audio hardware in off-the-shelf mobile devices can be exploited to discriminate mm-level position differences, based on which they develop a system that can locate the origin of keystrokes by using only a single phone behind a keyboard [8]. Marquardt *et al.* develop an application that can utilize accelerometers in a smartphone to sense the vibrations caused by keystrokes from a nearby keyboard and further identify the keystrokes [12]. Their proposed technique relies on a linguistic model and labeled training data and the system is highly sensitive to environment noise (e.g., people moving around).

The most related work to ours are two concurrent studies, which analyze the leak of users' passwords or typed words from smartwatches [10, 20]. Wang *et al.* [20] devise a system that can infer typed words on a keyboard by utilizing motion sensors in smartwatches. The system assumes to know the fixed initial position of the smartwatch and relies on a linguistic model to infer typed words, which makes it hard to deal with non-contextual inputs, such as passwords and PIN sequences. Liu *et al.* [10] apply sensors in a smartwatch to infer users' inputs on a keyboard or POS terminal by utilizing machine-learning based techniques. Their approach requires training of hand movements between keystrokes, and it is unclear how the system handles changing positions of the wrist during typing. Moreover, both of the above work can only achieve moderate accuracy in deriving the user inputs given limited number of tries. Different from previous work, our key entry inference system is training-free, contextual-free and does not involve additional devices. Furthermore, our backward PIN-sequence inference framework is not subject to environmental noises, such as ambient noise, light interference and people walking around.

3. ATTACK MODEL AND FEASIBILITY STUDY

The positions of wearable devices on human bodies naturally enhance the devices' capability of the activity recognition and facilitate many applications based on the context of activities. However, such strong sensing ability brings up new security and privacy issues. In this work, we study the possible personal secret leakage in a very common scenario that people wear wrist-worn wearable devices while using key-based security systems, such as ATM machines, password secured door entries, and keypad-controlled enterprise servers. In this section, we describe the attack model and explore the feasibility of utilizing wearable devices to recover personal key entries in key-based security systems.

3.1 Attack Model

We consider an adversary aiming at recovering a person's secret PIN entries leveraging embedded sensors (e.g., accelerometer, gyroscope and magnetometer) in wearable devices worn on his/her wrist. The adversary has the knowledge of where the victim visits the key-based security system and can obtain the layout of the keypad. We assume that the adversary is able to access the sensor data and communicate over networks on the smartphone, but cannot observe the PIN entry activities visually by any means. The wearable device is usually paired with the user's smartphone via Bluetooth and constantly sends sensor data to the person's smartphone for logging purpose. Most wearables are using Bluetooth Low Energy (BLE) to transmit sensor data. BLE comes with low security capability compared with Bluetooth, and as a result the sensor data could be sniffed by the adversary [16, 19]. But the adversary does not have access to training data, which is specific to a particular key-based security system. Particularly, we identify two representative attacking scenarios as follows:

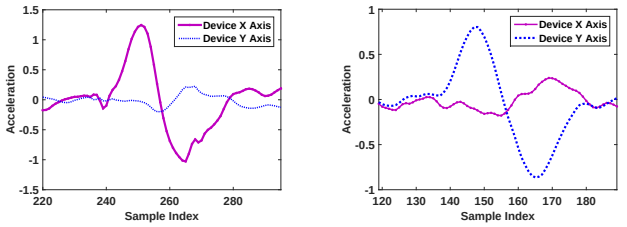
Sniffing Attacks. An adversary can place a wireless sniffer close to a key-based security system (e.g., ATM machine or key-based security door) to eavesdrop sensor data from the wearable device, which is worn on the victim's wrist when he/she enters security PINs into the security system. The adversary utilizes the wireless sniffer to capture Bluetooth packets sent by the wearable device to its associated smartphone [16, 19], and determines the victim's PIN sequence based on the sensor data extracted from Bluetooth packets.

Internal Attacks. An adversary can access the embedded sensors in the victim's wrist-worn wearable device by installing a malware app without the victim's notice [3]. The malware app waits until the victim accesses the key-based security system and keeps sending sensor data back to the adversary's server through the Internet. The adversary can aggregate the sensor data on the server to determine the victim's PIN sequence remotely.

3.2 Intuitions of Hand Movements behind Key Entry Activities

When accessing a key-based security system, a person's PIN sequence is entered through multiple key clicks. During each key click, there exhibits acceleration and deceleration of keys when pressed and released by the user. This simple information can serve as a guideline to discriminate different key clicks. The critical question we need to answer is that whether the sensors on wearable devices can discriminate between key clicks and capture the fine-grained movements between two consecutive clicks. In particular, we look for unique sensing patterns inherited from such acceleration and deceleration that could be used to facilitate the discrimination of key clicks and distance estimation of hand movement between two key clicks.

A key click can be separated into two consecutive time periods: *key pressing* and *key releasing* periods. The key pressing period starts when a person's finger touches the key and ends when the



(a) Moving along the X axis. (b) Moving along the Y axis.

Figure 3: Accelerometer readings from IMU.

finger presses the key to the bottom of the keypad (denoted as *pressing point*). The key releasing period starts when the person's finger releases the key from the bottom of the keypad and ends when the finger stops moving after it is detached from the key (denoted as *releasing point*). Intuitively, the hand accelerates towards the keypad while pressing the key before the pressing point, and decelerates and stops quickly due to the reaction force from the key that touches the bottom of the keypad. When releasing the key, the hand accelerates towards the opposite direction to the keypad and stops after the finger is detached from the keypad. We illustrate the hand's acceleration/deceleration in the Z-axis caused by key pressing and releasing in Figure 1. We use the keypad's coordinate system with the Z-axis perpendicular to the keypad plane and pointing out from the keypad, and the X-axis aligned to the direction connecting the first and the second key.

Furthermore, in between two consecutive key clicks, the key entry activity involves the hand movement from one key to another. As shown in Figure 1, the accelerations on the X axis present an obvious up-and-down trend, while the accelerations on the Z and Y axes remain stable. The intuition behind this phenomenon is that the hand usually accelerates and moves relatively in parallel with the keypad on the shortest trajectory between the first and second keys. After passing the middle point of the trajectory, the hand decelerates to stop when it reaches the Key 2's position. Such unique up-and-down acceleration trend is very useful to help capturing the small distance of hand movement between two keys.

Feasibility Study. To study whether the sensors on wearables can capture such detailed acceleration patterns during key entry activities, we conduct two sets of experiments on the number pad of a Dell USB wired keyboard L100 with an Invensense MPU-9150 9-axis motion sensor (i.e., IMU), which is a prototyping alternative to a wearable device. The sensor uses a moderate sampling rate of 100Hz and contains an accelerometer, gyroscope and magnetometer that are comparable to embedded sensors in wearable devices. During the experiments, the participant wears the sensor on his wrist and keeps his hand in parallel to the keypad below so that the sensor's Z axis points out and is perpendicular to the keypad. The first set of experiments moves from keys 4 to 5, which is along the sensor's X axis, and the second set of experiments moves from keys 5 to 8 along the sensor's Y axis. The distance between keys 4 to 5 is only $1.9cm$, the same as that between keys 5 to 8. We use a camera on top of the keyboard to record the moving distance ground truth of the sensor. We note that these two experiment setups are special as the sensor's coordinate system is fully aligned with the keypad's coordinate system.

We estimate the sensor's moving distance by applying the double integration to the acceleration readings of the X axis and the Y axis from the accelerometer on the sensor. The details of the distance estimation scheme are presented in Section 5. Figure 2 compares the ground truth and the estimated distance in 10 runs of aforementioned settings, respectively. We find that overall the estimation errors are less than $1cm$, the mean error of the 10 runs of

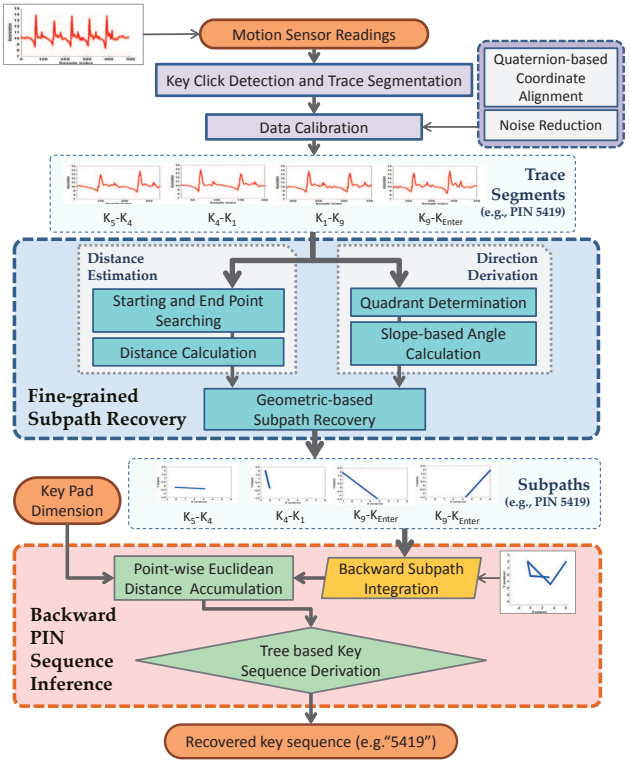


Figure 4: PIN-sequence inference framework.

each experimental setting is as low as $0.27cm$ and $0.24cm$ on the X and Y axes, respectively.

Additionally, we find that there is an unique up-and-down acceleration pattern captured by the sensor, which can be utilized to determine the sensor's moving direction. Figure 3 shows that the up-and-down acceleration pattern (like a sine wave) appears on X and Y axes respectively when the sensor is moving along X or Y axes. The capability of accurate distance estimation of the small moving distance between keys and the moving direction determination are the foundation for recovering the user's secret PIN sequence. Thus, these observations are encouraging as they indicate the sensors on wearables have the capability to capture the fine-grained hand movements to facilitate PIN sequence recovery.

4. SYSTEM DESIGN

In this section, we discuss the challenges in our system design and provide an overview of our system.

4.1 Challenges

The goal of accurately recovering personal PIN sequences by using the embedded sensor of wearable devices worn on the victim's wrist is not trivial. Our system design and implementation need to overcome the following challenges:

Robust Fine-grained Hand Movement Tracking. Using embedded sensors in wrist-worn wearable devices to reconstruct the trajectories of hand movements in key-entry activities is challenging since the sensors not only capture the acceleration patterns of key clicks and movements from key to key, but also are affected by the users's unconscious hand vibration and rotation. Furthermore, due to the limited size of the keypad, the distance between keys is small, making it hard to estimate using the low-grade sensors on wearables. Thus, we need to design distance estimation and direction derivation schemes to accurately estimate the hand moving distance between keys and track the direction of fine-grained hand

movements despite various interfering sensing factors.

Training-free Key Entry Recognition. Considering the attacking nature of our goal, it would be unlikely for the adversary to collect any training data (e.g., sensor data of hand movements) before recovering a user's PIN sequence. And it is also unlikely to have the user's cooperation during this process. Thus, we aim to infer the user's secret PIN sequence leveraging wearables without training efforts involving target users' participation.

Recovering PIN Sequence without Contextual Information. The target user's PIN sequences used in key-based security systems are usually consisted of numbers without contextual information or linguistic meaning. Our developed method should have the ability to recover sensitive information consisting of random combination of numbers. This requires our system to be able to recover PIN sequences without relying on linguistic model or dictionaries.

Sensing with Single Free-axis Wearable Device. Using a single wearable device to recover PIN sequence is necessary because usually there is only one wearable device available on the wrist of the hand that performs key entry activities. There is no reference point available besides the single wearable device. Furthermore, sensor readings are with respect to the wearable device's coordinate system, which is not stable and changes often according to the device's posture. In order to recognize key entry activities and derive fine-grained hand movement trajectories, it is important for our system to translate the sensor readings from the wearable device's coordinate system to a fixed coordinate system, such as the keypad's coordinate system.

4.2 System Overview

The main goal of our work is to demonstrate that using wearable devices could reveal people's secret PIN sequence to key-based security systems such as ATM machines, electronic-key based door entries, and enterprise servers. We design and implement a system that has the capability to reveal target user's secret PIN sequences through tracking the fine-grained hand movement trajectories related to key entry activities. The basic idea is to examine the acceleration of the user's hand movements when accessing key entry based security systems. Based on the feasibility study of two special cases in Section 3, wrist-worn wearables can capture the unique patterns of acceleration embedded in the hand movements caused by entering the secret PINs. Such unique patterns can be exploited to estimate hand moving distances and directions during the key-entry activities, which can be leveraged to reconstruct fine-grained moving trajectories of the user's hand and infer the PIN sequence traversed by the trajectories. We note that our approach can also be extended to recover letters on any kind of keypad.

The flow of our system is illustrated in Figure 4. Our system takes as input the raw sensor readings, such as acceleration, rotation rate, and quaternion, from the wearable device worn on a target user's wrist. Then the system performs *Key Click Detection and Trace Segmentation* to detect each key click by examining accelerations and separate the sensor readings into segments containing consecutive key entries. The *Data Calibration* utilizes *Quaternion-based Coordinate Alignment* and *Noise Reduction* techniques to translate each segment of accelerations into the measurements with respect to the coordinate system of the keypad, and remove noise from readings by using the Savitzky-Golay filter.

The core of our system consists of two components, *Fine-grained Subpath Recovery* and *Backward PIN-Sequence Inference*, which first estimate the distance and direction of hand movements in each segment of acceleration collected between two consecutive key entries, and then integrate the estimated distance and direction of each segment to determine the entire PIN sequence based on the physical constraints of the keypad and temporal relationship of the key entering sequence. We define a *subpath* as the trajectory of the user's hand movement between two consecutive key clicks inside one segment. As shown in Figure 4, the Fine-grained Subpath Recovery consists of two subtasks: *Distance Estimation* and *Direction Derivation*. The Distance Estimation identifies the unique acceleration patterns embedded in the key pressing and releasing activities and perform distance estimation based on such patterns. Additionally, the Direction Derivation leverages the estimated distance together with the acceleration patterns caused by the hand movement in each subpath to derive the hand moving direction.

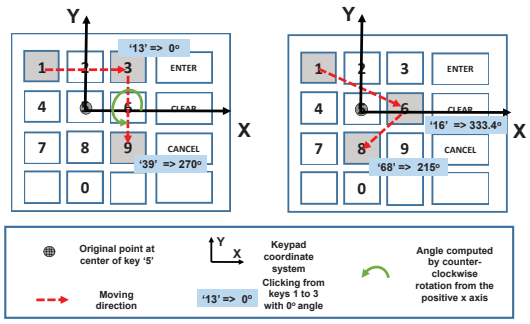


Figure 5: Illustration of the coordinate system on a typical key pad and examples of moving directions of key clicks, 13**,** 39**,** 16**, and** 68**.**

After obtaining the estimated moving distance and direction in each subpath, the system develops the Backward PIN-Sequence Inference to recover the user's PIN sequence. Specifically, our system first applies the *Backward Subpath Integration* to combine subpaths in a backward manner in time series. Then the system performs *Point-wise Euclidean Distance Accumulation* to calculate the accumulated Euclidean distance for each candidate of key sequence at each estimated key position (i.e., point-wise). Last, the *Tree based Key Sequence Derivation* generates a tree with the candidates of key sequence and their accumulated Euclidean distance. The key sequence candidate with the minimum accumulated Euclidean distance is chosen to be the output of the system, which is the inferred PIN sequence that the victim uses in the key-based security system.

5. DISTANCE ESTIMATION AND DIRECTION DERIVATION SCHEMES

Our system requires tracking hand movement trajectories on small keypads accurately without training. Inspired by the basic dead reckoning technique, we seek to derive such fine-grained trajectories based on hand movement distances and directions. Particularly, we develop *Distance Estimation* and *Direction Derivation* schemes to estimate the distances and derive direction for each subpath (i.e., between two consecutive key clicks).

5.1 Distance Estimation

In order to accurately estimate the hand movement distance between two consecutive key clicks, we need to identify the patterns in the sensor data corresponding to the hand movement precisely. Therefore, our system needs to first search the starting and ending points of the sensor data caused by the hand movements based on pressing and releasing points of key clicks; then calculate the hand moving distance by utilizing the extracted patterns from the sensor data. In the rest of the section, we assume the system has performed the *Key-click Detection* and segmented the sensor data to traces that capture hand movements between two consecutive key clicks. The sensor data in each trace are translated into keypad coordinate system through *Coordinate Alignment*. The details of

Key-click Detection and Coordinate Alignment will be discussed in Section 7. Figure 5 illustrates the coordinate system of a typical ATM keypad, where the center of key 5 is the origin; the directions of positive X and Y axes are in parallel with the direction from keys 5 to 6 and keys 5 to 2, respectively; and the Z axis is perpendicular to the X-Y plane, pointing out from the surface of the keypad. The four quadrants of the X-Y plane are defined as the standard quadrants in a two-dimensional Cartesian system. Figure 5 also shows some examples of moving directions of key clicks, e.g, 13 indicates clicking from keys 1 to 3.

Starting and Ending Points Searching based on Pressing and Releasing Points. The hand movements from one key to another happen after releasing the first key and end when touching the second key. Ideally, the hand movement distance can be calculated based on the acceleration (e.g., acceleration from the Z-axis) extracted between the releasing point of the first key click and the pressing point of the second key click. However, such coarse segmentation includes the sensor data resulted from hand vibrations usually result in large estimation errors. In Section 3, we find that the acceleration captured during the hand movements between consecutive key clicks has significant and unique patterns on X and Y axes (i.e., either up-and-down or down-and-up shapes due to different moving directions).

Apparently, such unique acceleration patterns include merely the dynamics of the key-to-key hand movements, and can be further utilized to facilitate accurate hand moving distance estimation. In order to determine the right segment of acceleration data corresponding to the unique acceleration pattern, we propose to further search the starting and ending points of the pattern based on the segment of sensor data. Specifically, we define the first zero-crossing point occurring before and after the unique acceleration pattern as the *starting point* and *ending point*, respectively. The intuition behind this is that when a hand moves from one key to another, its moving trajectory is mainly in parallel with the X-Y plane of the keypad. Therefore, the acceleration and deceleration of the hand during such movement dominates the acceleration on X and Y axes, and results in the acceleration that always experiences a pattern of $[0, a_{k,max}(a_{k,min}), 0, a_{k,min}(a_{k,max}), 0]$ as shown in Figure 6, where $a_{k,max}$ and $a_{k,min}$ denote local maximum and minimum of acceleration on X and Y axes with $k = x$ or y.

Thus, we design a strategy to locate the starting and ending points of the unique acceleration pattern so that we could estimate the distance between two key clicks accurately. Our strategy involves the following steps: 1) extract the acceleration on X and Y axes between the releasing and pressing points of two consecutive key clicks respectively; 2) examine the extracted acceleration to find the $a_{x,max}, a_{x,min}, a_{y,max}, a_{y,min}$; 3) determine the *dominated axis* by choosing the axis has the more significant unique acceleration pattern (i.e., a larger peak-to-peak value defined by $|a_{k,max} - a_{k,min}|, k = x$ or y); 4) find the starting point of the unique pattern on the dominated axis by searching the first time that acceleration crosses the axis (i.e., zero-crossing point) before $a_{k,max}$ or $a_{k,min}$, whichever occurs earlier; 5) similarly, find the ending point of the unique pattern on the dominated axis by searching the first zero-crossing point after $a_{k,min}$ or $a_{k,max}$, whichever occurs later. The accelerations within the starting and ending points derived above merely correspond to the hand movements between two consecutive key clicks and are utilized to calculate the hand movement distance and direction in our schemes.

Distance Calculation. The distance estimation between two consecutive key clicks is obtained by considering the movements in both X and Y axes. To perform accurate estimation, we compute the small movement between two samples in sensor data and then sum up to produce the distance estimation in one acceleration segment bounded by the identified starting and ending points. As the distance is two times integration of accelerations, we utilize trapezoidal rule to approximate each integration.

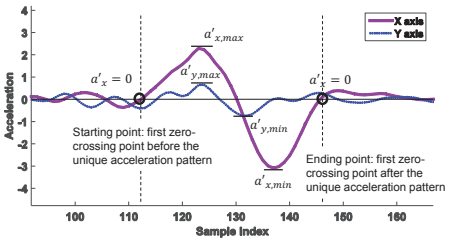


Figure 6: Searching for starting and ending points based on releasing and pressing points within an acceleration segment.

5.2 Direction Derivation

In order to recover the complete PIN sequence, our system needs to determine the moving direction of each subpath during the key-entry process in addition to the distance. We define the moving direction of a subpath as the angle between the positive X axis and the subpath with counter-clockwise rotation as shown in Figure 5. The moving direction is denoted as $\vartheta \in [0^\circ, 360^\circ)$. The basic idea is to find the direction based on the ratio of distances on X and Y axis derived from hand movement acceleration. In particular, we design a two-step approach, including the *Quadrant Determination* and *Slope-based Direction Calculation*. The Quadrant Determination first leverages the unique acceleration patterns to determine which quadrant of X-Y plane that the hand moving direction belongs to. Then the Slope-based Direction Calculation examines the slope angle of the moving direction in a quadrant ranging from 0° to 90° based on the hand movement distances on X and Y axes, and converts the slope angle to the moving direction ϑ.

Quadrant Determination. Intuitively, the hand movement acceleration projected on X and Y axes results in different combinations of the unique acceleration patterns in terms of the order of $a_{k,max}$ and $a_{k,min}$ on X and Y axes with $k = x$ or y. For example, when the hand moves towards 45°, the acceleration on X and Y axes both experiences the $a_{k,max}$ before the $a_{k,min}$, while the acceleration on the X axis experiences the $a_{x,max}$ after the $a_{x,min}$ and the acceleration on the Y axis experiences the opposite when the hand moves towards 135°. Therefore, we leverage the combinations of unique acceleration patterns on X and Y axes to determine the quadrant that a certain moving direction should belong to. Specifically, the quadrant of the moving direction can be determined by the following equation:

$$Q = \begin{cases} 1;\ if\ I_{a_{x,max}} < I_{a_{x,min}} \& I_{a_{y,max}} < I_{a_{y,min}}, \\ 2;\ if\ I_{a_{x,max}} > I_{a_{x,min}} \& I_{a_{y,max}} < I_{a_{y,min}}, \\ 3;\ if\ I_{a_{x,max}} > I_{a_{x,min}} \& I_{a_{y,max}} > I_{a_{y,min}}, \\ 4;\ if\ I_{a_{x,max}} < I_{a_{x,min}} \& I_{a_{y,max}} > I_{a_{y,min}}. \end{cases} \quad (1)$$

where Q is the quadrant index, $I_{a_{axe,max}}$ and $I_{a_{axe,min}}$ denotes the index of the local maximum and minimum on X and Y axes, respectively.

Slope-based Direction Calculation. After quadrant determination, we compute the slope angle of the moving direction within each quadrant based on the ratio of the distance on X and Y axes by utilizing the following equation:

$$\phi = \left| arctan\left(\frac{s_y}{s_x}\right) \right|. \quad (2)$$

Equation (2) returns the relative moving direction defined in a quadrant ranging from 0° to 90°, we further convert the ϕ to an absolute moving direction (i.e., the direction defined within keypad coordi-

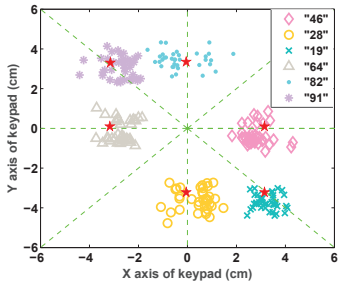


Figure 7: Illustration of the clustering results of distance estimation and direction derivation for 6 different subpaths $\{46, 28, 19, 64, 82, 91\}$ by treating the first key click as the origin. The red star is the ground truth.

nate ranging from 0° to 360°). Given the quadrant index Q, the absolute moving direction ϑ can be derived as follow:

$$\vartheta = \begin{cases} \phi; & if\ Q = 1, \\ 180^\circ - \phi; & if\ Q = 2, \\ 180^\circ + \phi; & if\ Q = 3, \\ 360^\circ - \phi; & if\ Q = 4. \end{cases} \quad (3)$$

Once we estimate the distance and derive the direction of a subpath, the relationship between two consecutive key clicks in the contained subpath is determined. Therefore, if the position of either key click is known, we can derive the position of the other key click according to the derived moving distance and direction. We show an example of distance estimation and direction determination for 6 subpaths $\{46, 28, 37, 64, 82, 73\}$. Figure 7 shows the clustering results in both distance and direction when treating the first click as the origin. We observe that each key-click combination is clustered together around the ground truth (shown as the red star) based on our distance estimation and direction determination schemes, indicating that our schemes have the capability to capture the fine-grained hand movement trajectories in key entry activities.

6. BACKWARD PIN SEQUENCE INFERENCE ALGORITHM

After performing *Fine-Grained Subpath Recovery* grounded on distance estimation and direction determination, we next describe how to reconstruct the hand-movement trajectory using the estimated subpaths to infer the target user's PIN sequence.

6.1 Backward Subpath Integration

We notice that all key-based security systems require the user to execute the verification by pressing key *Enter* or *Confirm*, which is at a known position on the keypad. We can then utilize this information to reconstruct the hand-movement trajectory on the keypad by examining the subpaths in a backward time sequence. That is, the position of key Enter can be considered as a end of the last subpath, and the starting of the last subpath indicates the position of the last key clicked before key Enter.

More generally, we concatenate the estimated end of the $(j-1)^{th}$ subpath to the starting of the j^{th} subpath and continue to repeat this step until reaching the starting of the first subpath. By integrating all the derived subpaths in such a backward head-tail connecting way, we can obtain a trajectory roughly matching the hand movements during the key-entry process, called the *Naively Integrated Trajectory*. Ideally, the vertices on the Naively Integrated Trajectory should be mapped to real-key positions with the last vertex mapping to the center of Key Enter.

6.2 Point-wise Euclidean Distance Accumulation

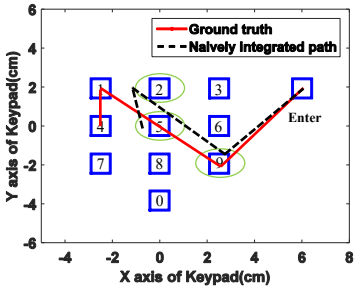


Figure 8: Example of the naively integrated trajectory having a large accumulated error cannot correctly map to the key positions of the PIN sequence "419" (though the estimation error of distance and direction of individual subpath is small).

Although we can recover each individual subpath based on the estimated distance and derived direction, each subpath contains small errors and the Naively Integrated Trajectory inherits and further accumulates such small errors in each subpath, resulting in mapping to the wrong-key positions on the keypad. Figure 8 shows an example that the naively integrated subpaths (i.e. in black dashed lines) cannot recover the correct target user's PIN sequence, e.g., "419", instead, they return "529" as a result. To reduce cumulative errors, we propose a *Point-wise Euclidean Distance Accumulation* approach. In this approach, instead of matching the Naively Integrated Trajectory directly to the keys on the keypad, we consider each subpath separately by comparing the closeness in terms of the Euclidean distance between the starting point of the subpath (i.e., point-wisely) and real key positions, while the ending point of the subpath is fixed on real keys.

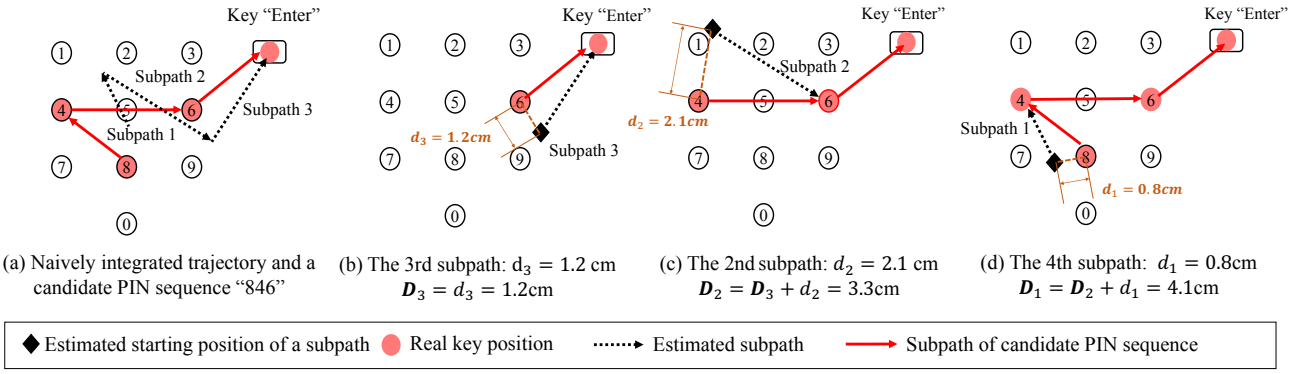
In particular, each subpath j contains the estimated distance (S_j) and direction (ϑ_j). Given a real key's position as an ending point (assuming this key is clicked at this ending point), we can estimate the starting point $(\widetilde{x_j}, \widetilde{y_j})$ of each subpath. We conduct this effort in a backward manner starting from Enter key because we know the ending point in the last subpath is the Enter key. The estimation of the starting point in the j^{th} subpath is obtained as following:

$$\begin{cases} \widetilde{x_j} = cos(\vartheta_j + 180) \times S_j + \mathcal{X}, \\ \widetilde{y_j} = sin(\vartheta_j + 180) \times S_j + \mathcal{Y}, \end{cases} \quad (4)$$

where $(\mathcal{X}, \mathcal{Y})$ are the coordinates of ten real number keys $\{1, 2, 3, ..., 9, 0\}$ on the keypad. Given that there are ten real number keys in the key pad, there will be ten estimation results of the starting points in subpath j. We note that, for the last subpath, $(\mathcal{X}, \mathcal{Y})$ is the coordinates of the key Enter. Once the starting point of the j^{th} subpath is estimated, our algorithm will recursively move to the previous subpath. By doing so, we introduce the concept of *accumulated Euclidean distance*, which is the sum of the Euclidean distances between the starting point of a subpath and the coordinate of a real key in the keypad, over all consecutive subpaths. We can recursively run the following equation to calculate the accumulated Euclidean distance:

$$\mathbb{D}_j = \mathbb{D}_{j+1} + d_j, \quad (5)$$

where $\mathbb{D}_j$ and $\mathbb{D}_{j+1}$ denote the accumulated Euclidean distance of two consecutive subpaths, respectively, and d_j is the Euclidean distance between the estimated starting point $(\widetilde{x_j}, \widetilde{y_j})$ of the j^{th} subpath and a real key in the keypad. The resulted final accumulated Euclidean distance measures the closeness of the real key combination, defined as *PIN sequence candidate*, to the estimated consecutive subpaths while leveraging the dimension of the keypad. The insight is that we would like to explore the possible candidate keys



(a) Naively integrated trajectory and a candidate PIN sequence "846"

(b) The 3rd subpath: $d_3 = 1.2$ cm
$\boldsymbol{D}_3 = d_3 = 1.2$cm

(c) The 2nd subpath: $d_2 = 2.1$ cm
$\boldsymbol{D}_2 = \boldsymbol{D}_3 + d_2 = 3.3$cm

(d) The 4th subpath: $d_1 = 0.8$cm
$\boldsymbol{D}_1 = \boldsymbol{D}_2 + d_1 = 4.1$cm

Figure 9: Example of point-wise Euclidean distance accumulation for candidate PIN sequence "846", where the real PIN is "419".

leveraging the estimation from each subpath without fixing to a particular key matching. In this way, we will not end up with only one Naively Integrated Trajectory, instead, we will obtain multiple key sequences as the candidates for PIN sequence recovery. Furthermore, by conducting the point-wise Euclidean distance accumulation for each candidate of PIN sequence, our algorithm balances the contribution of each estimated subpath and reduce the accumulated errors that impact the accuracy of PIN sequence inference.

Example. Figure 9 shows an example of how the Euclidean distance is accumulated point-wisely in backward for a specific candidate PIN sequence "846" (The real PIN entry in this example is "419"). In the sequence of Figure 9, (a) we first generate the Naively Integrated Trajectory consisted of three consecutive subpaths, *subpath* 1, *subpath* 2, and *subpath* 3, which need to be point-wisely compared with the candidate subpaths: "84","46", and "6$enter$" in the candidate PIN sequence "846". The generation of naively integrated trajectory is based on the estimated distances and derived directions of each subpath. (b) then we start by mapping the ending point of subpath 3 to the key Enter and set $\mathbb{D}_4 = 0$, and utilize the estimated moving distance and derived direction in the subpath to estimate its starting point on the keypad in a backward way. The Euclidean distance between the estimated starting point of subpath 3 and key 6 (i.e. the 3^{rd} key entry in the candidate PIN sequence "846") is found to be $d_3 = 1.2cm$, and the accumulated Euclidean distance for this subpath is $\mathbb{D}_3 = \mathbb{D}_4 + d_3 = 1.2cm$; (c) next, assuming the ending point of subpath 2 is mapped to key 4, we similarly estimate the starting point of the subpath and calculate the Euclidean distance between the estimated starting point and the position of key 4 (i.e., $d_2 = 2.1cm$). The accumulated Euclidean distance for the previous two supaths is $\mathbb{D}_2 = \mathbb{D}_3 + d_2 = 3.3cm$; (d) lastly, we assume the ending point of the subpath 1 to be key 8 and estimate the starting point of the subpath. We find the Euclidean distance between the estimated starting point and the position of key 8 to be $d_1 = 0.8cm$ and calculate the accumulated Euclidean distance for the entire candidate of PIN sequence "846" as: $\mathbb{D}_1 = \mathbb{D}_2 + d_1 = 4.1cm$. We note that our algorithm recursively calculates the accumulated Euclidean distance for every possible candidate of PIN sequence based on Equations (4) and (5) and select the candidate with the minimum accumulated Euclidean distance as the final result.

6.3 Tree-based Key Sequence Inference

To implement the Backward PIN-Sequence Inference algorithm, we develop a tree-based approach for the PIN-sequence inference. Next, we discuss how to build and optimize the tree in our algorithm.

Building a Tree with PIN Sequence Candidates. In order to record and compare different candidates of PIN sequence, we seek to build a decimal tree according to the backward order of all PIN sequence candidates. Each node is defined as a 2-tuple structure containing its corresponding key entry and the Euclidean distance accumulated on the path from the root node to the node, denoted as $< NodeKey, AccuDist >$. Because the tree is built based on a backward order, nodes in the j^{th} level of the tree correspond to the $(N-j)^{th}$ key entries of all candidates of PIN sequences. The root node is always the last key entry (i.e., key Enter), while the leaf nodes are always the first key entry of the candidate of PIN sequence (i.e., number keys on the keypad). Each node (except the leaf nodes) has 10 child nodes corresponding to keys 0 to 9. The branches from one parent node to its child nodes represent the subpaths between the keys corresponding to the parent and child nodes. The leaves of the tree stores the final accumulated Euclidean distance of each candidate of PIN sequence. Our algorithm searches for the leaf node having the minimum accumulated Euclidean distance, and traces back to recover the path from the leaf node to the root node. The inferred PIN sequence is generated by recording the key entries corresponding to the nodes on the recovered path.

Figure 10 shows an example of a tree for inferring a PIN sequence of "419", where the accumulated Euclidean distance for one candidate of PIN sequence "846" is $4.1cm$, while another candidate of PIN sequence "419" has the accumulated Euclidean distance of $1.6cm$, which is the minimum over all candidates. Therefore, the candidate of PIN sequence "419" will be determined to be the inferred PIN sequence.

Subpath Calibration and Tree Pruning. In order to improve the accuracy of our system, we take the advantage of the keypad dimension to calibrate subpaths. Intuitively, the distance of a subpath should not exceed the dimension of a keypad. Therefore, if the estimated distance of a subpath exceeds the dimension of a keypad, our system replaces the estimated distance of the particular subpath with the possible longest distance on the keypad. In addition, since every non-leaf node in a PIN-sequence tree has 10 child nodes, the j^{th} level has 10^j nodes. Apparently, it is not necessary to store and calculate the Euclidean distance in every node. Our algorithm prunes the tree by keeping the child nodes with the least m accumulated Euclidean distances for each parent node. In this way, leaf nodes are largely reduced from 10^N to m^N, where N is the length of the PIN sequence. In our experiments, we set $m = 4$, which balances the tree size and produces good performance.

7. IMPLEMENTATION

7.1 Key-click Detection

Given embedded sensor data from wearable devices, our system first performs key-click detection based on acceleration readings to find the key-click events and the number of keys in a PIN sequence

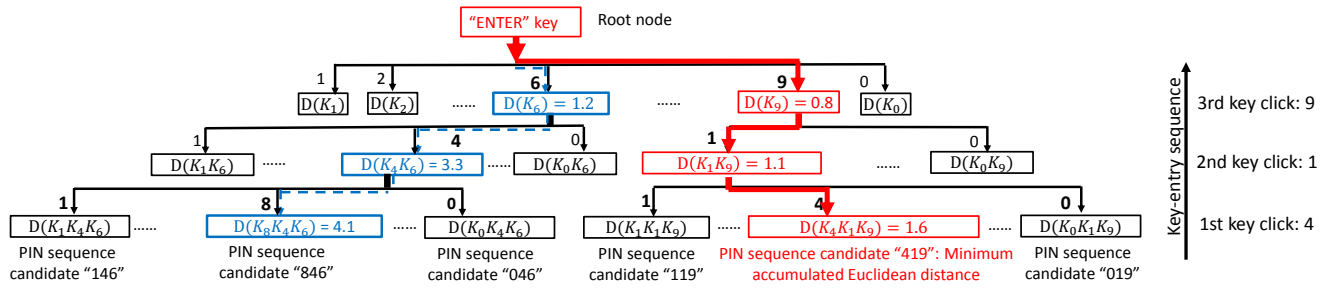


Figure 10: Illustration of the construction of the backward trajectory inference tree for recovering PIN "419".

and assist the trace segmentation. Key clicks usually cause significant changes of acceleration towards the keypad that has the potential to be distinguished from other hand movements. In particular, we calculate the magnitude of the composition of accelerations on three axes first, and apply a threshold to examine the normalized magnitude of the composed acceleration to detect key clicks. We empirically determine the threshold to be 0.6 based on our experiments with 20 participants in this work.

7.2 Key-click Trace Segmentation

After key-click detection, we roughly segment input sensor data into small chunks containing the data between two consecutive detected key clicks. After segmentation, the resulted small chunks contain the sensor data representing subpaths, which include the acceleration caused by hand movements from one key to another. In addition, to mitigate high frequency noise caused by hand vibration, we apply the $Savitzky-Golay\ filter$ to each chunk of sensor data respectively.

7.3 Quaternion-based Coordinate Alignment

When recovering the user's PIN sequence from the wearables' embedded sensors, our system involves three different coordinate systems, namely, *wearable coordinate*, *world coordinate* and *keypad coordinate*. The sensor readings from a wearable are defined within the wearable coordinate and thus cannot be used directly to represent hand movements because of the rotating wearable coordinate caused by frequently changed hand position. In this work, we employ quaternion to help convert sensor readings from the wearable coordinate to keypad coordinate for hand trajectory derivation.

Specifically, we first convert the sensor readings from the wearable coordinate to world coordinate by applying $\vec{a}_w = q_{dw}\vec{a}_d q_{dw}^{-1}$, where $\vec{a}_w$ and $\vec{a}_d$ are the sensor readings in the world coordinate and werable coordinate, respectively, and q_{dw} is the quaternion that represents the conversion from the werable coordinate to world coordinate. Then a_w will be further converted to the keypad coordinate via $\vec{a}_k = q_{wk}\vec{a}_w q_{wk}^{-1}$, where $\vec{a}_k$ denotes the sensor readings in the keyboad coordiante and q_{wk} denotes the quaternion that represents the conversion from the world coordinate to keypad coordinate. The quaternion q_{dw} can be extracted from wearables during hand movements, and q_{wk} can be derived from $q_{wk} = q_{kw}^{-1}$, where the quaternion q_{kw} can be collected by placing a sensor (i.e., smartphone, smartwatch, or IMU) aligned with the coordinate of the target keypad. We note that adversaries can utilize this method to obtain q_{kw} without attention at a time other than the user entering the PIN sequence.

8. PERFORMANCE EVALUATION

In this section, we present the experimental methodology and describe the evaluation metrics. We then present the most important results of our system with respect to PIN sequence recovery using the Backward PIN-sequence Recovery Algorithm. Finally, we show the performance of two supporting schemes for PIN sequence recovery, distance estimation and direction derivation schemes.

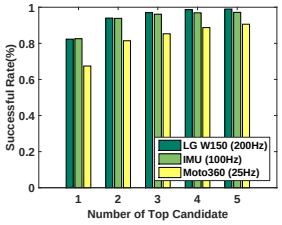
Figure 11: Experiments: three different kinds of keypads, detachable ATM pad, keypad on ATM machine, keyboard; and wearable devices.

8.1 Experimental Methodology

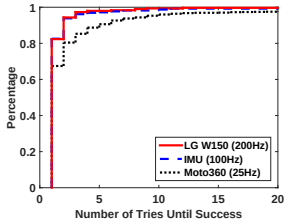
Keypads. We evaluate our system with three different kinds of keypads as shown in Figure 11: 1) A keypad on ATM machine (from PNC bank) with the dimension of $108mm \times 76mm$; 2) A real detached ATM keypad with the dimension of $127mm \times 95mm$, both 1) and 2) representing the use cases with different ATM pad sizes; and 3) A number pad of Dell USB wired keyboard L100 with the dimension of $77mm \times 97mm$, representing the use case of key-based security access to enterprise servers. The three keypads have different structures and key depths. It is important to evaluate their effects on our approach when capturing fine-grained hand movements. We focus on experiments on numbers to recover PIN-sequences.

Wearable Devices. In our experiments, we use three different types of wearable devices, including two smartwatches (i.e., LG W150 and Moto360) and an IMU (Invensence MPU-9150). These wearables represent different achievable maximum sampling rates (i.e., 200Hz, 25Hz and 100Hz, respectively). The LG W150 and Moto 360 are two commodity smartwatches running on Android Wear OS with Bluetooth LE. The IMU contains a 9-axis motion tracking sensor designed for consumer electronics. We use it as a prototyping alternative to a wearable device with its sampling rate set to 100Hz. During key-entry activities, the wearable devices collect acceleration and quaternion data and send them to a pre-associated storage device (i.e., smartphone via Bluetooth and laptop via an USB cable for smartwatches and IMU respectively). The ground truth of the hand moving distance and direction is computed through the video recorded by a camera set on top of the keypad. In particular, we use AutoCAD to connect two positions of the sensor in two captured video frames corresponding to the time points when the finger just leaves the first key and about to touch the second key, respectively. The measured distance and angle of the line (with the positive X axis of the keypad) connecting these two sensor positions are used as the ground truth of the distance and direction of the hand movement.

Data Collection. We conduct experiments of various key-entry activities with three different types of wearables on three kinds of

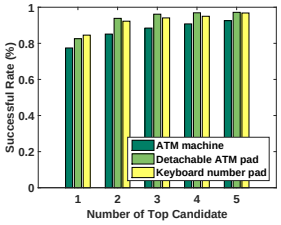


(a) Success rate of recovering PIN sequence within top-k candidates.

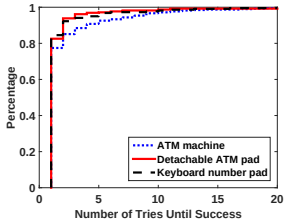


(b) Cumulative distribution function of the number of tries until success.

Figure 12: Performance of Backward PIN-sequence Inference with three kinds of wearables on the detachable ATM Keypad.



(a) Success rate of recovering PIN sequence within top-k candidates.



(b) Cumulative distribution function of the number of tries until success.

Figure 13: Performance of PIN-sequence recovery on three different keypads by using medium sampling rate 100Hz (IMU).

keypads. 20 volunteers are recruited to performance key-entry activities over an 11-month period. The volunteers are asked to enter keys in two ways: 4-digit PIN sequences consisting of five consecutive key clicks (with "Enter" as the last click) and a single subpath consisting of two consecutive key clicks. For each subpath, based on the keypad layout, we classify different subpath lengths into three representative scales: *short*, *medium* and *long*. Specifically, *short* covers subpaths between two adjacent keys with no keys in between (e.g., 45, 41 and 75); *medium* is for horizontal and vertical subpaths between two keys with one key in between (e.g., 46 and 82); and *long* contains subpaths of two keys neither horizontal nor vertical and with one or more keys in between (e.g., 10, 37 and 29). We collect 5000 PIN sequences from three keypads when having 20 volunteers wear three different kinds of wearables. For single subpath, we collect 3000 subpaths from three keypads including *long*, *medium* and *short* distances with volunteers wearing an IMU.

8.2 Evaluation Metrics

We develop the following metrics to evaluate our system with regard to the accuracy of distance estimation and direction determination schemes and the performance of our Backward PIN-sequence Inference Algorithm:

Distance Estimation Error. To evaluate the performance of our distance estimation scheme, we define the *Distance Estimation Error* as the difference between the estimated distance and the ground truth of the hand moving distance. The ground truth of the hand moving distance is computed through the recorded video during experiments. We study the Distance Estimation Error in two ways: *mean error* and *cumulative distribution function (CDF)*.

Direction Classification Accuracy. To evaluate the performance of our direction derivation scheme, we divide the 360° on the X-Y plane into 16 groups (i.e., 5 groups in each quadrant excluding 4 overlapped groups) and examine whether the derived direction is classified into the same group as that of the corresponding ground truth. The ground truth of angles is also computed through the recorded videos. The *Direction Classification Accuracy* is $\frac{\tilde{N}_c}{N_c}$, where $\tilde{N}_c$ is the number of directions have been classified into the same group containing the corresponding ground-truth direction, and N_c is the total experimental runs of direction classification.

Top-k Success Rate. Given an experimental run of a key-entry activity, our algorithm could return multiple top candidates of key-entry sequence in an ascending order of the accumulated Euclidean distance. We define that the inference algorithm is a *Top-k Success Hit* if the first k candidates of key-entry sequence returned from our algorithm contain the target user's key-entry sequence. We further define the *Top-k Success Rate* as the ratio ($\frac{\tilde{N}_s^k}{N_s}$) of the number of Top-k Success Hits ($\tilde{N}_s^k$) over the total number of experimental runs (N_s) when applying key-entry sequence inference to recover the target user's PIN sequence. Specially, when $k = 1$, the ratio indicates the rate of our algorithm that can successfully determine the target user's key-entry sequence without ambiguity.

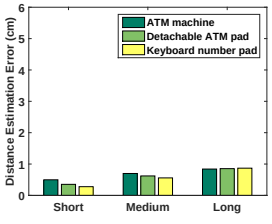
Tries Until Success. Since our system can provide multiple candidates as the result for key-entry sequence inference, the adversary has the chance to try out each key sequence returned in the candidate list to recover the target user's PIN sequence. We define the *Number of Tries Until Success* as the number of candidate key-entry sequence the adversary has tried (starting from the candidate with the smallest accumulated Euclidean distance) until he/she breaks the key-based security system, suggesting a success recovery of the target user's PIN sequence. Thus, the Number of Trails Until Success indicates the possible efforts that an attack needs to take to break the key-based security system.

8.3 Performance of Backward PIN-Sequence Inference

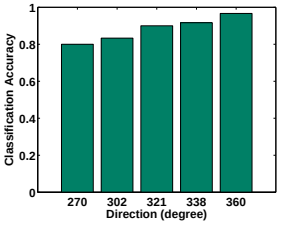
Wearable Devices. We first examine the performance of our PIN-sequence inference algorithm on the detachable ATM keypad with three different wearable devices. Figure 12(a) shows the top-k success rate of our system from three different types of wearable devices. We find that our system can effectively recover PIN sequences from all the three wearables, and higher success rate is achieved under higher sampling rates. In particular, by choosing the top-1 choice, our system can achieve over 82% success rate for the LG W150 and IMU, while the success rate is 67% for the Moto 360 . Furthermore, the PIN sequences can be inferred with increasing success rates if the adversary utilizes more choices from the top-k candidate list. Specifically, when using the top-2 choices, the adversary can achieve about 94% success rate with the LG W150 and IMU, and the success rate for the Moto 360 is over 80%. Although the Moto 360 achieves lower success rates than the LG W150 and IMU due to its much lower sampling rate (i.e., 25Hz), an adversary can still achieve a high probability to reveal the PIN sequences based on top-2 or 3 choices. This indicates that our system can tolerate the insufficient information introduced by wearable devices with low sampling rates.

Figure 12(b) depicts the cumulative distribution of the number of tries until successfully recovering the user's PIN sequence from three wearables. We find that the adversary can break over 97% PIN entries from the LG W150 and IMU within 5 tries, which is usually the maximum PIN tries on ATM machine. The number of PIN entries revealed increases to 99%, if the attacker conducts 10 tries. For Moto 360, the attacker can break 90% PIN entries within 5 tries and 96% within 10 tries. Therefore, regardless of the types of wearable, the attacker can break the user's PIN sequence with few tries. Although the LG W150 is set to use 200Hz sampling rate and generates the best performance, we find that using 100Hz sampling rate is enough to achieve comparable good results. Therefore, we present the results using the IMU for the rest sections.

ATM Keypads and Keyboard. Figure 13(a) shows the top-k

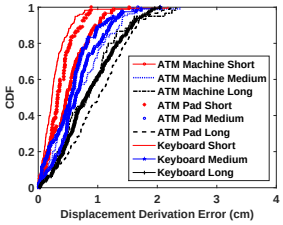


(a) Distance estimation error of three kinds of keypads.

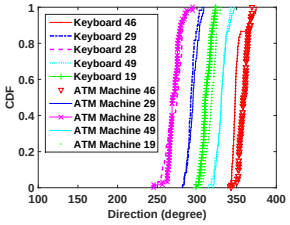


(b) Direction classification results of ATM machine.

Figure 14: Distance estimation mean error and direction classification results between two consecutive key clicks under 100Hz sampling rate (IMU).



(a) Cumulative distribution function of distance estimation errors.



(b) Cumulative distribution function of direction derivation errors.

Figure 15: Performance of distance estimation and direction derivation on three kinds of keypads under 100Hz sampling rate (IMU).

success rate to recover PIN sequences on three keypads. We observe that our system can achieve around 80% success rate for all three keypads with the top-1 choice. When using the top-5 choices, our system can achieve over 97% success rate with both of the detachable ATM pad and the number pad on keyboard, while on real ATM machine, the success rate is over 92.5%. Figure 13(b) confirms our observation in Figure 13(a). The results demonstrate that our Backward PIN-sequence Inference is effective when applied with keypads of different layouts and coordinates. The success rate is higher with both of the detachable ATM pad and the number pad on keyboard than that with the ATM machine. Our results suggests that the electronic magnetic field and the tilt angle of the ATM machine affect the PIN entry recovery result on ATM machine.

8.4 Distance Estimation of Different Kinds of Keypads

We next study the performance of two supporting schemes. The study of the distance estimation scheme is described in this subsection, and the results of the direction determination scheme is presented in the next subsection. We apply our distance estimation scheme to various subpaths across three different kinds of keypads. We compare the distance difference between ground truth (i.e., obtained from camera) and the estimated distance from sensor data. Take ATM machine as an example, the distances for *short*, *medium* and *long* are $2.5cm$, $5cm$ and $6.4cm$, respectively.

We observe that the mean error is proportional to the distance scale, i.e., short distance has relative smaller error compared with long distance, as shown in Figure 14(a). In particular, the mean error of ATM machine for short, medium and long distance are $5mm$, $7mm$ and $8.5mm$, respectively. For detachable ATM pad, the error of long, medium and short distance are $8mm$, $6mm$ and $3.5mm$, respectively. The mean error of long distance in keyboard number pad experiment is $8mm$, $5mm$ for medium distance and for short distance the error is as low as $3mm$. The experiment results from keyboard shows relative smaller distance error since the physical layout of keyboard number pad is smaller than ATM machine keypad and detachable ATM pad. We observe that such error difference is marginal and reveal the effectiveness of our scheme.

Figure 15(a) shows the cumulative distributive function of distance estimation errors. We observe that the 80th percentile errors are $8mm$, $10mm$ and $12mm$ for short, medium and long distance of ATM machine, respectively. For detachable ATM pad the 80th percentile error are $5mm$, $10mm$ and $13mm$, receptively and the 80th percentile error of number pad experiment are $4mm$, $8mm$ and $13.2mm$ respectively. The results also show the effectiveness and robustness of our scheme under various keypads.

8.5 Direction Derivation of Different Kinds of Keypads

Next, we evaluate our slope-based direction derivation scheme by showing the performance under three different kinds of keypads. According to the keypad layout, we select five representative directions in one quadrant. Take ATM machine as an example, the five directions within the fourth quadrant are: keys 2 to 8, keys 2 to 9, keys 1 to 9, keys 4 to 9 and keys 4 to 6. The corresponding direction angle for these subpaths on the keypad are: 270° , 302° , 321°, 338° and 360°. To evaluate our direction derivation scheme, we study the direction classification accuracy of classifying the directions of testing subpaths into the aforementioned 5 groups of directions angles. Figure 14(b) shows the direction classification accuracy with five directions on ATM machine. The X axis represents the ground truth direction between two keys on the ATM machine. We find that there are few subpaths mistakenly classified as incorrect direction. In particular, our scheme can achieve 80% classification accuracy for 270° and we observe that directions with larger angles have better accuracy, which is up to 97% accuracy for 360°. This may due to that when user performs vertical key clicks (e.g., key 2 to 8 with 270° on ATM pad), there might be a small inclined angle between hand moving direction and wrist moving direction. For keyboard and ATM pad, we have similar high classification accuracy. In addition, Figure 15(b) shows the cumulative distribution function of estimated five directions in the fourth quadrant. We find that all five directions obtained from our scheme only have small overlap for any two adjacent directions. Moreover, 90% of the derived direction are close to the ground truth direction within $\pm 10^\circ$. The above results show that our system provides effective distance estimation and direction derivation schemes under various keypads and is robust in real environments.

9. DISCUSSION

Wearing the Wearable Device on the Left Hand or Right Hand. Our training-free approach does not require mirroring the derivation from sensor data when applied to either the left-handed or right-handed user since the inherent physics of key entry activities will be preserved regardless of either case. We assume the victim use either hand wearing a wearable (i.e. a smartwatch or fitness tracker) to access key-based security systems. While it is very difficult to know the exact number of how many people sharing this style, we instead discuss the population of the potential wearable user victims. We take the right-handed user for discussion as the left-handed user share the same conclusion. Wearable devices are usually designed in a way that allows users to comfortably wear them on either wrist (e.g., smartwatches no longer necessarily have crowns as traditional watches do). There are many smartwatch users [2, 5] claiming that they wear smartwatches on their right wrists. Furthermore, for those wearing traditional watch on the left wrist, they tend to wear fitness tracker on the right wrist for health-related applications. Naturally, the right-handed people use their right hand to perform key entry and the sensors in their smartwatches or fitness trackers can be utilized by our approach to

reveal PINs. Given the growing cheaper price of these wearable devices, many people wear both a smartwatch and a fitness tracker on separate hand to better serve their work and health applications, which further increases the number of potential victims. Lastly, the increasing popular usage of wearables leaves adversary great chances to recover the user's sensitive information, making it vulnerable irrespective of the hand on which it is worn.

Using Sensor Moving Direction as Hand Moving Direction. We discuss the rationality of using sensor moving direction as hand moving direction. The current system is designed for recovering a PIN sequence by reconstructing hand movement trajectories. We leverage embedded sensor readings from wearable devices on a user's wrist to determine the direction. We use the sensor movement to represent the hand movement since the hand and the wrist are moving together. During our extensive experimental study, we observe that sensor movement and hand movement share similar moving trend. Therefore such a representation is reasonable.

10. CONCLUSION

In this paper, we show that the embedded sensors on wrist-worn wearable devices (i.e., smartwatches and fitness trackers) can be exploited to discriminate mm-level distances of the user's fine-grained hand movements during key-entry activities, exposing the user to a serious security breach. We present a PIN-sequence inference framework to recover the user's secret key entries when the user accesses key-based security systems such as ATM keypads and regular keyboards. The implemented system does not require any training or contextual information, which makes it applicable in real world adversarial contexts. In particular, our system exploits the physics phenomenon and unique patterns of key entry activities from the sensor data and develops distance estimation and slope-based moving direction derivation schemes to capture the small hand movement between two consecutive keys. Our system further applies the Backward PIN-sequence Inference Algorithm to reveal the user's complete PIN sequence, leveraging both the spatial and temporal constraints of the key entry to achieve a high success rate. Extensive experiments involving 20 volunteers on three different types of keypads over 11 months show that our system can achieve 80% accuracy in revealing the user's PIN sequences with one try, and over a 90% success rate within three tries, while recovering the hand movement trajectory has a mean error as low as $6mm$. Our findings are an early and significant step to understand the possible security vulnerabilities of a wearable device's embedded sensors. Future countermeasures may aim at camouflaging the sensitive sensor data transmitted from wearables to host devices. For example, a wearable can inject a certain type of noise to the data so that the data cannot be used to derive fine-grained hand movements while still effective for fitness tracking purpose (i.e., activity recognition or step counts). Moreover, in the two attack models we discuss, more secure encryption schemes are necessary to protect the BLE communication, while accessing to sensor data should be regulated by the wearable or its host's operating system to avoid leakage.

Acknowledgments. This work is supported in part by the NSF grants CNS0954020, SES1450091 and Army Research Office W911NF-13-1-0288.

11. REFERENCES

[1] All about skimmers. http://krebsonsecurity.com/all-about-skimmers/.

[2] Is it acceptable to wear a watch on the right wrist? http://www.askandyaboutclothes.com/forum/showthread.php?116570-Is-it-acceptable-to-wear-a-watch-on-the-right-wrist.

[3] Malicious cloned games attack google android market. naked security:. http://nakedsecurity.sophos.com/2011/12/12/malicious-cloned-games-attack-google-android-market/.

[4] Wearable device shipments predicted to surge 173% this year. http://www.cnet.com/news/shipments-of-wearable-device-to-surge-173-this-year/.

[5] Why wear a watch on the wrist where you're hand dominant. http://www.reddit.com/r/Watches/comments/1wzub5/question_why_wear_a_watch_on_the_wrist_where/.

[6] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *IEEE S&P*, pages 170–183, 2008.

[7] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *ACM CCS*, pages 245–254, 2006.

[8] J. Liu, Y. Wang, k. Kar, Y. Chen, J. Yang, and M. Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *ACM Mobicom*, 2015.

[9] L. Liu and et al. Toward detection of unsafe driving with wearables. In *ACM WearSys*, pages 27–32, 2015.

[10] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang. When good becomes evil: Keystroke inference with smartwatch. In *ACM CCS*, pages 1273–1285, 2015.

[11] F. Maggi and et al. A fast eavesdropping attack against touchscreens. In *IEEE IAS*, pages 320–325, 2011.

[12] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, pages 551–562, 2011.

[13] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: your finger taps have fingerprints. In *ACM MobiSys*, pages 323–336, 2012.

[14] A. Parate and et al. RisQ: recognizing smoking gestures with inertial sensors on a wristband. In *ACM MobiSys*, pages 149–161, 2014.

[15] Y. Ren, Y. Chen, M. C. Chuah, and J. Yang. User verification leveraging gait recognition for smartphone enabled mobile healthcare systems. *IEEE Transactions on Mobile Computing*, 2014.

[16] M. Ryan. Bluetooth: With low energy comes low security. In *USENIX WOOT*, pages 4–4, 2013.

[17] M. Sherman and et al. User-generated free-form gestures for authentication: Security and memorability. In *ACM Mobisys*, pages 176–189, 2014.

[18] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha. Beware, your hands reveal your secrets! In *ACM CCS*, pages 904–917, 2014.

[19] D. Spill and A. Bittau. Bluesniff: Eve meets alice and bluetooth. In *USENIX WOOT*, pages 5:1–5:10, 2007.

[20] H. Wang, T. T.-T. Lai, and R. Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *ACM MobiCom*, pages 155–166, 2015.

[21] J. Wang, K. Zhao, X. Zhang, and C. Peng. Ubiquitous keyboard for small mobile devices: Harnessing multipath fading for fine-grained keystroke localization. In *ACM Mobysis*, pages 14–27, 2014.

[22] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *ACM WISEC*, pages 113–124, 2012.

[23] T. Zhu, Q. Ma, S. Zhang, and Y. Liu. Context-free attacks using keyboard acoustic emanations. In *ACM CCS*, pages 453–464, 2014.