Chaperone: Real-time Locking and Loss Prevention for Smartphones

Jiayi Chen and Urs Hengartner, *Cheriton School of Computer Science, University of Waterloo;* Hassan Khan, *School of Computer Science, University of Guelph;* Mohammad Mannan, *Concordia Institute for Information Systems Engineering, Concordia University*

https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jiayi

This paper is included in the Proceedings of the 29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX.

Chaperone: Real-time Locking and Loss Prevention for Smartphones

Jiayi Chen[1], Urs Hengartner[1], Hassan Khan[2], and Mohammad Mannan[3]

[1]*Cheriton School of Computer Science, University of Waterloo*
[2]*School of Computer Science, University of Guelph*
[3]*Concordia Institute for Information Systems Engineering, Concordia University*

Abstract

Smartphone loss affects millions of users each year and causes significant monetary and data losses. Device tracking services (e.g., Google's "Find My Device") enable the device owner to secure or recover a lost device, but they can be easily circumvented with physical access (e.g., turn on airplane mode). An effective loss prevention solution should immediately lock the phone and alert the owner *before* they leave without the phone. We present such an opensource, real-time system called `Chaperone` that does not require additional hardware. Chaperone adopts active acoustic sensing to detect a phone's unattended status by tracking the owner's departure via the built-in speaker and microphone. It is designed to robustly operate in real-world scenarios characterized by bursting high-frequency noise, bustling crowds, and diverse environmental layouts. We evaluate Chaperone by conducting over 1,300 experiments at a variety of locations including coffee shops, restaurants, transit stations, and cars, under different testing conditions. Chaperone provides an overall precision rate of 93% and an overall recall rate of 96% for smartphone loss events. Chaperone detects these events in under 0.5 seconds for 95% of the successful detection cases. We conduct a user study (n = 17) to investigate participants' smartphone loss experiences, collect feedback on using Chaperone, and study different alert methods. Most participants were satisfied with Chaperone's performance for its detection ability, detection accuracy, and power consumption. Finally, we provide an implementation of Chaperone as a standalone Android app.

1 Introduction

Smartphone loss is a serious security risk that has affected millions of users. News articles on such incidents are abound. In 2018, Kaspersky Lab [8] reported that on average, 23,000 Android devices are being lost or stolen each month. In 2016, half a million UK residents had a mobile phone stolen, and 35% of these phones were stolen while they were being left out and unattended [21]. Most stolen phones are never recovered—e.g., 68% US users failed to retrieve their phones in 2014 [11]. Users are more likely to lose their smartphones in public places, e.g., coffee shops and bars, where strangers can steal them [4]. In 2019, smartphones were the most commonly lost item in the ride-hailing service Uber [28]. Wiese et al. [32] observe that 49% of office workers put their phone unattended on a desk, which incurs unauthorized access of co-workers to sensitive data [24]. Beyond privacy threats, stolen or lost devices can also significantly affect enterprise security [2, 18, 25].

Many solutions have been designed to secure an unattended smartphone or its data. We term these solutions as *post-loss* solutions. Some solutions aim to prevent unauthorized access to the sensitive data stored on the unattended device. This goal is mostly achieved by locking the phone's screen after a configurable idle period. However, an adversary, like a co-worker, may be able to pick up the phone before it locks. Ideally, the phone screen should be locked as soon as its owner steps away. Proximity-based solutions [6, 19, 33] target this goal by making the owner carry an additional device, and use RFID or Bluetooth to detect proximity to the phone. However, these solutions do not provide a very accurate measure of distance [12]. Another alternative is continuous authentication [9], which tries to detect when a non-owner is using the phone, and subsequently locks the phone. However, it may fail in certain cases, in particular, against mimicry attacks [10].

Some other solutions assist with the recovery of lost devices. "Find My iPhone" and "Find My Device" are device tracking services available from Apple and Google, respectively. Once the device owner realizes that they have lost the device, they can use these services to locate, recover, or disable their smartphone. Usually there is some delay between the device loss event and the owner's realization of it. For devices lost in public places, this delay is sufficient for strangers to steal the device and turn on airplane mode to render such solutions ineffective. Therefore, a phone that is about to become unattended in a public place should try to prevent this loss by alerting its owner (e.g., playing an alarm sound), in addition to locking its screen.

The main challenge for a device locking and loss prevention solution is to make the phone track the user's departure in a

contactless way, where the phone senses the user's motion without the user carrying it. In addition, the solution should satisfy the following requirements: (1) detect device loss in *real-time*, i.e., the device must react *before* its owner has left the premise; (2) work on common off-the-shelf smartphones without requiring additional hardware or OS root privileges; (3) perform robustly across common device loss scenarios, e.g., noisy and crowded public places, offices, and cars; and (4) have sufficiently low error rates for everyday device usage.

Given that smartphones are equipped with at least a pair of microphone and speaker, they are capable of active acoustic sensing. Li et al. [13] proposed iLock to automatically lock a device based on the user-device distance estimated by the Frequency Modulated Carrier Wave-based sensing technique [1]. However, iLock was only evaluated in two relatively ideal environments: a lab and a library. Our experiments show that it fails to work reliably in some common scenarios due to environmental factors (see Figure 1, and details in §4). High-frequency noise, movement of nearby people, and the presence of obstacles may interfere with iLock's distance estimation and result in false positives. For the loss prevention scenario, a false positive results in an unnecessary alert, like an alarm sound, which not only annoys the phone owner but also nearby people. Thus it is critical to ensure a low false positive rate for loss prevention.

We present `Chaperone`, a real-time smartphone locking and loss prevention solution using active acoustic sensing. Chaperone focuses on capturing a user's departure patterns and addresses the aforementioned challenges by tracking the departure procedure of the device owner across three dimensions (in reference to the smartphone): the motion state of the owner, the intensity of the motion, and the distance of the owner from the device. By incorporating multiple factors, Chaperone provides a robust real-time mechanism to detect when the user is *about to* leave the premise.

Contributions.

1. We design and implement Chaperone, a standalone, active acoustic sensing-based system that detects possible smartphone loss incidents in real-time on commodity smartphones. Chaperone requires no per-user training to operate in a new situation. Although it needs access to the device's microphone and speaker, Chaperone's standalone nature preserves privacy of the device owner and bystanders, as our carefully designed implementation does not offload any computation from the smartphone.
2. We conduct 1,345 experiments to demonstrate Chaperone's ability to operate under different conditions (including device orientations and positions, user leaving speeds, distances to nearby stranger, close objects, and concurrent sensing by multiple devices), and cover various real-world scenarios characterized by high-frequency ambient noise, crowded locations, and diverse layouts (including academic venues, restaurants, offices, cars, and transit stations). This is the first such comprehensive evaluation of active acoustic sensing in real-world scenarios compared to existing literature [3, 13, 15, 16, 22, 23, 27, 30, 36–40].
3. Chaperone provides an overall precision of 93% and an overall recall of 96%, outperforming iLock [13] (see §6 for details) by 14% in both precision and recall scores. Specifically, in complex real-world scenarios (e.g., lounge and bus stop), the performance gain is up to 32% in the recall score. For 95% of the successful loss detection experiments, Chaperone can lock the phone and alert the owner within 0.5 seconds. The experimental results provide strong indication that Chaperone is robust and effective in many everyday scenarios.
4. We conduct a user study (n = 17) to investigate people's smartphone loss experiences, collect feedback on using Chaperone, and study user perceptions of different alert methods for smartphone loss prevention. The results indicate that the participants are satisfied with the detection performance of Chaperone. We also report on the suitability of five alert methods for different locations.
5. We release Chaperone as an opensource, standalone Android app, and our collected dataset from both lab and real-world experiments, to help reproduce our findings, and improve acoustic sensing-based device loss prevention solutions. The project link is `https://github.com/cryspuwaterloo/chaperone`.

2 Related Work

Smartphone loss detection. Academic (e.g., [7]) and commercial (e.g., [19, 33]) solutions are available that require an additional device to detect proximity to the smartphone. Despite the overhead cost of additional hardware, these solutions do not provide a very accurate measure of distance [12]. Consequently, they may not be effective when the user leaves e.g., a ride without their smartphone. Yang et al. [34] propose *Surround-See*, a smartphone equipped with an omnidirectional camera that enables peripheral vision. One suggested application is warning users when they leave their phone behind. However, such special purpose cameras are unavailable on current smartphones. Mirsky et al. [17] study the scenario where an attacker picks up an unattended phone and starts using it. They show that within seven seconds, continuous authentication can detect the change in behaviour and lock the phone. However, the attacker may be able to mimic the owner's behavior [10] to prevent the phone from locking. In comparison, when the owner leaves, Chaperone can detect and lock an unattended phone within 0.5 seconds.

Liu et al. [14] focus on detecting pickpocket and grab-and-run phone theft events with machine learning. Their solution is limited to these two theft events and does not address unattended phone scenarios, making it complementary to Chaperone. Yu et al. [35] present a post-loss solution that uses

emergency call mechanisms to allow the device owner to wipe their device remotely after a loss. This solution works even if a thief removes the SIM card from the device. However, the solution is not designed to prevent the physical loss of the device. In terms of methodology, more close to our work is iLock by Li et al. [13], which uses active acoustic sensing for automated locking of the device. We conduct extensive experiments to show that Chaperone performs better in most real-world scenarios than iLock (see §7.1 and §8).

WiFi and acoustic sensing on smartphones. WiFi signals have been used to recognize human activities by detecting changes in the channel state [29, 31, 41]. However, it is difficult to extract WiFi channel state information on commodity smartphones [41]. WiFi-based approaches also require separate sender and receiver devices, and impose placement requirements, which makes them infeasible for loss detection in public places. In contrast to WiFi sensing, acoustic sensing works reliably using only a single device. In active acoustic sensing, a device generates a sound signal and senses the echo [3]. There has been extensive work on performing a variety of acoustic sensing tasks with commodity off-the-shelf smartphones; such tasks include: ranging [13], gesture tracking or recognition [23, 30, 36, 40], object detection [16, 27, 38], and user authentication [39]. However, no past approach has explored the feasibility of active acoustic sensing for smartphone locking and loss prevention, considering diverse background noise, crowd, and location layout conditions.

3 Threat Model

Our focus is the threat posed to an unattended smartphone by nearby opportunistic attackers. To start with, the smartphone is placed stationary on a surface intentionally (e.g., the owner puts it on a table), or unintentionally (e.g., the phone slips from the owner's pocket). Its microphone and speaker are not covered by other objects so that the transmission of sound is not blocked. (We examine the impact of nearby objects that partially block sound transmission in §8.3.1.) We assume the device owner is initially closer to the phone than others, including nearby people and the attacker, and the initial distance between the owner and the device is under 1m. This condition ensures that the device is initially in a relatively *secure* context compared to the later unattended status. We discuss more complicated situations in §10 (e.g., when a stranger is closer to the device than the owner).

After the initial placement, the owner may move away from the device, thereby exposing it to theft or unauthorized access. The attack may happen within a few seconds after the phone becomes unattended (i.e., when the owner moves away from the phone). A potential smartphone loss is defined as a smartphone owner leaving the phone behind in a public or untrusted place. We propose a preventive approach that can detect a potential smartphone loss situation, lock the phone, and generate an alert before the owner leaves the place. More than just putting a threshold on the distance from the smartphone, our approach detects the owner's departure and absence from the phone (i.e., the owner keeps moving away from the phone and is eventually absent). Therefore, in our experiments, we do not have a specific attacker role given that the detection should occur *before* the attack happens. Instead, we consider the influence of nearby people on our sensing approach, which captures the reflected signals from the owner (see §5) and other people and objects. Note that we use the terms *owner* and *user* interchangeably.

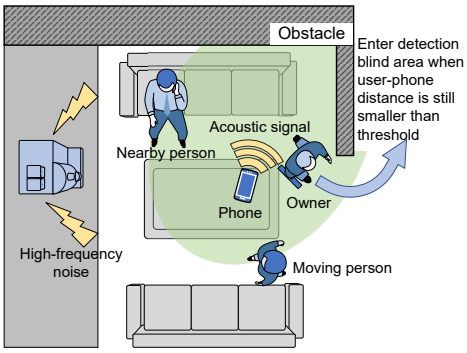


Figure 1: Potential factors that affect acoustic sensing. The green area depicts the detection range. The smartphone owner enters the detection-blind area caused by the obstacle while still being within the distance threshold, making the detector fail to follow the owner and track a nearby person instead.

4 Design Goals

An effective smartphone locking and loss prevention solution should have the following desirable properties:

Standalone. While it is possible to leverage specialized hardware (e.g., Surround-See [34]), a solution that works on common off-the-shelf smartphones is more likely to be adopted. Similarly, while an accessory (e.g., a smartwatch) connected to the smartphone can detect smartphone loss, a standalone solution relieves users from carrying an additional device.

Low detection delay with low energy consumption. We use the term detection delay to refer to the time period during which the owner is unaware of the device loss. For post-loss solutions, this delay may be large as they are dependent on the owner's realization of the device loss. In a loss prevention solution, the detection delay corresponds to the time duration between the device owner leaving, and the solution realizing that the owner is not present near the device, in turn, locking the phone. Thus, it is desirable to have low detection delay. However, low detection delay requires frequent sensing to ensure real-time detection. The local analysis of the acoustic data on the mobile device could be computationally intensive and consume significant battery power. Thus, we need to balance detection performance and energy consumption.

Few false positives and false negatives. A closely related

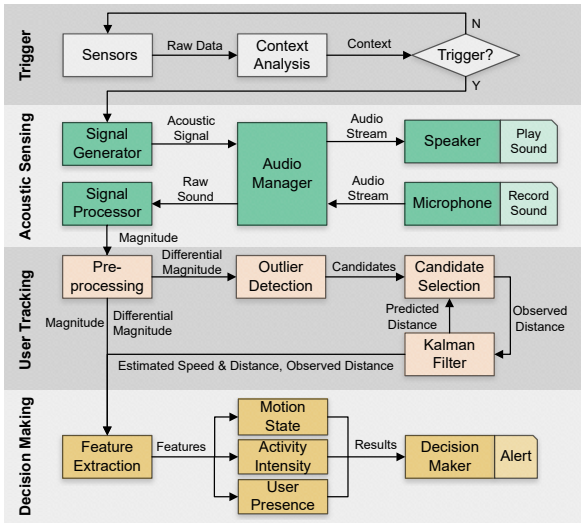


Figure 2: Workflow of Chaperone.

usability aspect is the number of false positives. For example, the smartphone owner may move to grab something from across the table in a restaurant, which may be misconstrued as the owner leaving by a solution with low detection delay. False positives are inconvenient and may negatively affect the adoption of a solution. Therefore, the solution should notify the user in real-time, while limiting the number of false positives. Similarly, the solution should have few false negatives, i.e., failure to detect actual user leave events. False negatives may cause device loss; therefore, the system should minimize false negatives even at the cost of increasing false positives.

Robust. In practice, smartphones are lost at a variety of locations including coffee shops, restaurants, cars, etc. [4, 28]. Location diversity implies different levels of background noise, nearby moving people, and obstacles in the physical layout of the location. Figure 1 shows an example of these factors in a small lounge scenario. In terms of background noise, active acoustic sensing for smartphones usually uses the high-frequency band up to 24kHz (see §5), and as a result, high-frequency background noise poses a threat. Such noise is often encountered in real-world scenarios, e.g., slamming of a door. A high-frequency noise source may emit noise for a short period of time, but it is likely to happen more often at certain locations (e.g., a restaurant). Therefore, it is important for a robust system to deal with high-frequency noise. The movements of other nearby people introduce more reflections of sound signals, and thus require careful consideration. In terms of layout, a location's physical layout may introduce obstacles, limiting the effective operational range of acoustic sensing. In Figure 1, the range where active acoustic sensing can effectively receive the echoes is limited by the lounge layout since the acoustic signal is blocked or reflected by the obstacles. If the owner follows the blue arrow, the phone fails to track the echo from the owner after the owner moves behind the obstacle. In summary, the solution should robustly

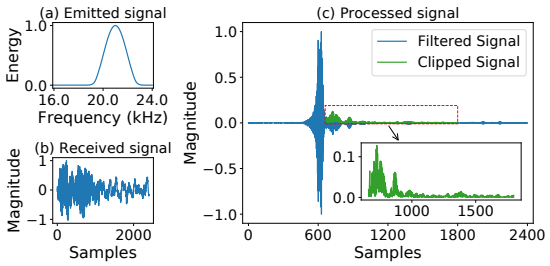


Figure 3: Signal processing in the acoustic sensing module (note: magnitude in the figure is normalized).

operate across a variety of locations, and require minimal or no location and environment-specific tweaking.

5 Chaperone: Design and Implementation

We leverage active acoustic sensing based on a high-frequency acoustic signal, which is inaudible to most humans and is not interfered by common noise in the lower-frequency band. The speed of sound is orders of magnitude greater than the speed of a person moving away from the device—sufficient for real-time detection. Chaperone consists of four main modules: trigger, acoustic sensing, user tracking, and decision making module; see Figure 2.

5.1 Trigger Module

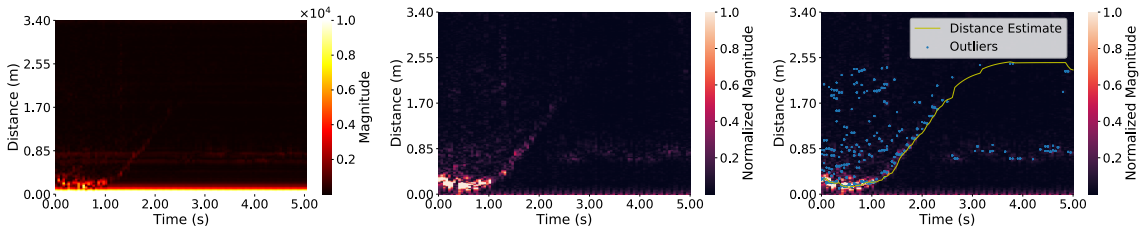
Chaperone does not need to continuously perform active acoustic sensing for many scenarios including the following: (1) The user is holding the device, or it is on the user's body. Google's Activity Recognition API provides this information using low-power sensors.[1] (2) The user is using the device while it is lying on a surface, e.g., playing a video while the device is on a desk. This can be determined by querying the device state to establish whether the device screen is off and it is in idle state. (3) The device is at a *trusted* location, e.g., the user's home; such locations can be configured by the device owner. The trigger module invokes active acoustic sensing only when the device is not in use (i.e., idle), not on the user's body, and in a potentially *untrusted* or *public* environment. This reduces the acoustic sensing overhead.

5.2 Acoustic Sensing Module

This module performs active acoustic sensing to keep track of the user's movement. It sends a particular acoustic signal, and processes the received echo signal to make meaningful conclusions about the user's movement (if any). It consists of an acoustic signal generator, audio manager (controlling the speaker and the microphone), and a signal processor.

The signal generator produces an inaudible acoustic signal based on sampling rate, frequency, length, and signal type,

[1] https://developers.google.com/location-context/activity-recognition



(a) Magnitude heatmap. (b) Differential heatmap. (c) Outliers and distance estimate.

Figure 4: Distance estimation procedure: (a) the bright part represents the captured echoes from nearby objects and people; (b) after excluding echoes from static objects, the user's movement from time 0–2.5s is highlighted, but we can still observe the echo from nearby people, e.g., 85cm away from the device during the time period 2.5–5s; (c) by using our candidate selection algorithm, we can track the user's movement and predict the movement when there is no valid observation (e.g., at time 3s).

and then passes the audio data to the audio manager. We use a sampling rate of 48kHz for supporting the acoustic signal up to 24kHz and a sensing period (i.e., a frame) of 50 milliseconds for real-time detection. In the first phase of the sensing period, the device emits a 1,200-sample acoustic signal and keeps recording the sound; see Figure 3a. In the following 1,200-sample idle phase, the device emits no signal but continues to sense for the reflection of the signal emitted during the first phase. The default acoustic signal used in Chaperone is a frequency sweep from 19–23kHz with fading at the start and the end of the signal, which is inaudible to most humans [5].

The audio manager interfaces with the smartphone's speaker and microphone. It simultaneously uses the speaker to periodically play the acoustic signal and the microphone to record the sound; see Figure 3b for an example of the raw sound. Since the recorded sound covers the whole frequency range, including environmental noise, the audio manager continuously passes the raw sound data to the signal processor to extract the reflected acoustic signal.

The signal processor is designed to obtain a magnitude vector $\mathbf{m}$ of the echoes. It first applies two filters, a band-pass filter and a matched filter, to the raw sound data to match the original acoustic signal. The band-pass filter keeps the dedicated frequency band, and the matched filter highlights the original acoustic signal by calculating the convolution of the filtered sound signal and the reversed original acoustic signal. Since it is impossible for an echo to occur before the direct transmission, we only keep the samples after the first peak (i.e., the sample with the locally highest magnitude caused by the direct transmission from the speaker to the microphone), and then obtain the processed acoustic data; see Figure 3c. The signal processor then calculates the magnitude vector $\mathbf{m}$ for the clipped signal. Since the delay of an echo is the round-trip time of sound traveling between the phone and an object (or user), each index of the vector can be mapped to the corresponding distance d according to the following time-of-flight distance measurement formula: $d = \frac{Mc}{2f_s} \cdot i$, where c is the speed of sound, f_s is the sampling rate of the acoustic signal and M is the downsampling rate. For example, given that $c = 340$ m/s, $f_s = 48$kHz and $M = 4$, the 10th element of vector m is the magnitude of the matched signal that is approximately 0.142m away from the phone. Finally, the signal processor passes the magnitude vector $\mathbf{m}$ for the current frame to the user tracking module.

5.3 User Tracking Module

This module locates the user by filtering echoes reflected from surrounding objects and background noise, and tracking the user among other moving bodies.

Pre-processing. In the first step, the pre-processing sub-module filters the echoes reflected from other objects. Figure 4a shows that the magnitude vectors capture echoes from the user as well as objects. We remove echoes from static objects by using the differential magnitude vector $\Delta\mathbf{m}_t = \mid \mathbf{m}_t - \mathbf{m}_{t-1} \mid, t \in \mathbb{N}^*$, which is the absolute difference between the current and the previous magnitude vectors. Figure 4b shows that this step excludes static objects and highlights echoes from the user. The pre-processing sub-module also determines if the current frame is affected by background noise. The overall magnitude of the differential magnitude vectors at the corresponding moments may become irregularly large due to high-frequency noise (see §4); we thus set a threshold on the average value to exclude such noisy frames. Note that if a frame is regarded as noisy, there is no valid observation at that moment. This error is adjusted by predicting the current distance based on the values from the previous frames using a Kalman filter.

Outlier detection. This sub-module detects potential dynamic movements of the user. Intuitively, an outlier (i.e., an exceptionally large magnitude) in a differential magnitude vector implies the existence of motion at the corresponding distance. We use median-absolute-deviation (MAD) outlier detection to obtain the outliers in the current frame. However, our outlier detection may be negatively affected by the motion of the user's body parts and the motion of other nearby people. Specifically, the intense motion of a user's body parts results in a non-trivial number of outliers; see the blue dots in Figure 4c. We handle these outliers by clustering them based on their relative distance, so that they are merged into a single

Algorithm 1 Candidate Selection Algorithm

Input: All m candidate tuples $C_m = \{(s_m, h_m, l_m)\}$ where s is starting distance, h: peak magnitude, l: cluster size; $\hat{d}$: predicted distance ; n history speeds $\tilde{\mathbf{v}} = \{v_0, v_1, \ldots, v_{n-1}\}$; $R_{\max}$: max range; q: base discount

Output: Observed distance *obs*

function CANDIDATE_SELECTION($C, \hat{d}, \tilde{\mathbf{v}}$)
  $obs \leftarrow -1, p_{\max} \leftarrow -1, e \leftarrow 0$ ▷ Initialization
  $\kappa_0 \leftarrow$ `getDirection`(v_{n-1})
  for $i \leftarrow n-2$ to 0 **do**
    $\kappa \leftarrow$ `getDirection`(v_i)
    if $\kappa = \kappa_0$ and $\kappa \neq 0$ **then** ▷ If direction changes
      $e \leftarrow e+1$ ▷ Add to discount exponent
    else break
  for $i \leftarrow 0$ to $m-1$ **do**
    $s_i, h_i, l_i \leftarrow C_i$, $r \leftarrow q^e R_{\max}$ ▷ discounted range r
    if $|\, s_i - \hat{d} \,| \leq r$ or $|\, s_i + l_i - \hat{d} \,| \leq r$ **then**
      if $h_i > p_{\max}$ **then**
        $obs \leftarrow s_i, p_{\max} \leftarrow h_i$
  return *obs*

candidate.

Candidate selection and Kalman filter. From the clustered candidates, we choose the candidate that corresponds to the user and use it to estimate the user-device distance and the user's speed. For the first frame (at $t = 1$), we choose the candidate closest to the phone, assuming that the user is the closest, and then feed the corresponding distance into the Kalman filter as the initial distance. Once the user is in motion, our assumption that the user is closest to the device may no longer be valid. For example, in Figure 4c, we can observe movement of another person at the distance mark of 0.8m (and at time 2.5s), while the user is actually 1.7m away from the phone. To address this scenario, we make the candidate selection and the Kalman filter work together to decide which candidate point to choose as the observed distance d_t at time t. The Kalman filter is also used to estimate both distance and speed. For candidate selection among the following frames, we reduce the candidate selection range if the user keeps the previous motion state; see Algorithm 1.

Since the Kalman filter itself predicts the current distance and speed at each round, we incorporate the a priori estimate of distance $\hat{d}_{t|t-1}$ (i.e., "predicted distance") from the Kalman filter to calculate the possible range for the next distance. The candidate selection module chooses the most consistent candidate based on the magnitude and uses its corresponding distance as the observed distance. Then, the Kalman filter updates the a posteriori estimate of distance $\hat{d}_{t|t}$ and speed $\hat{v}_{t|t}$ at time t. (We denote them as "estimated distance" and "estimated speed".) Note that if the user is stationary, or out of the detection range, there might be no matching candidate points. In that case, the Kalman filter is fed with the previous distance as the observation, assuming that the user is idle. After combining multiple frames, we obtain a trace of the user's movement based on the distance estimated by the Kalman filter (the yellow line in Figure 4c). All the distance and speed values, together with the magnitude vectors, are passed to the decision making module to determine whether to generate an alert.

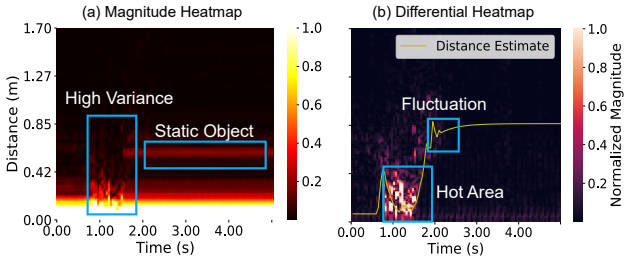


Figure 5: Example of distance tracking failure: the user tracking module can only track the user up to about 85cm.

5.4 Decision Making Module

This module detects whether the user is about to leave the device, based on the information obtained from the acoustic sensing and user tracking modules. As noted in §4, several environmental factors can limit the detection capabilities in real-world scenarios; see Figure 5, where the user tracking module fails when a distance-only approach is employed with the distance threshold set at 1m. As a result, a simple distance-only approach is unable to determine whether the user is stationary at that point or is behind the wall. Therefore, dealing with obstacles requires a more comprehensive analysis than relying on the estimated distance alone.

Classifiers for user state estimation. We rely on three classifiers: the motion state classifier determines whether the user is approaching, leaving, or stationary; the activity intensity classifier determines whether the user's activity is intense or moderate; and the user presence classifier determines whether the user is close to the device or far away. The features for these classifiers are derived from distance, speed, magnitude vector and differential magnitude vector estimates of the user tracking module; Table 1 lists our features and their usage in the classifiers. Feature values are populated by combining data from multiple continuous frames into one window. The window size w is set to five frames (i.e., 250ms), containing sufficient information to perform meaningful analysis without affecting the real-time capability of Chaperone. Within each window, we denote the first frame as t_1 and the last frame as t_w. As for the (differential) magnitude vectors, we focus on movements in the 15cm–1m range. A lower bound of 15cm excludes any direct transmissions from the speaker to the microphone, and our experiments show that an upper bound of 1m provides sufficient data to reliably detect smartphone loss.

Features for classification. Intuitively, speed and distance features are correlated to the user's motion and presence state. From the user tracking module, we know whether it has a

Features	Formula or Description	C1	C2	C3
IsObserved	whether user tracking module makes a valid observation	●	●	●
Distance	Observed distance: $d_{\text{obs}} = \sum_{i=t_1}^{t_w} d_i/w$ Estimated distance: $d_{\text{est}} = \sum_{i=t_1}^{t_w} \hat{d}_{i\|i}/w$ Difference from median: $\Delta d_{\text{est}} = d_{\text{est}} - \text{median}\{d_t\}$ $\Delta d_{\text{est}} = d_{\text{est}} - \text{median}\{d_{t\|t}\}$	●	●	●
Speed	Observed speed: $v_{\text{obs}} = (d_{t_w} - d_{t_1})/w$ Est. speed: $v_{\text{est}} = (\hat{d}_{t_w\|t_w} - \hat{d}_{t_1\|t_1})/w$ Numerical avg. speed: $\bar{v} = \sum_{i=t_1}^{t_w} \hat{v}_{i\|i}/w$	●	●	
Fluctuation	# of direction changes in est. speed	●	●	
Magnitude	$\bar{m} = \sum_{i=t_1}^{t_w} \sum_{j=d_0}^{r} \Delta\mathbf{m}_{i,j}/wr$	●	●	●
Hot area rate	$h = \sum_{i=t_1}^{t_w} \sum_{j=d_0}^{r} \mathbb{1}\{\Delta\mathbf{m}_{i,j} > \theta\}/wr$	●	●	●
Row variance	$\sigma_d = \sum_{i=t_1}^{t_w} (\mathbf{m}_{i,d} - \mu_d)^2/w$, $\mu_d = \sum_{i=t_1}^{t_w} \mathbf{m}_{i,d}/w$		●	●
Static object	# of static objects changed in $\mathbf{m}$			●

Table 1: Features for three classifiers. C1: motion state; C2: activity intensity; C3: user presence. A circle means a classifier uses the corresponding feature (empty indicates no use).

valid observation on the user's motion, and then we can obtain both observed and estimated distances and speeds. We also calculate the relative distance to the median of historical user-device distances, approximating the user's initial distance to reduce fluctuations caused by the user's activity. Besides, we employ the average speed, which is the slope of the line connecting the distances of the first and the last frames.

We also consider intensity-related features. Figure 5b shows that when the user is performing activities, such as typing or standing up, the movement of different body parts leads to the average differential magnitude close to the phone being dramatically larger (called a "hot area") than the ambient magnitude. Therefore, to describe the user's activity intensity, we use the average differential magnitude and the hot area rate, the proportion of the area whose magnitude is larger than a threshold θ. Besides, these activities may result in some fluctuations in the speed and distance estimation, which can be observed in frequent changes of the direction.

The magnitude vector also provides information about user presence; see Figure 5a. Even slight movement of the user can still cause an increase of variance in magnitude at the corresponding distance, implying the user's presence. Furthermore, it is possible to infer the user's presence based on the static objects nearby. When the user is near the phone, parts of the acoustic signal will be blocked by the body, and the objects behind the user may not appear on the spectrum. But after the user has left, these objects will begin to reflect the signal, and thus change the raw magnitude vector.

Decision maker. This sub-module determines the user state, and reacts based on the classification results of the three classifiers. We adopt a sliding window mechanism to make a decision across three windows, which improves the detection accuracy without sacrificing the real-time nature of the system. The decision maker uses the following criteria to decide whether a departure activity of the user happens: The user is leaving (i.e., the motion state classified as "leaving"), the activity intensity is fading (i.e., the activity intensity changed from "intense" to "moderate"), and lastly the user is no longer close to the device (i.e., the user presence state changed from true to false). Only when the user's movements satisfy all criteria, Chaperone will make a positive detection. This strategy helps reduce false positives by a distance-only approach.

As a reaction to a potential smartphone loss, Chaperone locks the phone immediately and triggers an appropriate alert method using, e.g., a ringtone, vibration, notification sound, or screen flashing. The alert scheme is chosen based on the contextual information collected by the trigger module. For example, if the environmental noise level is low, a gentle ringtone will be sufficient to get the user's attention. In §9, we systematically investigate user preferences for alert methods in terms of effectiveness and annoyance in different scenarios.

5.5 Implementation

We implement a Chaperone prototype as a standalone Android app. To help reproducibility, we also implement a remote-mode option, where the smartphone is responsible only for acoustic sensing, and a remote server stores and analyzes the raw acoustic data for user tracking and decision making.

For acoustic sensing, we use LibAS [26], an opensource framework for the rapid development of acoustic sensing apps. LibAS outputs the acoustic signal used by Chaperone and performs acoustic sensing. The operations required for user tracking and decision making (see Figure 2) are not provided by LibAS, so we had to implement them ourselves. The minimum SDK supported by Chaperone is API level 21. Audio data is collected in raw audio mode for Android 7 and up or using the microphone audio source for below Android 7.

Support for different smartphones. For most experiments, we use a Google Pixel (2.15GHz quad-core CPU, 2016) for data collection to train the classifiers in the decision making module. We successfully tested the prototype on Samsung S8, Huawei AL-10, and Google Pixel, Pixel 3, Nexus 5x, and Nexus 6P phones. Because of hardware differences, the magnitude scales of acoustic signals vary on different devices. To make Chaperone work on different devices, an additional configuration step is needed. First, we adjust the volume of the target phone to approximate the original acoustic signal strength to the Pixel. Then, we sample the received signal and map the magnitude scale of the target phone to it. This one-time configuration step is needed before deployment so that the classifiers can be used on other devices without retraining.

Latency. To balance detection performance and signal pro-

cessing overhead, we set the sensing period to 50 ms (see §5.2) and implement filters in native C for efficiency. It takes 25–35ms on the Pixel to generate raw magnitude vectors from the acoustic signal. User tracking considers echoes only within two meters from the device, which is sufficient for device loss detection, and takes less than 1ms to extract features. The decision making module uses pre-trained models and takes about 1–2ms for classification (see §6 for details). As a result, the overall latency of Chaperone for each sensing period is about 45ms on the Pixel. On the Nexus 5x (1.8GHz hexa-core CPU, 2015), processing takes 60ms, while on the Pixel 3 (2.5+1.6GHz octa-core CPU, 2018), it takes only 35ms. Therefore, Chaperone is effective for new and old devices.

Silent mode. When acoustic sensing is triggered on a device in silent mode, the media volume is set to high for exclusively sending inaudible acoustic signals. The ringtone volume remains on silent. Since silent mode implies that the user is in a quiet environment, Chaperone can adopt vibration or flashing for alerts, instead of a ringtone. When acoustic sensing is terminated, the device resumes the normal silent mode.

6 Evaluation Setup

Logistics. To evaluate the detection performance of Chaperone, we conducted experiments that simulated different smartphone loss scenarios. For the ground truth, we need labelled acoustic data that indicates when a user is at a certain distance from the device. This requires at least an experimenter and an observer. The experimenter acted as the device owner and performed a series of departing and everyday activities. We include scenario-specific everyday activities as they may introduce false positives (see §8 for details). The observer was responsible for real-time labelling of the departure events, t_d, and absence events, t_a. The departure event indicates that the experimenter is leaving the device, and the absence event indicates that the experimenter is 1m away from the device. The observer also labeled the user state information, which is used for the model training for the three classifiers. In total, eight experimenters (one undergraduate student and one graduate student who have no security background, six graduate students who have security background) simulated the device loss events in the experiments and one observer labeled the events for consistency.

Data collection. Our objective is to collect data from a diverse set of evaluation conditions and scenarios. We first controlled device orientation and the user's departing speed in lab experiments. Intuitively, when the microphone is facing the user, the echo reflected from the user is most effectively captured. But if a user puts the phone horizontally (i.e., 90°) on a table, the received echo signal is likely weak. As for departing speed, if the user departs quickly, the system's reaction time may be inadequate for real-time alerts. We collected 135 departure and 135 everyday activity events from an experimenter to evaluate nine combinations of these conditions (see §7.1). Another aspect that requires careful control is the effect of a nearby stranger on Chaperone—e.g., whether the departure of a nearby stranger results in a false positive, or the existence of the stranger when the user has departed results in a false negative. We collected 54 user-departure events and 54 nearby stranger-departure events with an experimenter and a stranger separated by three distances in a lab-based setup (see §7.2). Finally, we evaluated real-world conditions with varying factors (e.g., crowd, noise, and physical layout) at eight locations (library, office, restaurant, coffee shop, lounge, bus stop, in-vehicle, and academic venue). Eight volunteers helped to collect 366 departure events and 391 idle events; see Table 2. We comment on the environmental conditions of each location when we present the results in §8. In addition, we evaluated the effects of other interference factors by collecting 75 departure events in close-object experiments and 135 departure events in concurrent sensing experiments.

Each data collection experiment consists of two parts. In each scenario, the experimenter put the phone on a surface (e.g., dining table at a restaurant) within one-arm distance from the body. In the first part, the experimenter performed some everyday activities matching the given scenario. In the second part, the experimenter left at the requested speed. Each activity is about 2.5–10 seconds long. For layouts with multiple departing paths, the experimenter also took different paths. The observer was far away from the experimenter (more than 2m for lab experiments, at least 1m for real-world experiments) to capture the departing procedure. Finally, to measure the performance of Chaperone over longer idle periods, we collected 15–20 minutes of data in locations where the user stayed for a long time, such as libraries, meeting rooms, or restaurants. For these experiments, we count the total number of false positives in the given time duration.

Algorithms for comparison. We compare Chaperone with iLock's user-phone distance estimation approach [13]. We contacted the iLock authors for their implementation. Although we did not receive it, they provided implementation details missing from the paper. Combining with details from the related papers [1, 13], we implement iLock's distance estimation approach including background subtraction, peak finding, and a Kalman filter with outlier rejection. Given the available details, our implementation is close to the one by Li et al., although there may be minor differences. We label this algorithm as "iLock" for simplicity. We assume the phone will be locked and an alert will be raised whenever the estimated distance exceeds the threshold of 1m, as set by iLock [13]. iLock is prone to raise a positive detection for more involved scenarios to avoid false negatives. For example, when more than two users' movements are detected but only one exceeds the threshold, iLock locks the device without knowing whether it was the owner who crossed the threshold; this causes many false positives in the real-world experiments (see §8). To reduce false positives, we merge the candidate

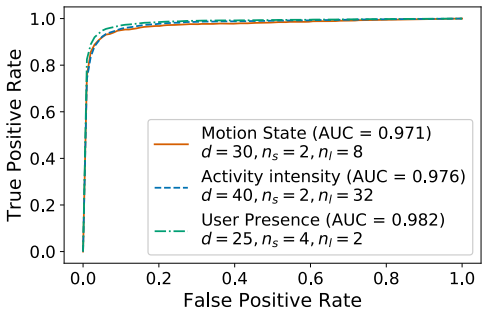


Figure 6: The ROC curve of the three classifiers.

selection strategy from Chaperone into iLock, which we label as "iLock++". This change improves the peak selection of iLock to better track the owner's movement.

Metrics. To evaluate the detection performance, we use precision and recall. We denote a departing activity as a *positive instance* and an idle activity as a *negative instance*. We define a successful detection as one made after the moment t_d when the user starts to leave. Precision is the fraction of successfully detected departing activities in all the positively detected ones, while recall is the fraction of successfully detected departing activities in all the positive instances. Note that if a positive detection is made before t_d due to false tracking, it is counted as both a false positive and false negative (i.e., it creates a false alarm and fails to detect the true event). We also evaluate the time delay of the alerts. We use human observations as reference points and correct them based on acoustic sensing to offset the human reaction time. An alert is deemed valid if it is sent after t_d. We use the moment t_a when the user is observed to pass the 1m line as the zero-point for calculating the delays. Then the detection delay can be calculated as $\hat{t_d} - t_a$, where $\hat{t_d}$ is the time when Chaperone detects the departure. A negative delay means an early detection before the human observation of the user passing the 1m line.

Hyper-parameter tuning. Our three classifiers are responsible for interpreting the user's current status from a variety of features. The performance of the classifiers is critical for the final decision making. Therefore, it is necessary to tune the hyper-parameters of these classifiers before conducting the experiments. We adopt Randomized Search Cross Validation [20] to tune the three hyper-parameters of the Random Forest algorithm: tree size d, minimum sample number for splits n_s, and minimum sample number of each leaf n_l. We use the lab experiment dataset for tuning, and manually label 6,118 data points (i.e., windows) with the current user state. This dataset is also used to train the model used in real-world experiments. The tuning objective is to maximize the area under the receiver operating characteristic curve (AUROC). Figure 6 shows the average ROC curve of 20-fold cross-validation with the best hyper-parameter settings for the three classifiers. In the following experiments, we always adopt the hyper-parameters for model training listed in Figure 6.

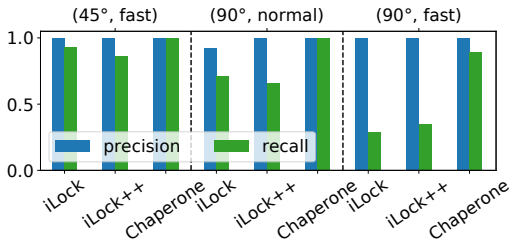


Figure 7: Precision and recall of iLock, iLock with Chaperone's candidate selection strategy (iLock++), and Chaperone.

7 Lab Experiments

7.1 Device Orientation and Departure Speed

We conducted experiments on nine different combinations of the following two factors—three phone orientation angles: 0° (vertical), 45°, and 90° (horizontal); and three departure speeds for the experimenter: *slow*, *normal*, *fast*. The logged departing speeds were experimenter dependent, and the average speeds were 1.25m/s (slow), 1.67m/s (normal), and 2.22m/s (fast). These experiments were conducted in a lab with a 70cm high desk. For each experiment, the phone was placed at the given angle on the desk in front of the experimenter and the experimenter stood at the desk about 20cm away from the phone. For each angle-speed combination, we logged 15 departure and 15 idle events.

Since Chaperone requires training the three classifiers, we use ten-time four-fold cross-validation to evaluate its detection performance. Namely, we split the data for all combinations into four subsets where data samples from different combinations are evenly distributed. We use three subsets for training and the fourth one for testing. The splitting is repeated for ten times, and eventually, we calculate the average precision and recall for each angle-speed combination. For iLock and iLock++, which are model-free, we directly evaluate their performance over each combination.

For angle-speed combinations (0°, fast/normal/slow), (45°, normal/slow), (90°, slow), all three algorithms achieved both 100% precision and 100% recall. Figure 7 shows the evaluation results for the three algorithms under the other three combinations. Even when the user departed at a fast speed and the phone orientation angle was 90°, the precision and the recall of Chaperone are 100% and 89%, respectively—a strong indication of robustness against different phone orientation angles and departure speeds. In comparison, if the user left at a normal or fast speed and the phone orientation angle was 90°, the recall scores of iLock and iLock++ decrease significantly. For the (90°, fast) combination, the recall score of iLock drops to only 29%, and iLock++'s is about 35% with successfully tracking two more departing activities based on the improved tracking strategy. The reason for the drop is that the strength of echoes from the user becomes weaker when the angle is larger, and the detection window is reduced due to the fast departing speed, where few valid measurements can be made by

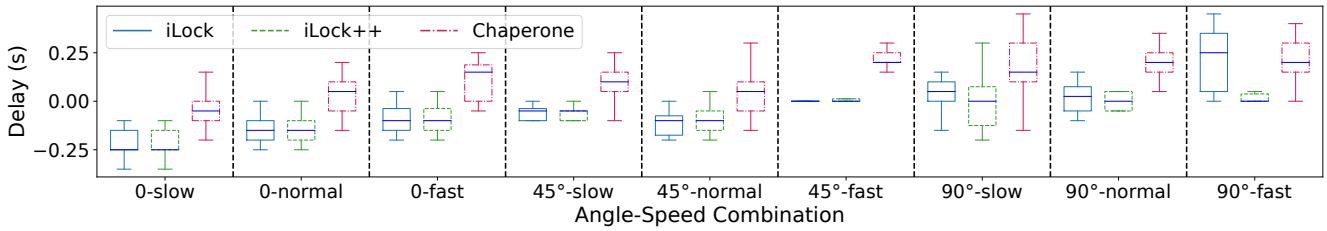


Figure 8: Detection delay (in seconds) for three algorithms.

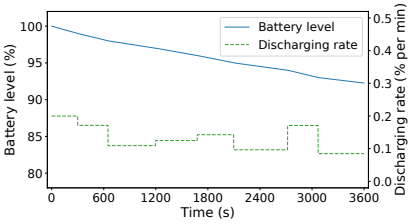


Figure 9: One-hour energy consumption when Chaperone is continuously sensing.

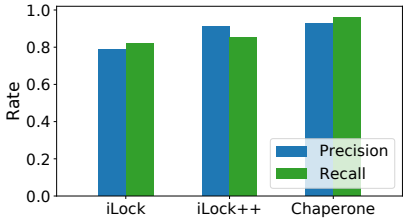


Figure 10: Overall performance of three algorithms across all locations.

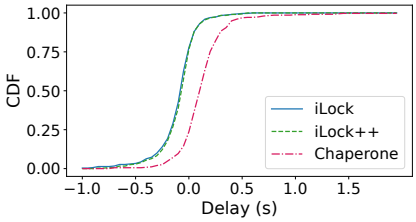


Figure 11: Detection delay. Negatives indicate detection before crossing 1m.

iLock and iLock++. They lost track of the user's trace before the 1m threshold under these conditions, which shows the ineffectiveness of the distance-only approach. Chaperone can still detect such situations based on the user state classifiers of the decision making module. All three algorithms have a very high precision score (i.e., no false positives for iLock++ and Chaperone), because the idle events performed in the ideal experiments were always close to the phone, which are easy to differentiate from a departure event. The experiment has shown that Chaperone, with the help of both the user tracking and decision making modules, outperforms iLock and iLock++ when handling a more complicated situation.

Figure 8 shows the detection delay for the three algorithms. Ideally, we expect an alert to be emitted within 1–2 seconds after leaving the phone to get the user's attention on time, i.e., while the user is still close. Chaperone consistently reacts within 400ms (95th percentile) for all nine combinations after the user passed the 1m line; in contrast, iLock and iLock++ can react within 200ms due to their simpler detection strategy. Chaperone's window-based decision mechanism incurs a delay of 400ms, but it is still fast enough for real-time use.

In summary, Chaperone performs significantly better under several angle-speed combinations. Both iLock and iLock++ perform poorly when the orientation angle is large and the user's departing speed is high. Chaperone handles this situation well by tracking the user's departure pattern instead of relying on user-phone distance only. All three algorithms manage to detect departure events in real time.

7.2 Effects of a Nearby Stranger

We conducted controlled lab experiments to investigate how a nearby stranger affects the detection performance. We used the same layout as in §7.1, and conducted the experiments as follows: Both the stranger and the user initially stood at the desk, and kept distances of 30cm, 75cm, and 100cm between them for different tests. The phone was vertically placed 20cm in front of the user on the desk. The user and the stranger were asked to alternatively depart from their positions.

For all three distance settings, Chaperone is able to detect all departure events with no false positives (precision and recall of 100%). iLock and iLock++, which are designed to defend against nearby attackers, also perform very well: among the 108 events, both algorithms had two false positives and one false negative for the 75cm user-stranger distance, and one false positive and one false negative for the 100cm user-stranger distance. The results show that interference from a nearby stranger has little impact on the detection in the lab environment. However, in real-world scenarios, there may be more than one person near the user. In addition, the activities from nearby people are unpredictable in terms of direction, intensity, timing, etc. Therefore, we further studied the potential of false positives/negatives resulting from nearby people in the real-world experiments; see §8.

7.3 Energy Consumption

Active acoustic sensing of Chaperone is triggered only when the Trigger module detects a potentially *vulnerable context* (e.g., at a bus stop). If the phone is in a private environment, e.g., home, Chaperone's processing needs will be negligent (i.e., no active acoustic sensing). However, Chaperone may still be occasionally triggered for a long period of time—e.g., the user is attending a conference, while leaving the phone on a table. Therefore, we use Android Battery Historian[2] to profile Chaperone's energy consumption on the Pixel with a 2770mAh battery. We fully charged the phone and kept it idle with no other applications running, except Chaperone and

[2] https://github.com/google/battery-historian

Location	Departure	Idle	iLock		iLock++		Chaperone	
			Precision (FP)	Recall (FN)	Precision (FP)	Recall (FN)	Precision (FP)	Recall (FN)
library	46	50	0.98 (1)	0.98 (1)	1.00 (0)	0.96 (2)	0.96 (2)	0.96 (2)
office	54	87	0.70 (20)	0.85 (8)	0.89 (6)	0.94 (3)	0.84 (10)	0.94 (3)
restaurant	71	93	0.89 (8)	0.90 (7)	0.95 (3)	0.89 (8)	1.00 (0)	0.99 (1)
coffee shop	36	42	0.79 (8)	0.86 (5)	0.94 (2)	0.86 (5)	0.92 (3)	0.94 (2)
lounge	50	59	0.78 (9)	0.64 (18)	0.81 (7)	0.60 (20)	0.87 (7)	0.96 (2)
bus stop	45	27	0.68 (15)	0.71 (13)	0.88 (5)	0.78 (10)	0.92 (3)	1.00 (0)
in-vehicle	58	33	0.72 (18)	0.79 (12)	0.88 (7)	0.91 (5)	1.00 (0)	0.91 (5)
acad. venue	6	0	-	0.8 (1)	-	0.8 (1)	-	1.0 (0)

Table 2: Precision and recall of three algorithms in eight locations (FP: # of False Positives, FN: # of False Negatives).

User #	Departure	Idle	Precision (FP)	Recall (FN)	Location (# of cases)
1	50	67	0.87 (7)	0.90 (5)	office (69), in-vehicle (39), bus stop (9)
2	31	21	1.00 (0)	0.97 (1)	in-vehicle (52)
3	39	44	0.93 (3)	0.95 (2)	coffee shop (50), lounge (33)
4	17	31	0.84 (3)	0.94 (1)	library (20), coffee shop (28)
5	50	52	0.98 (1)	0.98 (1)	library (20), restaurant (44), lounge(38)
6	108	110	0.93 (8)	0.99 (1)	library (39), restaurant (92), bus stop (49), lounge (38)
7	29	30	0.96 (1)	0.93 (2)	library (17), restaurant (28), bus stop (14)
8	42	36	0.93 (3)	0.95 (2)	office (72), acad. venue (6)

Table 3: Per-user results of Chaperone and case distribution in eight locations (FP: # of False Positives, FN: # of False Negatives).

system services. We placed the phone on a table with the maximum volume, while Chaperone was continuously conducting detection; the battery level dropped from 100% to 92.3% in an hour—see Figure 9. The peak discharging rate was about 0.2% per minute, with an average of 0.13% per minute. For comparison, one-hour music playing with the same volume consumed about 4% of the battery, while one-hour movie playing consumed about 9% (the idle phone took about 0.3%). Although Chaperone's battery consumption during active acoustic sensing is significant, it is still acceptable for daily use with help of the trigger module — low-frequency sensing with motion and location sensors can help avoid unnecessary acoustic sensing and save battery. Our survey (see § 9) also showed that the extra battery consumption was acceptable for most participants considering their smartphone usage habits and Chaperone's trigger mechanism.

8 Real-World Experiments

Summary. We evaluated Chaperone against a variety of real-world scenarios. We did not employ scenario-specific data for training the classifiers. Instead, we trained them using the data that we collected from one experimenter during the lab experiments (§7.1), following our "Robust" design goal (§4, require minimum tweaking for unseen scenarios). Figure 10 shows the overall detection performance of the three algorithms over 366 departing activities and 391 idle activities in real-world scenarios. The precision and recall scores of Chaperone are 93% and 96%, respectively, compared to iLock's 79% and 82%, respectively. With using Chaperone's candidate selection strategy, the precision of iLock++ increases up to 91% and the recall is slightly improved to 85%. Figure 11 shows the cumulative distribution function of the delay for the three algorithms; over 95% successful detection instances happen within 500 ms after the user crosses the 1m threshold. Although Chaperone has a longer delay than iLock, the delay gap is still acceptable. These results demonstrate Chaperone's efficacy in previously unseen real-world scenarios. Table 3 shows the precision and recall scores of eight experimenters in different locations. The three classifiers were trained with only one experimenter's (i.e., #3) data collected during the lab experiments. From the results, we can see that the pre-trained classifiers worked well for all eight experimenters, indicating that Chaperone is user-independent. We now discuss the results for the individual scenarios (summarized in Table 2).

8.1 Evaluation under Different Scenarios

Library. The experimenter shared a group study table with two or three students at our university library. Occasionally, strangers passed by near the table. The background noise came from people's chatting and the building's ventilation. The everyday activities involved reading and writing by the experimenter. In this environment, the detection rates of the three algorithms are mostly identical. As this scenario is close to the setting in the ideal experiments but with a few nearby strangers, iLock and iLock++ can also handle it well. The three algorithms shared a common false negative, caused by the simultaneous movements from both the user and a passer-by. The false positives were caused by interference from a nearby stranger's abrupt movements.

Office. The experimenter was alone in a narrow office cubicle and performing activities, such as using the keyboard and

monitor, and standing up to fetch documents from a shelf. There was background noise from computers, typing, and a regular office swivel chair. The cubicle has a semi-open structure, and we placed the phone at different positions on the table. When it was placed close to a cubicle divider, the acoustic signals were partially blocked. iLock has significantly lower precision and recall scores since it failed to handle partially blocked signals well, or was misled by changes in the magnitude of the echoes from the chair. With Chaperone's tracking strategy, iLock++ has the same recall score as Chaperone and a slightly higher precision score. For Chaperone, ten false positives were a result of the user's movements matching the preset departure pattern of Chaperone. For example, three false positives came from the eight document fetching cases—when the experimenter momentarily came close to the 1m line and then returned. All the three false negatives were related to the false positives in departure activities where Chaperone sent alerts before t_d (see "Metrics" in §6).

Restaurant and coffee shop. Since the layouts and results for the restaurant and coffee shop scenarios (one restaurant and two coffee shops) are similar, we present them together. For these scenarios, the experimenters were eating/drinking at different tables (e.g., round, corner, bar counter), and shared the tables with one or two nearby people (within 1m). Both types of places were noisy, crowded, and other customers were passing by. There was high-frequency noise from the entrance door, dragging of chairs, and dining carts in the restaurant; an espresso machine also sometimes produced high-pitched noise in the coffee shops. Chaperone performs very well in the restaurant: precision 100% and recall 99%. In coffee shops, three false positives from two experimenters have been observed when they temporally moved away from the counter seat, but the precision score is still 92%. iLock's precision is lower in the coffee shop than in the restaurant because of the interference from the occasional high-frequency noise from the espresso machine, while iLock++ is less affected. However, both iLock and iLock++ do not perform well in tracking the departure activities in some specific layouts where the experimenters passed by a near obstacle (e.g., a pillar) on their departure trace.

Lounge. We used a spacious, quiet lounge, where the experimenter was sitting on a couch, and the phone was placed either on a coffee table, or a couch (to simulate the phone being dropped from the pocket). The couch was shared with a stranger, and occasionally, there were people passing by. iLock and iLock++ do not perform well in the lounge with low recall scores of 64% and 60%. Due to the combination of the environmental factors (signal partially blocked by the furniture), and the user's departure trace (walking in a direction where the signal transmission is weak), iLock and iLock++ can hardly capture the user's movement as they highly rely on distance estimation. In contrast, Chaperone detects 96% of the departure activities. We record six false positives (including two in actual departure activities but where Chaperone sent alerts before t_d) for situations where the user reclined on the couch while the smartphone was on the coffee table. Similar to the document fetching cases in the office scenarios, the body reclining movement pattern is similar to the departure pattern, which misled Chaperone. One false positive is recorded when the smartphone was left on the couch and the user stood up from the couch, where a significant moving-away event was captured by Chaperone.

Bus stop. We experimented at two types of bus stops: an open-air bench and a small glass-enclosed waiting area. The experimenter left the phone either on the bench or a seat in the waiting area. There was high-frequency noise from passing vehicles and alert signals from trams. Several other people were also waiting for a bus or passing by. In this scenario, Chaperone significantly outperforms iLock and iLock++, detecting all the departure activities (recall: 100%). We note four false positives for Chaperone (the precision is still 92%) when the phone was placed between a stranger and the user, where the stranger-phone distance was very close to the user-phone distance. When the stranger moved away, Chaperone tracked their movement and resulted in a false positive. iLock and iLock++ were prone to be misled by the stranger's movement, especially when the user's movement range was intersected by the stranger's. In addition, the high-frequency noise sometimes interfered with the detection of iLock and iLock++ and produced false positives. Chaperone was unaffected by such high-frequency noise thanks to its noise handling strategy. Results from this scenario strongly suggest that Chaperone can operate reliably in such noisy environments.

In-vehicle. Since a significant number of smartphone losses happen during ride hailing [28], we specifically target this scenario, which includes several challenges: the car space is much smaller than other scenarios, and the leaving procedure is very short—the user opens the car door, steps out, and closes the door. Also, when exiting the car, friction noise is produced by clothes and the seat, as well as the clunking noise from the car door. We simulated the cases where the user leaves the phone on either the front or back seat in a sedan with different noise conditions for the state of the engine, radio, and air-conditioner. Chaperone has no false positives, and the recall reached 91%, outperforming iLock and iLock++. The false positives for iLock and iLock++ were the result of noise in the narrow car space when the user was stationary. However, the common noise in the car did not affect Chaperone's user tracking (due to the incorporation of noise detection, candidate selection algorithms and three user state classifiers). The false negatives for Chaperone primarily came from the short leaving procedure, and the movement of the car door when the user was closing it. To reduce false negatives, one possible solution is to shorten the decision window when the phone detects that it is in a vehicle. Nevertheless, Chaperone provides overall good performance for the car scenario.

Academic venue. We collected data at a workshop (a lecture room with over 50 people), and a conference keynote (a large

hall with over 900 people). We tested the keynote scenario at the end of the talk when the conference participants were leaving from the hall (crowded and very noisy). Due to the limited data collection time, we only collected three departing activities for each place. Chaperone worked well without any false negatives. One common false positive for iLock and iLock++ in the keynote hall is that they lost track of the user because the echo strength dropped quickly to the same level as the noise before the user was reaching the 1m line.

Summary. For real-world conditions iLock resulted in more false positives than in lab experiments. Using Chaperone's candidate selection strategy, iLock++ offers higher precision, especially in restaurant and coffee shop scenarios where environmental factors introduced more noise; iLock++ also improved in detecting more departure activities in office and in-vehicle scenarios. Chaperone outperforms both iLock and iLock++ in complicated scenarios like the lounge and bus stop. The decision making module determines the user's departure activities based on user's motion state and activity intensity, rather than estimating user-phone distance only. In general, Chaperone consistently performs well in terms of recall rate. However, among the eight locations, Chaperone has lower precision scores in the office and lounge scenarios, apparently, because users have a large movement range in these scenarios, and some specific activities (e.g., document fetching) are similar to the preset departure pattern of Chaperone. A possible solution is to enable different departure patterns for different types of locations.

8.2 Evaluation under Longer Idle Periods

The experiments in §8.1 evaluated false positives during everyday activities of short duration. In some scenarios, acoustic sensing may be triggered for a longer period of time while the phone is idle on a table with the user around, e.g., in a meeting room at an untrusted place. In this case, a false positive can be quite annoying. Therefore, we evaluated false positives with Chaperone running for 15–20 minutes in the following scenarios: office, library, restaurant, and meeting room. We configured Chaperone to continue to run after detecting a departure event. The experimenter performed everyday activities matching the scenarios. We observed no false positives in the office, library, and restaurant scenarios, while two false positives occurred in the meeting room scenario. Both false positives happened when the user was stationary, and the closest colleague, sitting around 30cm away, did some movements (e.g., adjusted their seat), misleading the detection. Overall, the false positive rate of Chaperone is acceptable for longer acoustic sensing sessions under different situations.

8.3 Effects of Other Interference Factors

As Chaperone relies on acoustic sensing, it may be affected by the following scenarios: 1) The sound transmission is partially blocked by an object very close to the speaker and microphone of a smartphone; 2) Multiple nearby Chaperone-enabled smartphones are conducting acoustic sensing concurrently. We evaluate these cases by running Chaperone in the standalone mode with the trained classifier models used in the real-world experiments. In each experiment, the smartphone(s) are placed in front of the experimenter with 0 degree orientation angle (i.e., vertical) and the experimenter moves at a normal speed. For each setting, we conduct 15 experiments.

8.3.1 Close-object Experiments

For the real-world experiments, we did not control the environment, including the presence/absence of nearby objects. We further perform controlled experiments to study the effect of nearby objects that partially block transmission. The Pixel phone that we use in these experiments, utilizes the bottom speaker and microphone for acoustic sensing. (We discuss smartphones with different hardware layouts in § 10.) Intuitively, if an object that is wider and thicker than the smartphone is placed very close to the bottom of the smartphone, the sound transmission will at least be partially blocked.

Although many factors, such as object numbers, surface materials, and placements, may affect the sound transmission, our main focus is to test Chaperone under different blocking effects. Therefore, we change the distance between the object and the phone to study the blocking effects. We conducted the close-object experiments in an office, and placed the Pixel (8.5mm thick) on a desk with a laptop on its left side and two books on its right (within 50cm to the phone). We investigated the effect of a single object in front of the bottom speaker and microphone. We used two objects—a 200-page notebook (landscape-oriented, height: 19mm, width: 266mm) and a 16-oz steel coffee mug (height: 198mm, width: 84mm), and phone-object distances of 5cm and 15cm. Besides, we tested the situation where the notebook was stacked on top of the Pixel (placed at the notebook's centre) with an 8mm gap between the desk and the notebook. When the notebook was placed 5cm away from the phone, the departure of the experimenter was detected in 13/15 cases; for the coffee mug at the same distance, 11/15 cases were detected. Since the coffee mug has a larger surface than the notebook to reflect the signal, it becomes more difficult to track the user's movement. However, when the mug was placed 15cm away, 14/15 cases were detected. When the phone was covered by the notebook, Chaperone detected 12/15 cases. Overall, Chaperone can still function when signal transmission is partially blocked.

8.3.2 Concurrent Sensing Experiments

Another situation of interest is when multiple Chaperone-enabled devices conduct acoustic sensing with the same acoustic signal at the same time. Intuitively, the interference caused by the direct transmission (from the speaker of a phone to

the microphone of the other) can be offset by the differential magnitude, since both acoustic signals have identical period, and the overlying signals are constant in each sensing period. Our pilot tests show that the acoustic signal generated by another phone, together with its echoes, could also be detected as additional noisy frames by Chaperone, which is not considered in our design. One solution is to adopt a higher noise threshold when Chaperone detects another identical acoustic signal close-by. In the experiments, we tuned the threshold and made no other changes in the noise detection sub-module to handle concurrent sensing.

We follow the basic setting used in the nearby stranger experiments: two Chaperone-enabled phones (the Pixel and Pixel 3, using the same classifiers) are placed in parallel on the desk and two experimenters stand in front of the two phones and leave alternatively. Each experimenter repeats the departure activity for 15 times. The shoulder-to-shoulder distance between the two users was 30cm, while the distance between the two phones was 75cm. No false positives were detected, while one false negative on the Pixel 3 and three false negatives on the Pixel were observed. To simulate the case where two Chaperone users are very close to each other, we reduced the distance between the phones and the distance between the users by 10cm: to 65cm and 20cm, respectively. Both phones detected 11/15 cases without any false positives. In comparison, when only the Pixel was conducting acoustic sensing, and the other conditions remained the same, we still observed five false negatives. Since the two users shared a largely overlapping activity range, it became more difficult to distinguish them. Nevertheless, there is no significant change in detection performance brought by concurrent sensing.

We added another Pixel (and an experimenter) to conduct concurrent sensing experiments with three devices by placing them in parallel, 75cm apart, with the Pixel 3 in the middle. For the 45 experiments, there was one false negative on each Pixel and three false negatives on the Pixel 3, with no false positives on any device. These results indicate that Chaperone can function concurrently on multiple devices with limited performance penalty. Similar to the close-object experiments, we cannot exhaust all possible settings and related factors, such as more smartphones (and users) and different placements. However, our experimental results have shown the feasibility of Chaperone in common concurrent sensing situations.

9 User Study

Our real-world device loss experiments show promising results for Chaperone. To validate the subjective nature of some of the results (e.g., the acceptability of the detection delay to the users) and to understand users' concerns for the adoption of Chaperone, we conducted an IRB approved user study.

9.1 Objectives and Methodology

We divide our objectives for the user study into three main themes: investigating device loss experiences and users' reactions, acceptability of Chaperone, and effective alert mechanisms for device loss. For device loss experiences, we collected data about the occurrence, location, reaction, and the final outcome of the event. For the acceptability of Chaperone, we collected data on detection ability, detection accuracy, power consumption, and overall effectiveness. We also collected data on what participants liked or disliked about Chaperone and whether they would use Chaperone on their devices. Finally, we asked participants regarding their preferred alert mechanisms for different environments based on perceived effectiveness and annoyance.

To achieve these objectives we conducted a three-part study: a semi-structured interview on smartphone loss experiences, a hands-on experience of Chaperone, and a semi-structured interview for their feedback on Chaperone. While a longer field study may have provided better insights, the nature of smartphone loss events cannot be controlled in a field study.

We recruited participants from the campus (excluding our research lab) and local community through word-of-mouth. We did not require participants to have experienced smartphone loss. For a realistic evaluation of Chaperone, the user study was held in a busy campus cafeteria during weekdays. At the cafeteria, participants responded to a brief demographic survey and the smartphone loss experiences interview.

For the hands-on experience, participants were asked to test Chaperone with real-time distance-tracking display on both the Pixel and the Pixel 3. They could test Chaperone freely and/or under the guidance of the investigator. At this stage, Chaperone alerted the user only through a pop-up message when it detected a potential smartphone loss. Then, we enabled a ringtone-and-vibration based alert without telling the participants about it, asked the participants to simulate a smartphone loss scenario, and observed their reaction to the alert. We chose the Pixel's "Nudge" as the alarm sound with alarm volume at 100%. We then demonstrated participants different alert methods including a strong ringtone (i.e., Pixel's "Classic Bell"), screen flashing, and notification sound to get their feedback on their preferences for each method for different locations. Finally we conducted the semi-structured interview to get their feedback on the acceptability of Chaperone and their preference for alert methods. We provide detailed interview questions in our project link.

9.2 Findings from the User Study

We have 17 participants (7 females, 10 males) in the study. 13 participants are 18–25 years old, and the rest are 26–30 years old; 15 are with Computer Science or IT background.

Smartphone loss and unattended experiences. In the first semi-structured interview, 11/17 participants reported having

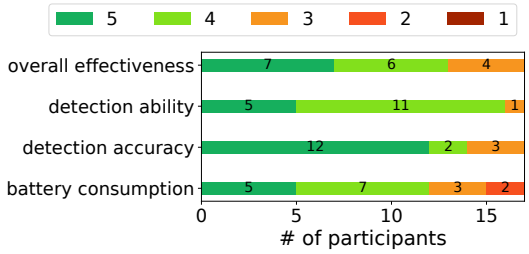


Figure 12: Participants' rating of Chaperone on a 5-point Likert scale (5: Very satisfied, 1: Not satisfied at all)

experienced smartphone loss. Four participants reported more than one loss. Among the 15 reported loss cases, two were due to pickpocketing (beyond Chaperone's threat model), ten were due to participants forgetting their phones, and three were due to phones slipping out of participants' pockets. Besides, 7/17 participants reported that they had unintentionally left their phones unattended to do something quick (e.g., go to a washroom) in public places. None of these unattended phone cases resulted in device theft or unauthorized access. Thus, we focus on the 15 smartphone loss cases.

The reported locations of the loss incidents include: library (four cases), street (three), washroom (two), in-vehicle (two) and one each for bus stop, meeting room, semi-open dormitory area, and gym. The participants realized the absence of the phone within 30 minutes for five cases (including two pickpockets), and more than one hour for eight cases. In two cases, the participants realized only when someone found the phone and returned it. Except two pickpockets and a forgotten phone in the semi-open dormitory area (later stolen by someone), the participants eventually recovered their phones.

In nine cases, participants went back to all possible places to look for their phone, which reportedly took them another one hour (four cases), or more than two hours (three cases) to recover their phone. Three participants used "Find My Device" services; one of them managed to recover the lost phone, while the other two failed.

This short survey indicates the importance of a preventive approach: finding lost/forgotten phones is time-consuming, and in some cases, such phones may never be found.

Feedback on Chaperone. For the device loss simulation, the researcher noted that 11/17 participants reacted (e.g., stopped leaving, turned/moved back) to the ringtone-and-vibration based alarm and another five participants mentioned that they had heard the alarm but they thought it was from somebody else's phone. (Recall that tests were done in a busy cafeteria.) In practice, Chaperone users would be aware of their alert sound so this confusion would unlikely happen; we did not inform participants about the type of alert to prevent them from explicitly waiting for it and biasing the results. 15 participants heard only the ringtone, but not the vibration; only one participant reported hearing the vibration. As for the inaudible acoustic sensing signal, all participants reported not noticing it during the whole demonstration.

During the interview, when asked what participants liked about Chaperone, all reported liking the idea of alerting a smartphone user when leaving the phone behind to prevent smartphone loss. In terms of dislikes, nine participants suggested the ringtone used in the hands-on experience should be more noticeable. Five thought the real-time distance tracking was not very accurate because they noticed small fluctuations in the real-time trace display although they did not lead to any false positive or false negative.

To measure Chaperone's acceptability, participants rated Chaperone on a 5-point Likert scale, as shown in Figure 12 (a higher score means a higher satisfaction), for its overall effectiveness (*Assuming that you want to use a device loss prevention solution, do you think Chaperone is an effective system?*), detection ability (*How do you rate the Chaperone's ability to capture a smartphone loss?*), and detection accuracy (*How do you rate the Chaperone's detection accuracy? (a counterexample is that Chaperone sends an unwanted alert when the owner is not actually leaving*). The average effectiveness score is 4.2, the average detection ability score is 4.2, and the average detection accuracy score is 4.5. The results show that the participants were satisfied with the performance of Chaperone.

For the power consumption, we first shared with participants the battery consumption rate of Chaperone conducting detection reported in §7.3, and then explained that it is only triggered when all conditions (see §5.1) in the trigger module are satisfied; i.e., the real power consumption will depend on smartphone usage habits. Therefore, we asked the participants to rate the impact of Chaperone's power consumption based on their habits from 1 (i.e., significant) to 5 (i.e., negligible). The average score is 3.88, implying that the power consumption is acceptable for most participants. Participants mentioned that they usually do not spend a long time in untrusted or public places, and therefore, the extra power consumption by Chaperone is still acceptable considering the potential benefits. Two participants rated the power consumption impact as 2. Their reported reason was that they are heavy smartphone users and their smartphones can hardly accommodate any additional battery consumption.

Alert. We also asked participants to comment on the alert they received during the hands-on experience. Among 16 participants who perceived it, twelve thought the timing of the alert was good to attract their attention, three thought the alert was a little late and they might miss it if the alarm sound was not loud enough in a noisy environment, and one participant thought Chaperone sent the alarm a little early and suggested to allow adjustable sensitiveness for the alert.

For participants' rating of different alerts, 13 participants rated the effectiveness of a strong ringtone as "Very effective" or "Effective," while eleven participants thought vibration was "Not effective at all" since the vibration was too weak to alert the user in a noisy environment. As for screen flashing, ten

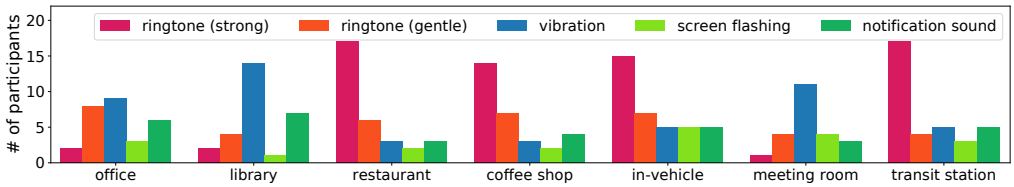


Figure 13: Participants' preferences of alert methods for different locations.

participants rated it as "Not effective at all" since the phone is usually behind a leaving user. Participants were also asked to choose their preferred alert methods for seven location types based on their perceived effectiveness and annoyance. They were allowed to choose none or multiple alert methods for each location. Most participants chose noticeable alert methods like strong ringtones for noisy places, while for the quiet places gentle ringtones and vibrations were preferred; see Figure 13. Five participants chose screen flashing for the in-vehicle scenario as a complementary alert method since it can make the phone noticeable in a dark environment. The results suggest that the trigger module can help Chaperone to determine an appropriate alert method based on the current context. Ten participants mentioned they needed a customized ringtone for device loss and nine participants expected the volume of the ringtone to be automatically adjusted based on the ambient noise level. Three participants requested further actions like e-mail notifications if the user failed to respond to the alert within a pre-specified time. These comments and suggestions are useful in designing a context-aware alert mechanism for Chaperone.

Adoption. We asked the participants: *Would you like to install Chaperone as a device loss prevention app on your phone? (Yes/No/Maybe)*. 8/17 participants answered "yes" because they thought Chaperone helped reduce the risk of smartphone loss. Eight participants answered "maybe"; four of them believed they had a good habit of always keeping their phones with them but they still wanted to try it to record how often they leave their phones behind, two had privacy concerns due to Chaperone requesting the microphone permission, one was worried about the effectiveness of the alarm in very noisy environments, and the other one expected that Chaperone could learn from a user's habits to trigger the sensing smartly and save battery. Only one participant answered "no" as they did not need a device loss prevention application due to the perceived low probability of losing the device.

Threats to validity. Our user study has some reasonable limitations similar to other studies involving human subjects including the limitation of scope to people willing to participate, self reported and subjective views, and participants might be inclined to provide favorable responses to the researchers. More specific limitations follow. Most of our participants are current undergraduate or graduate students in Computer Science, lacking diversity in participants' background. Although we did not require participants to have smartphone loss experiences, the advertised content for the user study mentions the study is to "test the context-aware techniques on smartphones to prevent smartphone loss", which may have attracted users with such experiences. Another threat to validity is that the first interview about smartphone loss experiences may have primed participants for adoption of Chaperone. During the user study, we used a Pixel and a Pixel 3 as the demonstration phones. The participants reported their perceptions of different alert methods based on their experience of using these two devices. These results may not be fully applicable to other devices due to hardware differences (e.g., max volume difference, vibrator difference). In addition, since our user study focuses on collecting smartphone users' perception about Chaperone and its alert mechanism based on one demo session, it may not cover potential issues regarding long-term usage of a product-ready Chaperone.

10 Discussion

We discuss a few issues relevant to the deployment and usage of Chaperone, including limitations of our current prototype.

Very close attackers. In §3, we assume that the attacker is initially farther away from the phone than the owner. Li et al. [13] consider an attacker who is initially closer to the device than the owner. The potential consequence is that the system may track the wrong person since it assumes that the initially closest person is the owner. To defend against such an attacker, Li et al. adopt a dedicated approach that requires two microphones and inertial measurement sensors to distinguish the attacker from the owner. However, it provides acceptable accuracy only when the owner and the attacker are facing each other, i.e., not side-by-side. Their approach also requires the owner following a straight path away from the phone with a consistent relative user-phone orientation (unlike Chaperone). Finally, both microphones may not always be available at the same time, since the top or rear microphone could be covered when the phone is lying on a surface. For Chaperone, a potential defense against such an attacker is to trigger sensing right after the user puts down the phone on a surface to track the user's hand movement immediately, assuming the user's hand is the closest moving object at that moment.

Active attackers. As mentioned in the threat model (§3), Chaperone targets nearby opportunistic attackers, not Chaperone-aware active attackers. An active attacker may attempt to disarm Chaperone so that the auto-lock and alert mechanisms are not triggered. We briefly discuss two types

of active attacks here. 1) Jamming attack: If an attacker continuously generates loud noise over the inaudible high-frequency band used by Chaperone, the echo of Chaperone's own acoustic signal will be lost. However, it is possible to measure the ambient noise to detect such an attack and alert the user before conducting acoustic sensing. 2) Misleading attack: A nearby attacker makes significant movements to produce strong reflected signals so that Chaperone tracks the attacker instead of the user. When the owner is leaving, the signal reflected by the owner becomes weaker and the nearby attacker's movements overlap the owner's departure trace. Note that, the attacker must be very close to the target smartphone (i.e., within one meter) and make significant movements before the owner moves too far away. For a better defence against the misleading attack, a potential avenue is to improve the motion tracking algorithm (e.g., include motion history), or use additional detection methods (e.g., RF sensing [29]) to distinguish different people.

False positives and negatives. We noticed some common false positives caused by (relatively) longer range user movements (e.g., reclining on the couch) in the lounge scenarios, and false negatives caused by moving objects (e.g., the car door) in the in-vehicle cases. Since our models were trained with data from the lab-based experiments, it was challenging to handle these special cases. An apparent solution is to train the models with more real-world data covering these situations. We use only lab data for our models to measure Chaperone's robustness in newly encountered situations—which we assume Chaperone to face frequently in practice.

Smartphone hardware differences. We focus on the environmental factors that affect acoustic sensing. Most experiments were conducted on a Pixel, while a few concurrent sensing experiments were done with a Pixel 3. To support different devices, we explain how to transfer the classification model in § 5.5. A systematic study on how hardware differences affect Chaperone's sensing ability and the necessary parameter adjustments due to these differences is future work. Two possible areas are the following: 1) Sensing ability: given differences in microphones and speakers, the recording quality above 19kHz may vary on different smartphones, directly affecting the magnitude of the received signal. 2) Hardware layout: the positions of speakers and microphones may differ for different smartphones. Even for the bottom microphones, some may be placed on the bottom edge (e.g., Pixel) while some may be on the bottom front (e.g., Pixel 3). Such differences may result in different sensing ranges because of the directionality of microphones.

Measurement inaccuracies. During our data collection, a human observer marked the reference points for the moment t_a when the user passes the 1m line. A standalone distance sensor may have provided a more accurate labelling. However, setting up such a sensor in public places, like restaurants and coffee shops, is inconvenient, and therefore, we settled for labelling by a human observer.

11 Conclusion

We present Chaperone as a standalone, opensource Android app that uses acoustic sensing to detect smartphone loss and lock the phone in real-time. Our real-world experiments show that it can operate reliably in diverse real-world scenarios characterized by high ambient noise, crowded locations, and diverse physical layouts, *without* retraining our classifiers for specific scenarios. Our user study provides positive evidence that Chaperone can indeed be made into a practical tool to help prevent device loss, and thereby reduce serious privacy and security threats caused by lost smartphones. Beyond device loss, Chaperone's design and our extensive real-world datasets will help advance acoustic sensing research.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We are grateful to Dr. Aanjhan Ranganathan for spending time and effort in coordinating the revision process. We thank Dr. Tao Li, the author of iLock, for providing details about iLock, and thank Dr. Ju Wang, Dr. Wen Cui, and Dr. Lin Cai for their great help on acoustic sensing. We acknowledge the support of NSERC for grant RGPIN-2014-0549.

References

[1] F. Adib, Z. Kabelac, D. Katabi, and R. C Miller. 3D tracking via body radio reflections. In *NSDI '14*, 2014.

[2] Business-news.eu. 4% or €20 million: A stolen mobile may cost you hundreds, but could cost your employer millions. https://www.business-news.eu/2019-4-or-20-million-a-stolen-mobile-may-cost-you-hundreds-but-could-cost-your-employer-millions, 2019.

[3] C. Cai, R. Zheng, and M. Hu. A survey on acoustic sensing. *arXiv preprint arXiv:1901.03450*, 2019.

[4] DigitalTrends. Study reveals americans lost $30 billion worth of mobile phones last year. https://www.digitaltrends.com/mobile/study-reveals-americans-lost-30-billion-of-mobile-phones-last-year/, 2012.

[5] G. Elert. Frequency range of human hearing. *The Physics Factbook*, 2003.

[6] Google. Jacquard Jacket. https://atap.google.com/jacquard/, 2018.

[7] M. J. Hussain, L. Lu, and S. Gao. An RFID based smartphone proximity absence alert system. *IEEE TMC*, 16(5), May 2017.

[8] Kaspersky Lab. Around 23,000 devices go missing every month, finds Kaspersky Lab. Press release (Aug. 9, 2018). https://www.kaspersky.com/about/press-releases/2018_missing-devices, 2018.

[9] H. Khan, A. Atwater, and U. Hengartner. Itus: An implicit authentication framework for Android. In *MobiCom '14*. ACM, 2014.

[10] H. Khan, U. Hengartner, and D. Vogel. Targeted mimicry attacks on touch input based implicit authentication schemes. In *MobiSys '16*. ACM, 2016.

[11] L.A. Times. 68% of smartphone theft victims never recover device, report says. https://www.latimes.com/business/technology/la-fi-tn-70-smartphone-theft-victims-never-recover-device-20140507-story.html, 2014.

[12] P. Lazik, N. Rajagopal, O. Shih, B. Sinopoli, and A. Rowe. ALPS: A Bluetooth and ultrasound platform for mapping and localization. In *SenSys '15*, 2015.

[13] T. Li, Y. Chen, J. Sun, X. Jin, and Y. Zhang. iLock: Immediate and automatic locking of mobile devices against data theft. In *CCS '16*. ACM, 2016.

[14] X. Liu, D. Wagner, and S. Egelman. Detecting phone theft using machine learning. In *ICISS '18*. ACM, 2018.

[15] W. Mao, J. He, and L. Qiu. CAT: High-precision acoustic motion tracking. In *MobiCom '16*. ACM, 2016.

[16] W. Mao, M. Wang, and L. Qiu. AIM: Acoustic imaging on a mobile. In *MobiSys '18*. ACM, 2018.

[17] Y. Mirsky, A. Shabtai, L. Rokach, B. Shapira, and Y. Elovici. SherLock vs Moriarty: A smartphone dataset for cybersecurity research. In *AISec '16*. ACM, 2016.

[18] NetworkWorld.com. More data breaches caused by lost devices than malware or hacking, Trend Micro says. https://www.networkworld.com/article/2988643/device-loss-data-breach-malware-hacking-trend-micro-report.html, 2015.

[19] Patec. Mini Bluetooth anti-lost anti-theft alarm kid pet object finder. https://www.amazon.ca/Patec%C2%AE-Bluetooth-Anti-lost-Anti-theft-Object/dp/B00MPGKL1U, 2014.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[21] Protect Your Bubble. Nearly half a million Brits had phones stolen last year. https://uk.protectyourbubble.com/our-blog/blog/2017/03/23/nearly-half-million-brits-mobile-phone-stolen-last-year, 2017.

[22] Y. Ren, C. Wang, J. Yang, and Y. Chen. Fine-grained sleep monitoring: Hearing your breathing with smartphones. In *INFOCOM '15*. IEEE, 2015.

[23] K. Sun, T. Zhao, W. Wang, and L. Xie. VSkin: Sensing touch gestures on surfaces of mobile devices using acoustic signals. In *MobiCom '18*. ACM, 2018.

[24] Symantec. The Symantec smartphone honey stick project. https://www.symantec.com/content/en/us/about/presskits/b-symantec-smartphone-honey-stick-project.en-us.pdf, 2012.

[25] TheRegister.co.uk. A quarter of banks' data breaches are down to lost phones and laptops. https://www.theregister.co.uk/2016/08/25/us_bank_breaches_survey/, 2016.

[26] Y. Tung, D. Bui, and K. G. Shin. Cross-platform support for rapid development of mobile acoustic sensing applications. In *MobiSys '18*. ACM, 2018.

[27] Y. Tung and K. G Shin. Use of phone sensors to enhance distracted pedestrians' safety. *IEEE TMC*, 2018.

[28] Uber. The 2019 Uber lost & found index. https://www.uber.com/newsroom/2019-uber-lost-found-index, 2019.

[29] W. Wang, A. X Liu, M. Shahzad, K. Ling, and S. Lu. Understanding and modeling of WiFi signal based human activity recognition. In *MobiCom '15*. ACM, 2015.

[30] W. Wang, A. X. Liu, and K. Sun. Device-free gesture tracking using acoustic signals. In *MobiCom '16*. ACM, 2016.

[31] Y. Wang, K. Wu, and L. M Ni. WiFall: Device-free fall detection by wireless networks. *IEEE TMC*, 2017.

[32] J. Wiese, T. S. Saponas, and A.J. B. Brush. Phoneprioception: Enabling mobile phones to infer where they are kept. In *CHI '13*. ACM, 2013.

[33] Xiaomi. Mi Band. https://www.mi.com/global/miband/, 2014.

[34] X. Yang, K. Hasan, N. Bruce, and P. Irani. Surround-See: Enabling peripheral vision on smartphones during active use. In *UIST '13*. ACM, 2013.

[35] X. Yu, Z. Wang, K. Sun, W. T. Zhu, N. Gao, and J. Jing. Remotely wiping sensitive data on stolen smartphones. In *ASIACCS '14*. ACM, 2014.

[36] S. Yun, Y. Chen, H. Zheng, L. Qiu, and W. Mao. Strata: Fine-grained acoustic-based device-free tracking. In *MobiSys '17*. ACM, 2017.

[37] H. Zhang, W. Du, P. Zhou, M. Li, and P. Mohapatra. DopEnc: Acoustic-based encounter profiling using smartphones. In *MobiCom '16*. ACM, 2016.

[38] B. Zhou, M. Elbadry, R. Gao, and F. Ye. BatMapper: Acoustic sensing based indoor floor plan construction using smartphones. In *MobiSys '17*. ACM, 2017.

[39] B. Zhou, J. Lohokare, R. Gao, and F. Ye. EchoPrint: Two-factor authentication using acoustics and vision on smartphones. In *MobiCom' 18*. ACM, 2018.

[40] M. Zhou, Q. Wang, J. Yang, Q. Li, F. Xiao, Z. Wang, and X. Chen. PatternListener: Cracking Android pattern lock using acoustic signals. In *CCS '18*. ACM, 2018.

[41] Y. Zhu, Z. Xiao, Y. Chen, Z. Li, M. Liu, B. Y Zhao, and H. Zheng. Et tu Alexa? When commodity WiFi devices turn into adversarial motion sensors. In *NDSS '20*, 2020.