No Seeing is Also Believing: Electromagnetic-emission-based Application Guessing Attacks via Smartphones

Xiaoyu Ji, *Member, IEEE*, Yushi Cheng, Wenyuan Xu, *Member, IEEE*, Yuehan Chi, Hao Pan, Zhuangdi Zhu, Chuang-Wen You, Yi-Chao Chen, and Lili Qiu, *Fellow, IEEE*

Abstract—Mobile devices have emerged as the most popular platforms to access information. However, they have also become a major concern of privacy violation and previous researches have demonstrated various approaches to infer user privacy based on mobile devices. In this paper, we study the electromagnetic (EM) emission of a laptop that could be harvested by a commercial-off-the-shelf (COTS) mobile device, *e.g.*, a smartphone. We propose `MagAttack`, which exploits the electromagnetic side channel of a laptop to guess user activities, *i.e.*, application launching and application operation. The key insight of `MagAttack` is that applications are discrepant in essence due to the different compositions of instructions, which can be reflected on the CPU power consumption, and thus the corresponding EM emissions. `MagAttack` is challenging since that EM signals are noisy due to the dynamics of applications and the limited sampling rate of the built-in magnetometers in COTS mobile devices. We overcome these challenges and convert noisy coarse-grained EM signals to robust fine-grained features. We implement `MagAttack` on both an iOS and an Android smartphone without any hardware modification, and evaluate its performance with 30 popular applications, 30 YouTube videos, and 50 top websites in China. The results demonstrate that `MagAttack` can recognize aforementioned 30 applications with an average accuracy of 98.6%, and identify which video out of the 30 candidates being played with an average accuracy of 97.5% and visiting which website among the 50 candidates with an average accuracy of 90.4%.

Index Terms—Electromagnetic emission, mobile devices, user activity inference

1 Introduction

Mobile devices have emerged as the most popular platforms to assist daily activities and exchange information over the Internet. According to Gartner [1], there will be more than 11 billion phones, tablets and laptops by the end of 2018. Along with the rapid growth are the privacy concerns. The proliferation of mobile devices has been a major concern in the security and privacy communities. Various side channels have been utilized for electrical-appliance usage analysis [2], [3], [4], decryption of cryptographic computation [5], [6], [7], and human-device activity recognition [8], [9], [10], [11]. These side-channel attacks, especially the last ones, draw increasing attention due to the widespread use of mobile devices and the increasingly intensive interaction between human and smart devices.

Prior researches [9], [10], [11], [8], [12] have shown several side-channel attacks that can sense human-device activities. Zhuang *et al.* and Zhu *et al.* utilize acoustic emanations to infer user keystrokes [9], [11]. Cai *et al.* use motion sensors to infer user tapping and gesture inputs on smartphones [10]. Clark *et al.* use the AC power consumption to recognize the web page that a user browses on a laptop [8]. However, Clark's scheme requires modification of the power outlet to measure the AC power consumptions and can only work when the laptop is being charged. Jana *et al.* sense user applications based on the footprints of applications on memory usage [12], which is intrusive since they need to log into the system and run a background process in parallel with the target application.

In this paper, we investigate a new EM-based side-channel attack for user activity inference. We propose `MagAttack`, which detects and recognizes user activities by tracking the EM emissions from the laptop's CPU. Compared with existing side-channel attacks, `MagAttack` is non-intrusive and can be implemented on the COTS mobile devices without any hardware modification. The underlying principle of `MagAttack` is that, for an application, each time when being launched, a fairly unique and consistent sequence of CPU instructions are executed, as shown in Fig. 1. When a CPU executes different instructions, it emits various EM signals accordingly, which can be further captured by the built-in magnetometer in a mobile device for privacy inference and hereafter we name this kind of attack *application guessing attack*. `MagAttack` recognizes two levels of application guessing attacks: (1) which application is being launched, *i.e.*, application recognition, and (2) what a user is doing with the application, *i.e.*, operation recognition.

- *Corresponding author: Yushi Cheng*
- *X. Ji, Y. Cheng, W. Xu and Y. Chi are with the College of Electrical Engineering, Zhejiang University, Hangzhou, CN. E-mail: {xji, yushicheng, wyxu, johannc@zju.edu.cn}*
- *H. Pan and Y. Chen are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, CN. E-mail: {panh09@sjtu.edu.cn, yichao0319@gmail.com}*
- *Z. Zhu is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 488240, USA. E-mail: {zhuzhuan@msu.edu}*
- *C. You is with the NTU IoX Center, National Taiwan University, Taipei, TW. E-mail: {cwyou@ntu.edu.tw.}*
- *L. Qiu are with the Department of Computer Science, University of Texas at Austin, Austin, TX 78712-1757, USA. E-mail: {lili@cs.utexas.edu}*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

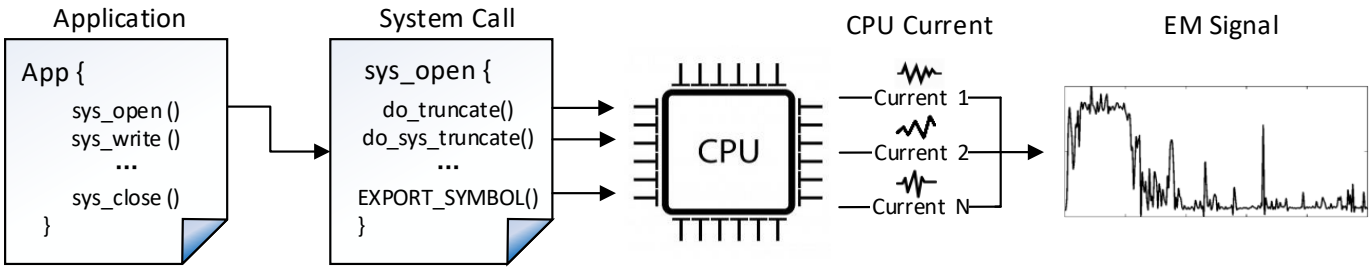


Fig. 1: The cause of EM emission from the laptop's CPU. An application consists of various system calls, which are composed of different instruction sequences and those instructions generate corresponding currents, and finally EM signals.

For example, we can figure out that a user is launching a web browser and recognize which web page the user is visiting. The significance of `MagAttack` lies in that it can be the prerequisite of other user privacy violation attacks. For example, when inferring user passwords through keystrokes [9], [11], `MagAttack` can be employed first to detect the launching of finical applications such as PayPal. Besides, with the help of `MagAttack`, an adversary can learn a user's interests and habits by continuously tracking the application usage of the user.

Inferring laptop user activity via the EM side channel is promising yet challenging. First, both the location and orientation of a smartphone affect the captured EM signals. The former decides the initial EM amplitude as a result of the earth's magnetic field, and the latter determines the changing trend of the EM amplitude. Second, the magnetic sensors in COTS mobile devices such as smartphones usually have low sampling rates. Different from the sensors used in [6], [7], [13], [4] with working frequency ranging from MHz to GHz, a magnetometer in mobile devices can only measure frequency up to 100 Hz. In other words, information from high frequency signals is lost. Third, EM signals might be inconsistent when the application changes, *e.g.*, a website is updated. Actually, many websites change their contents and components such as pop-out advertisements on a daily basis, and the captured EM signals are different as a result.

To address the aforementioned challenges, we first propose a reduction scheme to eliminate the influence from the earth's magnetic field and the ambient noise. Then, for application launching detection, we design a sliding-window based pre-screening algorithm for preliminary detection, and a fine-grained Support Vector Machine (SVM) classifier for refinement. For application recognition, we employ Short Time Fourier Transform (STFT) and Principal Component Analysis (PCA) for feature extraction, and use a Random Forest (RF) classifier to achieve accurate recognition. For operation recognition, we use Wavelet Transform and Multi-Resolution Analysis algorithm (MRA) to deal with the more dynamic EM signals compared with that of application recognition.

Application Scenario: we envision that `MagAttack` can be used in public areas where an adversary sits near a victim and is able to put his/her smartphone in the vicinity of the victim's laptop, e.g., attached on the backside of the table where the laptop is put on, and draw no attention. The adversary's goal is to infer the victim's laptop activities for habit tracking or further privacy violation attacks such as password inference. To the best of our knowledge, this is the first side-channel attack to infer laptop user activities by the means of CPU EM emissions. In summary, our contribution includes:

- We analyze the underlying correlation between the applications and the corresponding CPU EM emissions. We propose to use a mobile device to infer user activities on a laptop by tracking the EM emissions from the laptop's CPU.
- We investigate the distinctiveness of EM emissions caused by various user activities, and elaborately design `MagAttack` to differentiate them reliably.
- We implement `MagAttack` on commercial smartphones without any hardware modification, and evaluate it with 30 popular applications, 30 YouTube videos, and 50 top websites in China. The results demonstrate that `MagAttack` can detect application launching with a precision of 96.5% and a recall of 91.8%, recognize 30 applications with an average classification accuracy of 98.6%, and classify 30 videos with an average accuracy of 97.5% and 50 websites with an average accuracy of 90.4%.
- Compared with our prior work [14], we enhance the method for application/operation classification and improve the recognition accuracy for both applications and operations. In addition, we demonstrate the feasibility of our methods with a larger number of applications, shorter EM sample lengths (i.e., shorter collection time), and lower sampling rates, and extending our methods to new scenarios such as identifying which video the user is watching.

2 Measuring EM Emission to Infer Application

In this section, we first introduce the built-in magnetometer on mobile devices, and then show the feasibility of `MagAttack`.

2.1 Magnetometer on Mobile Device

A magnetometer is an instrument that can measure the direction, strength, and relative change of a magnetic field at a particular location. The built-in magnetometer on mobile devices is usually a Hall Effect sensor, which is small, cheap and low in sensitivity ($<= 5mV/mT$). The sampling rate of the built-in magnetometer is configurable, which usually varies from 4 Hz to 100 Hz for smartphones. Due to its low cost and extensive functions, *e.g.*, employed with gyroscopes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

TABLE 1: Top 10 system calls invoked when different applications being launched.

Safari	Chrome	iTunes
workq_kernreturn (17.6%)	kevent (24.0%)	workq_kernreturn (13.9%)
bsdthread_ctl (13.0%)	write (12.4%)	geteuid (9.5%)
stat64 (8.4%)	workq_kernreturn (6.6%)	stat64 (8.2%)
pread (8.2%)	read (6.1%)	bsdthread_ctl (7.9%)
madvise (6.5%)	recvmsg (5.8%)	getdirentires64 (4.9%)
openat (3.5%)	stat64 (5.0%)	getattrlist (4.7%)
getdirentires64 (3.0%)	bsdthread_ctl (4.3%)	madvise (4.6%)
kevent_qos (2.9%)	psynch_mutexdrop (2.7%)	read (3.5%)
mmap (2.6%)	psynch_mutexwait (2.4%)	open_nocancel (3.2%)
geteuid (2.5%)	mmap (2.0%)	kevent_qos (2.9%)

and accelerometers for motion tracking, the magnetometer is widely equipped on COTS mobile devices, and thus can be a good alternative for EM measuring.

2.2 Application Launching

During application launching, the Launch Services framework provides primary support. It sends a message to `WindowServer`, which in turn calls `fork()` and `execve()` (both are system calls) to run the requested application [15]. The requested application then, runs user functions in the user space and invokes system calls to interact with the kernel and access the hardware. Thus, an application executes both user functions and system calls when being launched.

Upon user activity inference, we focus on the system calls invoked during application launching. As shown in Fig. 1, an application consists of a series of system calls. Our hypothesis is that different system calls are invoked at different frequencies for various applications. To validate it, we use a system trouble-shooting tool *dtrace*[16] (available on Linux, Mac OS, and Windows [17]) to capture the name and time of the executed system calls when various applications (Safari, Chrome, and iTunes) are being launched on the same laptop (a MacBook Air). The results in Tab. 1 demonstrate that the most intensive system call for Safari is `workq_kernreturn()`, which accounts for 17.6%. While for Chrome, it is `kevent()` that holds the largest portion of 24.0%. For iTunes, `workq_kernreturn()` appears as the most intensive system call as well but with a different proportion of 13.9%. It confirms our hypothesis that the system calls invoked when launching different applications, are discrepant in both type and frequency, even for applications of the same type.

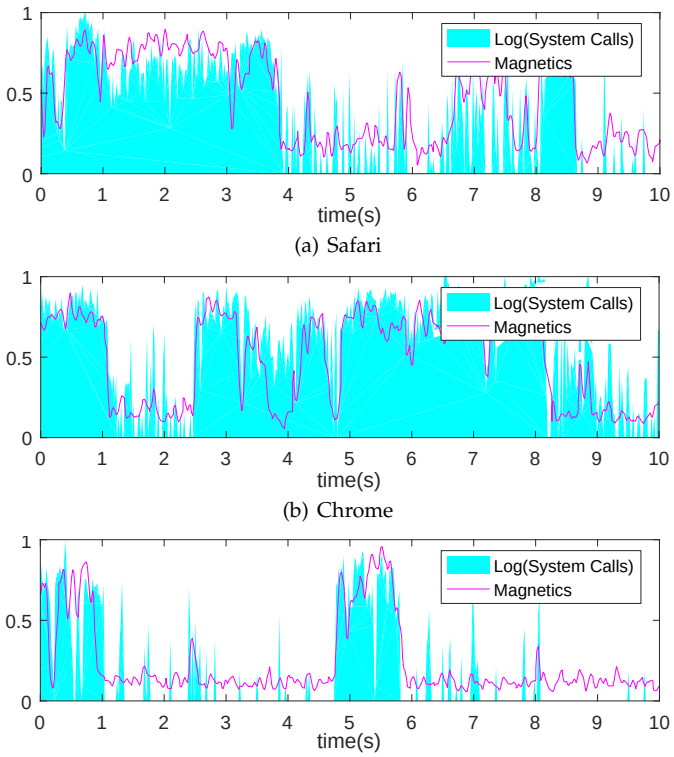
2.3 From System Call to EM Emission

A system call is a wrapper function that consists of a sequence of instructions [18], as revealed in Fig. 1. As a result, various system calls are composed of different instruction sets and thus generate distinct CPU power consumptions.

A CPU chip consists of a large number of CMOS (Complementary Metal Oxide Semiconductor) [19] transistors arranged in a lattice form, which perform basic arithmetic, logical, control and input/output operations specified by the instructions. Energy consumption of the CPU heavily depends on the power dissipation of the CMOS lattice. Average CPU power consumption [20] P_{avg} can be calculated as follows:

$$P_{avg} = \frac{CV(\alpha)^2 AF(\alpha)}{2} \quad (1)$$

where C represents the CMOS capacitance and is a function of the transistor size and the wire length. V is the supply voltage to CPU. A is the average switching frequency of the CMOS transistors and F is the clock frequency. V and F are further related to the CPU load α. When executing various instructions, the CPU involves different numbers of CMOS transistors and generate different loads, resulting in distinct power consumptions as well as CPU currents. Since various system calls are composed of different instruction sets, the CPU currents for those system calls are likely to be diverse, which contribute to distinct EM signals [21].



(a) Safari

(b) Chrome

(c) iTunes

Fig. 2: EM signals are correlated with the system call traces but distinct among applications. Note that the two traces are normalized for illustration and the Y-axis represents the frequency of system calls or the magnitude of EM signals.

2.4 Feasibility of `MagAttack`

As various applications invoke different system calls when being launched, the emitted EM signals shall correlate with the system calls executed by the CPU but remain distinct among applications. To validate our hypothesis, we capture the EM signals with an iPhone SE smartphone when different user applications (Safari, Chrome and iTunes) are launched on a MacBook Air laptop. During experiments, the target laptop is placed on a round table with the attack smartphone attached on the backside to draw no attention,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

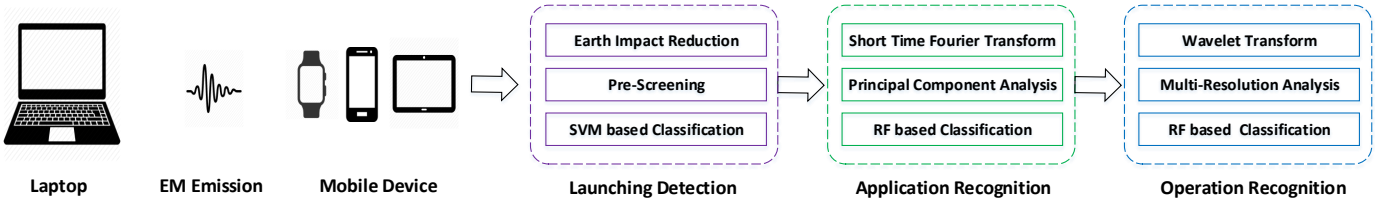


Fig. 3: Workflow of MagAttack, which detects application launching, recognizes user activities based on EM signals collected by mobile devices in the target laptop's vicinity.

as shown in Fig. 7. The detailed information of the smartphone and the laptop is depicted in Tab. 2 and Tab. 3 in Sec. 5. Meanwhile, we use *dtrace* to capture the name of the executed system calls and the time they are being called in microsecond granularity.

Since the system call is recorded in microseconds while the EM signal is recorded in 10-millisecond granularity (the sampling rate of the iPhone SE magnetometer is 100 Hz), we transform the system call trace into a time-versus-number histogram. The transformed trace is a two-column matrix $E = [\vec{t}; \vec{n}]$, where vector $\vec{t}$ records time units in 10-millisecond granularity, and vector $\vec{n}$ records the number of system calls during that time unit. Then, we align the magnetic trace with the system call trace by shifting the former one so that the two traces have maximal correlation coefficient. We plot the logarithmic system call traces and the EM signals in Fig. 2 (both are normalized for illustration), from which we can observe that:

- EM signals show strong resemblance to the system call traces captured at the same time, *e.g.*, the system-call-intensive moment also has a high EM magnitude.
- EM signals demonstrate distinct patterns among applications, even for those of the same type: Safari and Chrome, *e.g.*, the EM signal of Chrome is more dynamic and has more peaks compared with that of Safari.

These findings shed light upon inferring user activities on laptops via EM signals captured by nearby mobile devices. Since various user activities invoke different system calls, the resulting CPU power consumptions cause varying EM signals, which are distinct and associated with the activities and thus in turn can be utilized to conduct user activity inference.

3 Threat Model

In this section, we present the threat model of MagAttack. Since the adversary's goal is to infer user activities on user's laptop without his awareness, we consider the following attack scenario: *in a public area such as a library, a target is using his laptop. The adversary sits near to him, and tries to figure out what the target is doing on the laptop (e.g., what applications the target launches and what operations the target performs)*. In such a scenario, we assume that the adversary has following abilities.

Vicinity to Target Laptop. We assume the adversary's mobile devices can be in the target laptop's vicinity, e.g., attached on the backside of the table where the laptop is put on, and draw no attention.

Algorithm 1: Earth Impact Reduction

Input: $mag = \{mag_x(t), mag_y(t), mag_z(t)\}, t = 1 \dots n$: three-dimensional signals

Output:
- $M = M(t), t = 1 \dots n$: aggregated signals.
- $M_{norm} = M_{norm}(t), t = 1 \dots n$: aggregated and normalized signals.

$M = mag$
for $i \in \{x, y, z\}$ **do**
  $M_i = M_i - avg(M_i)$ // centralization
for $t \in [1, 2, \dots, n]$ **do**
  $M(t) = \sqrt{M_x(t)^2 + M_y(t)^2 + M_z(t)^2}$ // aggregation
$M_{norm} = \frac{M - min(M)}{max(M) - min(M)}$ // normalization

No Target Laptop Access. We assume that an adversary may target at any users of her choices, but she has no direct access to the target laptop. She cannot physically touch/see the screen, or install a malware.

No User Interaction. The adversary cannot ask users to perform any operations, such as pressing a button or running a specific application.

With the above assumptions, the adversary can launch application guessing attacks, i.e., guessing which application is being launched and what is the very operation of a user when using the application. Both attacks can violate the privacy of users and can be the first step of other severe attacks such as keystroke inference [9], [11].

4 MagAttack Design

4.1 Overview

To infer user activities, the adversary first puts her mobile device in the target laptop's vicinity and draws no attention. Then, the attack device collects the electromagnetic emissions from the laptop's CPU, based on which MagAttack detects application launching, recognizes running applications, and figures out user operations, as shown in Fig. 3.

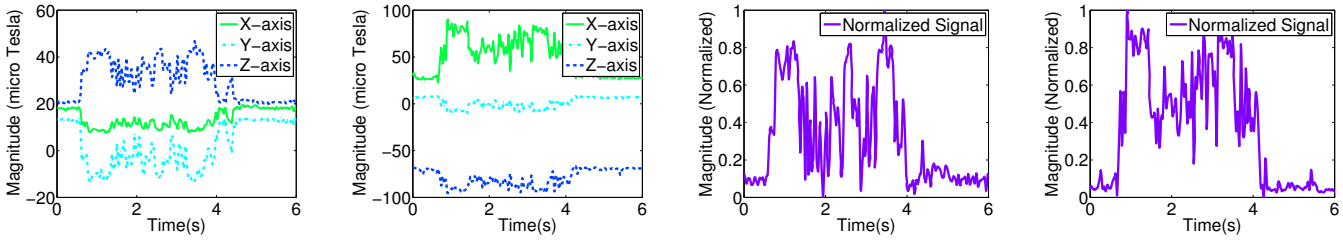
4.2 Launching Detection

In this subsection, we elaborate how to detect the launching process of an application, as the first step of user application recognition.

4.2.1 Earth Impact Reduction

Due to the impact of the earth's magnetic field, the captured EM signals are geo-spatial dependent. Even for an application launched on the same laptop but with different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing



(a) 3-dimensional EM signals (setting 1). (b) 3-dimensional EM signals (setting 2). (c) 1-dimensional normalized signals (setting 1). (d) 1-dimensional normalized signals (setting 2).

Fig. 4: Before vs. after earth impact reduction. Setting 1 and 2 refer to two different geo-spatial locations and laptop-smartphone orientations for launching the same application.

geo-spatial locations or laptop-smartphone orientations, the EM signals can vary a lot. As shown in Fig. 4(a) and 4(b), the 3-dimensional EM signals at two different locations/orientations differ in values on the axis of x, y, and z. `MagAttack` shall eliminate the earth impact to achieve location/orientation-free attack.

The mobile device sensor records surrounding EM signals in three dimensions (x, y, z). The initial EM magnitude of each axis depends on the location of the mobile device, and the changing trend lies with the laptop-smartphone orientation. To eliminate the impact of location and orientation, we utilize the relative change of the EM signals instead of the original data. We assume this change is caused by the launching and thus is consistent with the same application regardless of positions/orientations.

To achieve it, we first centralize the magnetic magnitude of each axis to resolve its relative change to the earth's magnetic field, and then aggregate and normalize the relative changes in all three axes, as shown in Algorithm 1. As a result, we can see from Fig. 4 that after earth impact reduction, the 1-dimensional normalized signals under two different settings become more similar and can be further identified as the same one, as discussed later. In addition, to eliminate the ambient EM interference caused by the running applications on the attack device, we let it run no applications other than magnetic signal collection during attacks.

Note that we pay no special attention to the ambient EM emissions from other electronic devices, since the smartphone's built-in magnetometer can only measure the magnetic induction signals in the near field, which attenuate quickly with distance (around several centimeters). As a result, the magnetic signals from other electronic devices, if not very close, may not affect `MagAttack`.

4.2.2 Pre-screening

EM signals usually remain stable when no application is started but can vary significantly during the process of application launching. To improve the accuracy and efficiency of launching detection, we design a pre-screening algorithm that uses a time window to scan through the EM signals and filter out the time windows that are unlikely to contain the start of an application.

As an application can start at any time, we detect the high variances of the EM signals over a certain time period. A smaller time window and moving step can achieve higher accuracy at the cost of lower detection efficiency. To strike the balance between accuracy and efficiency, we set the time window to be 1 s and the moving step to be 0.1 s. In addition, we utilize the Exponential Moving Average (EMA) approach [22] to update the variance threshold during the period without application launching:

$$\delta_{t+1} = (1 - \alpha)\delta_t + \alpha \times Var(t) \tag{2}$$

where δ_t and $Var(t)$ are the threshold and the EM variance at the time period t, respectively. α represents the degree of weighting decrease and a larger α indicates a more dominant current variance in updating the threshold. In our implementation, we set α to be 0.1. A window W_t is detected when its EM signal variance is substantially larger than the threshold δ_t:

$$Var(t) \geq \beta \times \delta_t \tag{3}$$

where β is the coefficient of the threshold. A larger β indicates that fewer sliding windows will be detected. In our implementation, we set β to be 3. Upon detecting a sliding window with a large variance, we can further classify it with a Support Vector Machine (SVM) based classifier.

4.2.3 Launching Detection

In addition to application launching, other user operations on the laptop may also contribute to high variances of EM signals. To address it, we utilize a SVM based classifier to further refine the launching detection results.

As the CPU instructions involved with a launching operation often take seconds to complete, a 1-second EM trace may not hold enough features to differentiate launching from other operations. Therefore, for each 1-second EM trace detected by the pre-screening algorithm, we append it with the subsequent $k - 1$ one-second EM traces. In our implementation, we choose k to be 4 based on our observation that the variance of EM signals becomes indistinctive after 4 seconds since we start the application. For each k-second normalized EM time series, we smooth it with the Wavelet reconstruction at level 4, and then employ Short Time Fourier Transform and Principal Component Analysis to extract a feature vector. Since we use the same feature extraction techniques for launching detection and application recognition, we defer to present the technical details of feature extraction in the next subsection. Then, we feed the feature vector of each k-second EM trace to a SVM based binary classifier with a kernel type of the radial basis function [23], whose output is whether it is the start of an application. Combined with the pre-screening, `MagAttack`

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

is able to detect application launching accurately and reliably.

4.3 Application Recognition

After detecting the launching of an application, we aim at figuring out what the application is.

4.3.1 Data Pre-processing

After launching detection, we obtain a number of k-second time windows that contain the EM traces of application launching. Hereafter, we use *time window/interval* to represent the EM trace contained in that time window/interval for short. For these time windows, we perform two preprocessing operations: *window expansion* and *window alignment* before feature extraction.

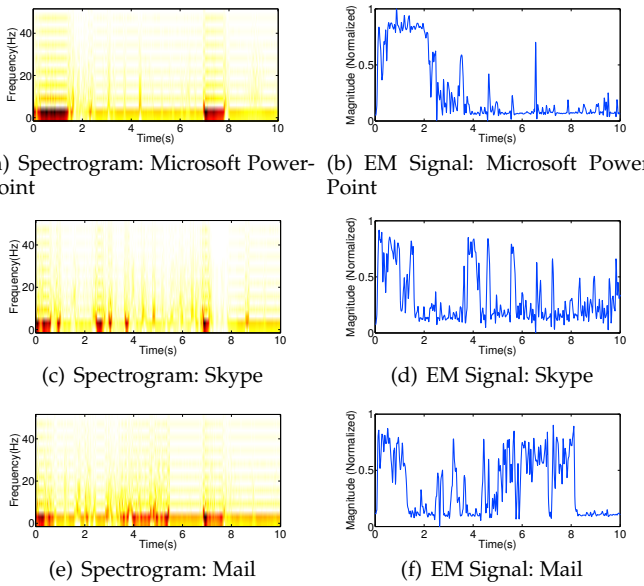
Window Expansion. To guarantee that the selected time window contains sufficient information, even for applications that need a long time to initialize (*e.g.*, more than 10 seconds), we append each aforementioned k-second time window with the subsequent $m - k$ one-second time windows, where $m \geq k$. In our implementation, we choose m to be 10. This expansion can help `MagAttack` achieve higher accuracy by including more features during the launching process.

Window Alignment. Due to the finite granularity of the window sliding approach, the starting time of each window deviates more or less from the ground truth. To reduce the impact of deviations, we align these time windows in two steps. First, we compute the centroid of these time windows using the average Dynamic Time Warping (DTW) scheme [24], which is the time window that has the minimum averaged DTW cost to the others. Then, we use the centroid as the base to shift each other time window in the time series. As a result, each shifted time window has the maximum correlation coefficient with the base signal and is still m seconds.

4.3.2 Feature Extraction

We then extract a feature vector for each aligned time window. We first divide a EM time window into overlapped time intervals and conduct Fast Fourier Transform (FFT) on each interval, which extracts time-variant features in the frequency domain. Then, we conduct the Principal Component Analysis (PCA) [25] on the FFT result of each time interval and obtain the first PCA component, which aggregates features in different frequency scales. Finally, we sequentialize the PCA component of each time interval to construct a feature vector for application recognition.

Short Time Fourier Transform. For each aligned time window, we divide it into time intervals using a sliding window with an interval size of w and a step size of $0.5 * w$. Then, each time interval is zero padded to the length of $2 * w$, before conducting FFT to get the STFT spectrogram. We calculate the abstract value of the FFT results and obtain the first half. As thus, for each time window, we get a $t \times l$ spectrogram matrix S, where t rows correspond to t time intervals, and l columns are the FFT results of that time interval. In practice, we set w = 320 milliseconds. Fig. 5 illustrates the STFT spectrograms of 3 Mac OS applications (Microsoft PowerPoint, Skype and Mail), where the X-axis represents the time interval, the Y-axis represents the frequency, and the color represents the energy of the frequency.



(a) Spectrogram: Microsoft PowerPoint (b) EM Signal: Microsoft PowerPoint

(c) Spectrogram: Skype (d) EM Signal: Skype

(e) Spectrogram: Mail (f) EM Signal: Mail

Fig. 5: STFT spectrum of EM signals reconstructed on the first wavelet level for different applications.

Principal Component Analysis. We then use the PCA to track the correlation of FFT results among different frequencies, and combine them by extracting the first principal component. We conduct PCA on the FFT results of each time interval in three steps: data preparation, coefficient calculation, and feature vector construction.

(1) Data Preparation. Let s be the number of time windows. With STFT, each time window is transformed into a spectrogram matrix with t rows. For each time interval, we extract its FFT results from aforementioned spectrogram matrices to construct a new interval matrix. In this way, we build t interval matrices $H_1, H_2, \ldots, H_t$, and each matrix has s rows.

(2) Coefficient Calculation. For each interval matrix H_i, we calculate its principal component coefficient matrix C_i. Each column of C_i contains coefficients for one principal component and the columns are arranged in the decreasing order of component variance. We then obtain the first principal component of H_i, *i.e.*, the first column of C_i, for feature vector construction.

(3) Feature Vector Construction. We conduct PCA on the spectrogram matrix S to build feature vector V. The i^{th} element of V is calculated as:

$$V(i) = \sum_{j=1}^{l} S(i,j) * C_i(1,j) \tag{4}$$

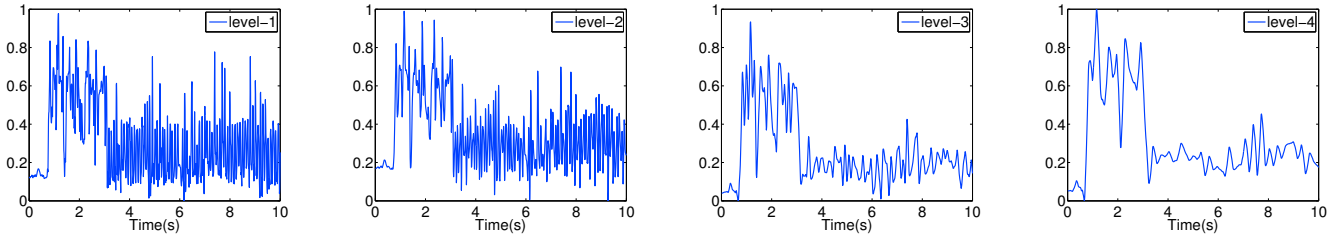
where l is the column number of the spectrogram matrix S as well as the length of FFT results for each time interval.

In this way, for each m-second time window, we extract a feature vector V with t elements, where t is the number of time intervals that the time window is divided into. We envision that this feature vector retains the time-varying frequency features of the EM signals.

4.3.3 Application Classification

Given the feature vectors extracted from the training data, we conduct application classification with supervised learn-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing



(a) Level-1 reconstruction. (b) Level-2 reconstruction. (c) Level-3 reconstruction. (d) Level-4 reconstruction.

Fig. 6: Level 1-4 reconstructed EM signals using the Wavelet MRA.

ing. To select the appropriate classification algorithm, we compare 10 commonly-used classifiers in this paper, which are 1) Logistic Regression, 2) Gaussian Naive Bayes, 3) K-Nearest Neighbors, 4) Linear Discriminant Analysis, 5) Quadratic Discriminant Analysis, 6) Decision Tree, 7) Support Vector Machine, 8) ExtraTrees, 9) Random Forest, and 10) Gradient Boosting. We choose these classifiers since they are widely-employed in website/application/device fingerprinting [26], [27], [28], [29] and we compare them to select the most effective one. Based on the performance of these classifiers (shown in Fig. 8(a)), we employ Random Forest (RF) [30] as our classification method since it outperforms other classifiers. We assume the high performance of Random Forest comes from the capability of dealing with the high-dimension feature vector extracted by PCA and avoiding overfitting.

4.4 Operation Recognition

In addition to application recognition, `MagAttack` attempts a more fine-grained detection, *i.e.*, operation recognition. We analyze two typical scenarios: 1) webpage browsing, and 2) video playing, and regard visiting different web pages or playing different videos as different operations. However, our method is not limited to these scenarios and can be applied to other applications as well.

For operation recognition, we capture the EM signals when an operation is being launched, and extract a feature matrix from its EM signals. Specifically, we use Wavelet Multi-Resolution Analysis (MRA) [31] to get the de-noised signals at N Wavelet levels. Then, we extract a feature vector from each of the N reconstructed signals using the same approach in Sec. 4.3.2, and arrange the N feature vectors in rows to construct a feature matrix. With the obtained feature matrix, we use the same RF classifier in Sec. 4.3.3 to achieve operation recognition.

4.4.1 Wavelet Multi-Resolution Analysis

Launching a user operation, *e.g.*, opening a web page or video, usually requires executing network-related CPU instructions over a short time interval, resulting in EM signals with time-varying frequency characteristics. Especially, EM signals generated by opening a web page are usually more inconsistent than those generated by application launching. The reason is that for applications, CPU instructions executed by different launchings are fairly consistent while for web pages, they are likely to be various as a result of dynamic contents, *e.g.*, pop-out online advertisements.

To address it, we use the Wavelet MRA to de-noise the EM signals of user operations at different granularity scales before extracting time-frequency features. The insight is that, although each time launching a user operation may involve dynamic contents and thus different CPU instructions, the EM signals of the same operation are similar at a coarser granularity scales with subtle differences at a finer granularity. We analyze EM signals of different operations at N granularity scales with N different weights, and N is set to be 5 based on the Shannon entropy-theory [32]. In the following, we elaborate the details of the employed MRA approach.

Wavelet Decomposition. First, we decompose the EM signals generated by operation launching with two complementary filters: a low-pass filter, which generates approximation coefficients, and a high-pass filter, which generates detail coefficients. For an EM time series, we decompose them iteratively from level 1 to level N, and get a coefficient vector D_n as:

$$D_n = (a_n, d_n, d_{n-1}, d_{n-2}, \ldots, d_1) \tag{5}$$

where a_n contains the approximation coefficients at level n and d_n contains the detail coefficients at level n, with $1 \leq n \leq N$.

Wavelet Reconstruction. Then, we reconstruct an approximation signal from level 1 to level N, respectively. For each level n, we calculate the reconstructed signal by upsampling and convolution from level 1 to level n with the approximation coefficient a_n. An illustration is provided in Fig. 6, which shows an increasing order of denoising from level 1 to level 4. After denoising, we extract a feature vector for each of the N reconstructed approximation signals using the same feature extraction approach in Sec. 4.3.2. Then, the N feature vectors are combined in rows to form a feature matrix for operation classification.

5 Performance Evaluation

To evaluate the performance of `MagAttack`, we have conducted experiments with 30 popular applications, 30 YouTube videos, and 50 top websites in China across 60 days. In summary, the performance of `MagAttack` is:

- `MagAttack` achieves a precision of 96.5% and a recall of 91.8% in application launching detection, an average accuracy of 98.6% to recognize 30 applications, an average accuracy of 97.5% to classify 30 YouTube videos, and an average accuracy of 90.4% to classify the 50 top websites in China.
- `MagAttack` can operate with little influence from operating system, sensor model and sampling rate.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

Fig. 7: The experimental scenario of MagAttack. We keep the attack device under the table to draw no attention. The round table is 2.5 cm in thickness.

5.1 Experiment Setup

We conduct MagAttack in a lab with 2 laptops and 2 smartphones. The detailed settings are as follows.

Target Device. We use a MacBook Air laptop as the main target device. In addition, we use a Lenovo T440p laptop to evaluate the performance of MagAttack on various operating systems. The detailed information of each target device is shown in Tab. 2.

Attack Device. We use an iPhone SE smartphone as the main attack device to capture the laptop EM emissions. In addition, we use a Nexus 5 smartphone to evaluate the performance of MagAttack with various attack devices. The detailed information of each attack device is shown in Tab. 3.

Attack Scenario. We utilize the attack smartphone to record the EM emissions when the target laptop is launching different applications or operations. The target laptop is placed on a round table with the attack device attached on the backside to draw no attention, as shown in Fig. 7. The round table is 2.5 cm in thickness. Then, we acquire the measurements from the built-in magnetometer and transfer these EM measurements to a cloud server for further processing.

Application/Operation. For application recognition, we choose 30 popular applications available on Mac OS that cover several popular categories such as productivity, business, entertainment, tool, and social networking. For operation recognition, we take the video player GOM Player and the web browser Chrome as two examples, and use 30 offline videos randomly downloaded from YouTube and the top 50 web sites of China listed in Alexa [33]. For the convenience of data collection, we use a *Python* script to launch applications/videos/web pages iteratively. Each application/web page is being launched for 10 s while each video is being launched for 20 s, with a 5 s blank period between two successive launchings. The reason why videos are launched for a longer time is that it takes more time to collect sufficient data for video recognition.

5.2 Metrics

We use *precision* and *recall* to evaluate the performance of MagAttack on launching detection, and *accuracy* to evaluate the performance of MagAttack on application recognition and operation recognition.

TABLE 2: Summary of experimental laptops.

Machine Type	MacBook Air	Lenovo T440p
OS Version	Mac OS 10.10.5	Win 7 Home
Processor	Intel Core i5 1.4 GHz	Intel Core i5 2.6 GHz
Memory	4-GB DDR3 1600 MHz	8-GB DDR3L 1600 MHz

TABLE 3: Summary of experimental phones.

Phone Type	iPhone SE	Nexus 5
OS	iOS	Android
Sensor Rate	100 Hz	50 Hz

Precision. Precision is denoted as $\frac{TP}{FP+TP}$, where TP represents the true positives, *i.e.*, the number of times that MagAttack correctly detects an application launching. Similarly, FP refers to the false positives, the number of times that MagAttack falsely classifies a time interval as an application launching.

Recall. Recall is denoted as $\frac{TP}{FN+TP}$, where FN is the number of time intervals that contain an application launching but are not detected by MagAttack.

Accuracy. For each application/operation, accuracy is defined as the ratio of the number of correctly recognized samples to the total number of testing samples. We use the average of the accuracy for all applications/operations as the final recognition accuracy of MagAttack.

5.3 Launching Detection Results

We first evaluate the launching detection performance of MagAttack. In this set of experiments, each of the 30 applications is launched for 100 times, with different laptop-smartphone geo-spatial locations and random laptop-smartphone orientations. As a result, we have 3000 application launching events. To detect them, we first employ the pre-screening model to locate candidate time windows that have high probabilities to contain application launching events. The performance achieved by the pre-screening is a recall of 96.1% but a precision of 43.2%, which means a few false positives occur. To discard these false positive candidates, we then use the SVM model to further refine the pre-screening model decision, where we adopt the 10-fold cross-validation scheme to avoid over-fitting. Finally, their combined performance achieves a precision of 96.5% and a recall of 91.8% for application launching detection.

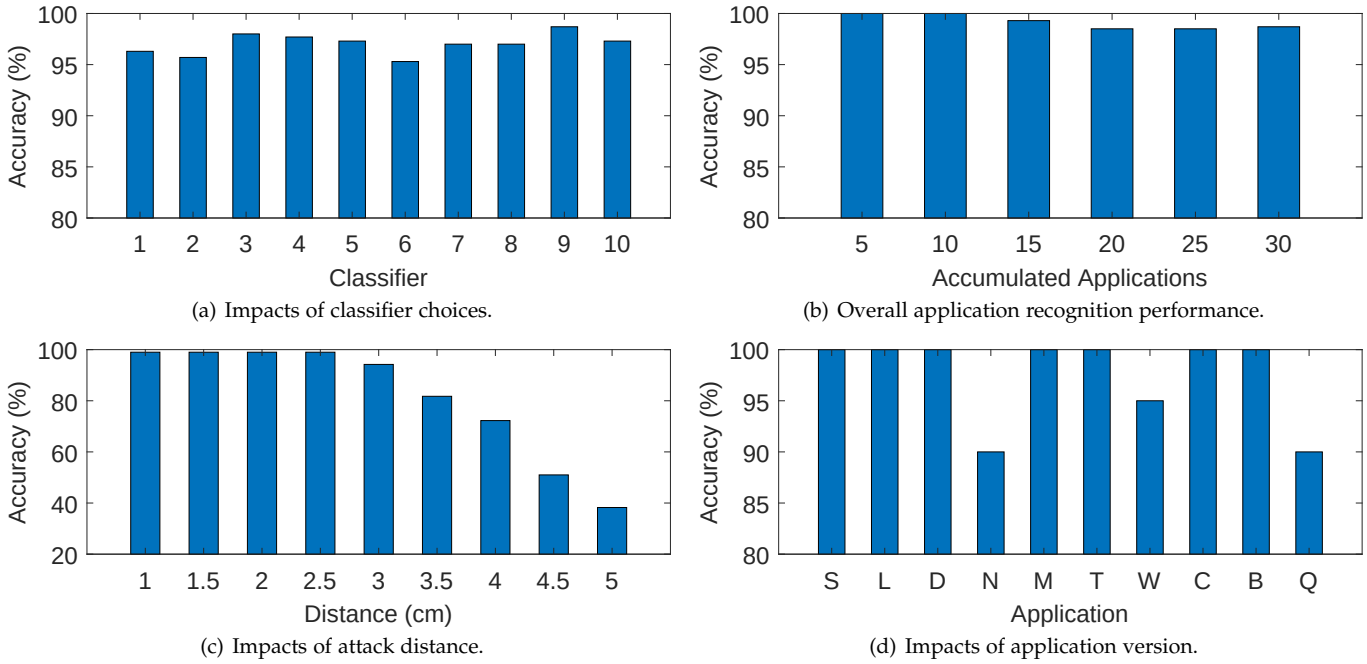
5.4 Application Recognition Results

After launching detection, we obtain a total of 3000 EM samples for 30 applications, each of which lasts for 10 seconds. We then evaluate the application recognition performance of MagAttack with these samples.

5.4.1 Overall Evaluation

To evaluate the performance of MagAttack on application recognition, we first select the appropriate classifier and EM sample length, and then conduct the overall performance evaluation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing



(a) Impacts of classifier choices.
(b) Overall application recognition performance.
(c) Impacts of attack distance.
(d) Impacts of application version.

Fig. 8: Performance of application recognition.

TABLE 4: Impact of EM Sample Length.

Sample Length	Accuracy (%)
2 s	96.5
4 s	98.0
6 s	98.7
8 s	98.7
10 s	98.7

Classifier Choice. To select the appropriate classifier for application recognition, we compare 10 commonly-used supervised learning algorithms. They are 1) Logistic Regression, 2) Gaussian Naive Bayes, 3) K-Nearest Neighbors, 4) Linear Discriminant Analysis, 5) Quadratic Discriminant Analysis, 6) Decision Tree, 7) Support Vector Machine, 8) ExtraTrees, 9) Random Forest, and 10) Gradient Boosting. In this set of experiments, 20 EM samples of each application are used and each EM sample length lasts for 10 s. We employ the 10-fold cross validation to evaluate the performance of classifiers since it can combine measures of fit and thus derive a more accurate estimation for model prediction performance. All the hyperparameters for each classifier are determined by grid search. The results in Fig. 8(a) demonstrate that all the classifiers show an accuracy above 0.9, with the classifier 9) Random Forest, 3) K-Nearest Neighbors, and 4) Linear Discriminant Analysis being the best 3 classifiers. In the following experiments, we employ Random Forest (*RandomForestClassifier* from the scikit-learn library [34]) since it shows the best accuracy. The used hyperparameters are: $n\_estimators$ = 1000, $max\_features$ = 1, $random\_state$ = 666. Other hyperparameters are kept as default.

EM Sample Length. A longer EM sample length may achieve a better recognition accuracy at the cost of a longer data collection time. To strike the balance of attack accuracy and cost, we investigate the appropriate EM sample length for application recognition. We set the length of aforementioned EM samples to be 2, 4, 6, 8 and 10 seconds, and conduct the corresponding 10-fold cross validation, respectively. From the results shown in the Tab. 4, we can observe that the accuracy of `MagAttack` does not change significantly as the length of the EM sample decreases. With 6-second EM samples, `MagAttack` can achieve comparable performance with an average accuracy of 98.7% for recognizing the 30 experimental applications. Even with 2-second EM samples, `MagAttack` can still achieve an average application recognition accuracy of 96.5%. In the following experiments, we employ 6-second EM samples by default.

Overall Performance. With the RF classifier and the appropriate EM sample length, we then evaluate the overall performance of `MagAttack` for recognizing 30 applications. In this set of experiments, we use 20 EM sample of each application for training and 60 samples for testing (they are never used for classifier selection). The detailed results of application recognition are shown in Fig. 8(b), from which we can observe that `MagAttack` achieves an average accuracy of 100% when differentiating 5 and 10 applications, 99.3% for 15 applications, 98.5% for 20 and 25 applications, and 98.6% for 30 applications. With the increasing of applications, the recognition accuracy is not obviously decreased. Overall, `MagAttack` can achieve an average recognition accuracy of 98.6% across the 30 experimental applications, and the training overhead of a single application is 42 ms on average.

5.4.2 Impact of Background Application

When recognizing a launched application, there might be other applications running in the background, which generate EM emissions as well and thus may interfere with the application recognition. To evaluate the impact of background applications, we train the classifier using samples without background applications, and test it using samples collected with one application running in the background. We use 3 background applications in this set of experiments: Microsoft Word, Microsoft PowerPoint, and Safari that are operated by a user from time to time and have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

TABLE 5: Impact of Background Application.

Background Application	Accuracy (%)
Microsoft Word	93.0
Microsoft PowerPoint	94.0
Chrome	94.0
None	98.6

TABLE 6: Impact of Sampling Rate.

Sampling Rate (Hz)	Accuracy (%)
10	84.7
20	96.6
50	98.3
100	98.6

an average CPU workload of 2-3%. We assume those user-involved applications may introduce more interference. The results in Tab. 5 show that MagAttack is slightly impacted by the existence of background applications. Nevertheless, MagAttack can still achieve an average accuracy of 93.7% when recognizing 30 experimental applications with one background application.

5.4.3 Impact of Sampling Rate

The sampling rate of the default attack device iPhone SE is 100 Hz. To investigate the impact of the sampling rates, we test MagAttack by down-sampling the collected EM samples to 10, 20 and 50 Hz, and conduct the corresponding evaluation respectively. From the results shown in Tab. 6, we can observe that the performance of MagAttack slightly drops when the sampling rate decreases. However, with a sampling rate of 50 Hz, MagAttack can achieve comparable performance with an average application recognition accuracy of 98.3%. Even with a sampling rate of only 20 Hz, MagAttack can still achieve an average application recognition accuracy of 96.6%. It provides encouraging signs that MagAttack can be launched even with smart devices that have limited sampling rate capability.

5.4.4 Impact of Device Distance

Due to the limited sensitivity of the built-in magnetometer, the captured EM signals become weak when moving the attack device away from the laptop. To investigate the influence of the attack device placement, we vary the vertical distance between the laptop and the phone. We train the classifier with traces collected at a distance of 2.5 cm, and test it with traces collected from various distances. Starting from 1 cm, we gradually enlarge the distance between the laptop and the phone with a step of 0.5 cm. The performance of MagAttack at each distance is shown in Fig. 8(c), from which we can see that within a distance of 1-3 cm, MagAttack can achieve a high accuracy ($> 94\%$). If the attack distance is beyond this range, we consider to employ specialized hardware to enhance the signal reception capability.

5.4.5 Impact of Application Version

Another factor that may affect the performance of MagAttack is the application version. The update of an application may change its involved instructions and thus its EM patterns. To investigate the impact of application versions, we randomly choose 10 applications and download their old version to conduct experiments. The chosen applications and their experimental versions are summarized in Tab. 7. During the experiments, we train the classifier with data collected from current versions of these applications and test the performance of MagAttack with data collected from old versions of the 10 experimental applications. The results shown in Fig. 8(d) reveal that 7 out of 10 applications can be recognized with an accuracy of $\sim$100%. Overall, MagAttack can achieve an average accuracy of 97.5% in differentiating these 10 applications even the training and testing samples are from different versions.

TABLE 7: Summary of experimental application versions.

ID	App	Version (training)	Version (testing)
S	Skype	7.59	5.8.0.945
L	Slack	4.8.0	4.3.3
D	DingTalk	5.1.15	4.7.27
N	Netease Cloud Music	2.3.2	1.5.10
M	QQ Music	7.1.2	6.3.5
T	Thunder	3.4.1	3.2.6
W	Wunderlist	3.4.21	3.4.20
C	Chrome	72.0.3626.9	84.0.4147.89
B	Baidu NetDisk	3.3.2	2.2.3
Q	QQ	8.3.6	6.6.1

To further reduce the impact of application versions, we assume that MagAttack can include more application versions during the training process. Specifically, the adversary can build an application library that contains common applications and their popular versions, and use the data from the library to train a comprehensive classifier. We believe that this kind of overhead might be affordable for adversaries who attempt to track the behaviors of others.

5.5 User Operation Recognition Results

In the experiments of operation recognition, each user operation is launched for 100 times. For data collection, we record a 10-second EM sample for each webpage-browsing operation and a 20-second EM sample for each video-playing operation.

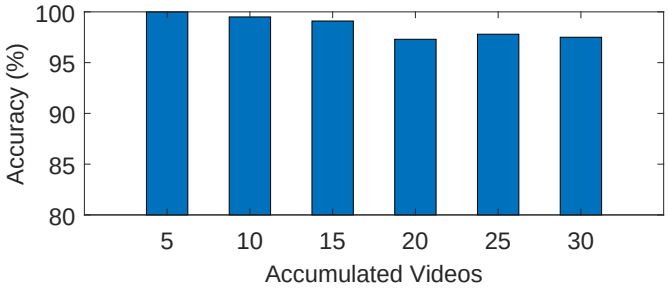
5.5.1 Video Recognition Performance

For video-playing operation recognition, we collect 3000 samples from 30 YouTube videos across 5 days. We use 20 EM samples of each video for training, and use the rest for testing. The results in Fig. 9(b) show the accuracy when MagAttack classifies among different numbers of videos. Specifically, MagAttack achieves an average accuracy of 100% when differentiating 5 videos, and 99.5% for 10 videos. With the increasing of videos, the recognition accuracy slightly decreases. Overall, MagAttack can achieve an average accuracy of 97.5% in differentiating all the 30 videos.

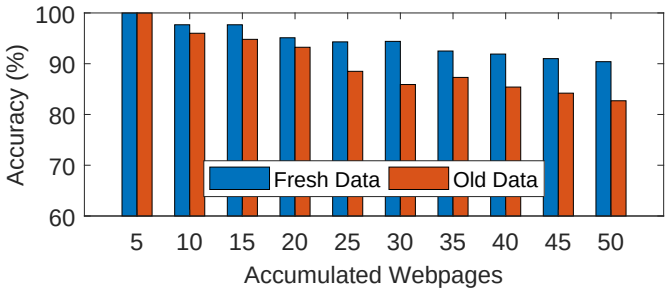
5.5.2 Webpage Recognition Performance

To evaluate the webpage-browsing recognition performance of MagAttack, we collect 5000 samples for 50 web pages across 3 days. The "Fresh Data" in Fig. 9(b) shows the accuracy when MagAttack classifies among different numbers of web pages. Specifically, MagAttack achieves an average accuracy of 100% when differentiating 5 web pages, and 97.8% for both 10 and 15 web pages. With the increasing of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing



(a) Video recognition performance.



(b) Web page recognition performance on fresh and old data.

Fig. 9: Performance of user operation recognition.

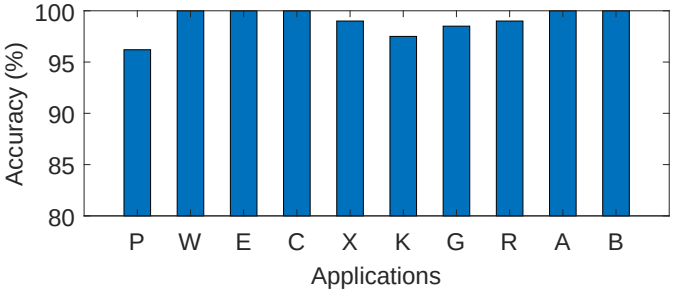


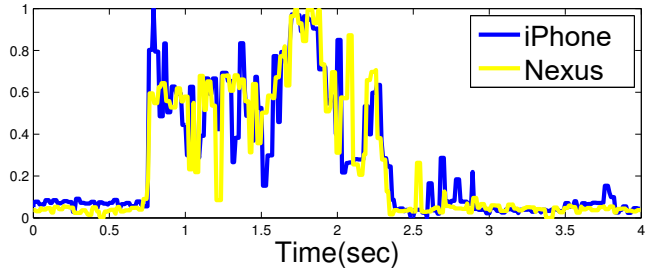
Fig. 10: Performance of application recognition on the Windows laptop.

web pages, the recognition accuracy slightly decreases. Nevertheless, MagAttack can still achieve an average accuracy of 90.4% in differentiating all the 50 web pages.

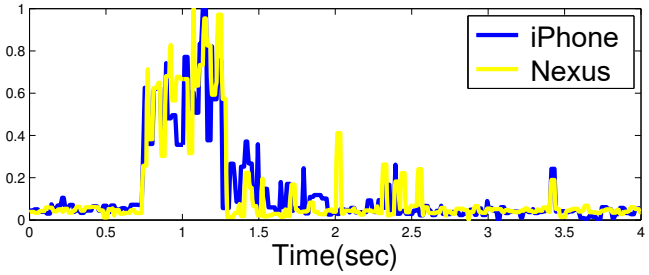
In addition, we investigate the impact of data freshness on the webpage-browsing operation recognition since web pages may update their contents from time to time and thus the data freshness may have impacts. We collect another 500 testing samples 20 days after collecting the training samples. Based on which, MagAttack shows an average accuracy of 82.7% as revealed in "Old Data" in Fig. 9(b). The reason accounts for the performance decrease is that most web pages, unlike applications or offline videos, update their page components on a daily basis. These modified web page components impact the EM signals captured on different days, and thus cause performance decrease.

5.6 Scalability of MagAttack

In addition to the performance of application and operation recognition, we investigate the scalability of MagAttack on various operating systems and attack devices. Specifically, we utilize a target device with operating systems other than Mac OS, and an attack device other than iPhone to perform user application recognition.



(a) Microsoft Word



(b) VLC Player

Fig. 11: EM signals collected using different phones.

5.6.1 Impact of Different Operating Systems

To evaluate it, we conduct experiments on a Lenovo T440p laptop with a Windows OS. Similarly, we collect 50 samples for each of the following 10 popular applications available on Windows OS: PowerPoint (P), Word (W), Excel (E), Chrome (C), Internet Explorer (X), Skype (K), KuGou (G), Windows Media Player (R), Adobe Reader (A) and MATLAB (B), and employ the 10-fold cross validation to evaluate the recognition performance. The results shown in Fig. 10 demonstrate that MagAttack works well on the Windows OS, with an average recognition accuracy of 99.0%. Thus, we have reason to believe that MagAttack is OS-independent and can attack laptops with different operating systems.

5.6.2 Impact of Different Attack Devices

To evaluate it, we use two different mobile phones, *i.e.*, an iPhone SE and a Nexus 5 in Tab. 3, to track the same laptop at the same time. We collect 50 samples for each of the 10 applications and draw the EM signals of two applications (Microsoft Word and VLC player) recorded by two smartphones in Fig. 11 for illustration. From the results we can observe that EM signals of the same application recorded by different smartphones are quite similar. To quantitatively evaluate the impact of different attack devices, we train the system using samples collected by one smartphone, and test it with samples collected by the other device. Since the iPhone SE has a sampling rate of 100 Hz while the Nexus 5 only has 50 Hz, we down-sample the EM signals collected by the iPhone SE to achieve the same sampling rate. The classification results demonstrate that MagAttack can achieve an accuracy of 99.3% in the aforementioned case. Thus, we believe that MagAttack is independent on the model of attack devices as well as the sampling rate of magnetometers.

6 Discussion

In this section, we discuss the defense countermeasure of MagAttack, and the limitations of our system.

6.1 Defense

We propose two defense strategies against MagAttack from both the hardware and software perspectives.

Hardware-based Defense. One condition that enables MagAttack is that the laptop CPU leaks EM emissions which can be captured by a vicinal magnetic sensor. To address it, the electromagnetic shielding on laptops can be enhanced. For instance, the CPU and other vital components can be shielded with metal films, which may reduce a majority of the EM leakage.

Software-based Defense. The root cause of MagAttack is that various instructions generate different EM signals that in turn can be utilized to differentiate user activities. To defend MagAttack, a group of stochastic instructions can be executed by the CPU in the background, which may add random noise to the EM signals generated by user activities, thus interfere the recognition of MagAttack.

6.2 Limitations

Our implementation of MagAttack based on existing hardware has three main limitations. First, the attack distance between the laptop and the mobile device is close, in terms of 3 centimeters. Due to the limited sensitivity of the built-in magnetometer in the mobile devices on today's market, the captured EM signals from the COTS mobile device become too weak after moving the mobile device further away from the laptop. We envision that COTS mobile devices can be equipped with better sensors in the future.

Second, with a smartphone, i.e., a single magnetic sensor, our methods require to precisely align the phone with the target device's CPU. However, we assume it can be mitigated by employing a sensor array as discussed by [21], which can effectively enlarge the EM measurement area and thus reduce the requirement of device alignment.

Third, our algorithms may need to be re-trained in case of different CPU architectures or operating systems. We envision that robust features across laptops of different CPUs and various operating systems may be discovered in the future.

Fourth, currently our method works when one application or operation is performed at a time. We envision that new techniques for recognizing concurrent multiple activities may be developed in the future. We hope this work can attract more effort from the community to explore this field which has not been well studied yet.

7 Related Work

Side-channel Attacks Based on EM Emissions. Pioneer work using EM leakage as the side-channel usually requires customized hardware to capture EM emissions [6], [7], [13], [3], [4]. Genkin *et al.* extract the key of RSA software implementation on a Lenovo laptop using a near-field magnetic probe with a frequency of around 100 kHz [6], [7]. Vaucelle *et al.* detect the existence of ambient electromagnetic fields using a magnetometer bracelet with a frequency of up to 50 kHz [13]. Chen *et al.* detect the usage of electrical appliances by monitoring the device electromagnetic interference (EMI) radiations with an expensive EMI measurement equipment [3]. Chen's scheme works only when the laptop is electrically connected to a power line interface, which is plugged in the wall outlet. Wang *et al.* recognize the electrical appliance usage using a wrist-worn magnetic sensor and a set of data acquisition device, with a sampling rate of 16-bit resolution at 44.1 kHz [4]. In comparison, MagAttack uses magnetometers in the COTS smartphones with a sampling rate of around 100 Hz to detect and recognize user activities on a vicinal laptop. Biedermann *et al.* present a class of EM side-channel attacks on computer hard drives using smartphone magnetic field sensors [35], which detect what type of the operating system is booting up or what application is being started based on the ongoing operations of hard drives. However, Biedermann's scheme cannot work for applications without disk operations. Furthermore, they haven't presented the underlying principle that enables those attacks. In comparison, MagAttack works for any user applications or operations, and investigates the feasibility of such attacks from a view of CPU instructions. Guri *et al.* present a covert channel that can leak data from isolated, air-gapped computers to nearby smartphones by controlling the magnetic fields emanating from the computer by regulating workloads on the CPU cores [36]. In comparison, MagAttack targets at inferring user application/operation using the EM emissions from the CPU when application/operation launched. Ning *et al.* demonstrate a side-channel attack that sniffs mobile Apps based on the correlation between magnetometer readings and LED displays on smartphones [37]. One difference between MagAttack and this work is that MagAttack utilizes the EM emissions from the CPU caused by different CPU instructions of various applications/operations while this work employs the intra-device magnetometer readings caused by different graphic designs of difference Apps. Thus, Ning's scheme works for coarse-grained user applications only but not fine-grained user operations. Matyunin *et al.* propose to identify activities running on mobile devices based on the reaction of intra-device magnetometer sensors to CPU activity [29]. MagAttack differs from this work at three aspects: 1) MagAttack targets at inferring inter-device user applications/operations rather than intra-device ones, 2) MagAttack proposes to use the Wavelet MRA method to deal with the dynamic EM signals of user operations which outperforms simply PCA-based approach used in [29], and 3) MagAttack investigates the feasibility of such attacks at a deeper level of CPU instructions rather than average CPU loads (the former is the root cause of the latter). Another related work of MagAttack is our prior work [14]. In comparison, this work enhances the method for application/operation classification and thus improves the inference accuracy. In addition, this work demonstrates the feasibility of our methods with more applications, shorter EM sample lengths (i.e., shorter collection time), and lower sampling rates, and the applicability to new scenarios such as identifying which video the user is watching.

Side-channel Attacks Using Mobile Devices. Prior work has demonstrated a number of side-channel attacks using the sensors in COTS mobile devices [38], [39], [10], [40], [41]. Xu *et al.* and Schlegel *et al.* show side-channel attacks using cameras [38] and microphones [39], respectively. Cai *et al.* and Aviv *et al.* show that motion sensors in mobile phones, such as accelerometers and gyroscopes, can

be used to learn the user tapping and gesture input [10], [40]. Jana *et al.* recognize web pages that a user browses by tracking the memory footprint variations of the browser on Android [12], which is intrusive since the attackers need to login to the system as a process running in parallel with the browser. These attacks are used to breach the privacy of the mobile device user. In comparison, `MagAttack` breaches the privacy of a laptop user. Zhang *et al.* exploit smartphone magnetometers to recognize nearby household appliances [41]. In comparison, `MagAttack` aims to infer information regarding nearby laptop user's activities.

Side-channel Attacks on Recognizing Laptop User Activities. Prior work has exploited other forms of side-channels than EM emissions for laptop user activity recognition. Zhuang *et al.* and Zhu *et al.* use acoustic signals as the side-channel information to infer user keystrokes [9], [11]. Clark *et al.* utilize the AC power consumption of a laptop to recognize the web page that a user browses [8]. Clark's scheme requires the modification of the power outlet for measuring AC power consumption. Lu *et al.* tap the encrypted web traffic to recognize the web page that a user browses [42]. In comparison, `MagAttack` uses EM emissions as the side-channel information, which is non-intrusive and can be implemented on the COTS mobile devices without hardware modification.

8 Conclusion

In this paper, we propose `MagAttack`, which demonstrates the feasibility of using a COTS mobile device to infer user activities on a nearby laptop based on the EM side-channel leakage from the laptop's CPU. We implement `MagAttack` on the COTS smartphones without hardware modification and evaluate it with 30 commonly used applications, 30 YouTube videos, and 50 top popular web pages in China. The experimental results show that `MagAttack` can conduct launching detection, application recognition, and operation recognition with high accuracy. Future directions include designing specialized hardware to enlarge the detection distance and exploiting robust features across various application versions.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFB0904900, 2018YFB0904904).

References

[1] Gartner, "Gartner says worldwide device shipments will increase 2.1 percent in 2018," https://www.gartner.com/newsroom/id/3849063, 2018.

[2] S. Gupta, M. S. Reynolds, and S. N. Patel, "Electrisense: single-point sensing using emi for electrical event detection and classification in the home," in *Proceedings of the 12th ACM international conference on Ubiquitous computing (Ubicomp'10)*. ACM, 2010, pp. 139–148.

[3] K.-Y. Chen, S. Gupta, E. C. Larson, and S. Patel, "Dose: Detecting user-driven operating states of electronic devices from a single sensing point," in *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communications (PerCom'15)*. IEEE, 2015, pp. 46–54.

[4] E. J. Wang, T.-J. Lee, A. Mariakakis, M. Goel, S. Gupta, and S. N. Patel, "Magnifisense: Inferring device interaction using wrist-worn passive magneto-inductive sensors," in *Proceedings of the 17th ACM international conference on Ubiquitous computing (Ubicomp'15)*. ACM, 2015, pp. 15–26.

[5] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99)*. Springer, 1999, pp. 388–397.

[6] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs," *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 95–112, 2015.

[7] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation," in *Proceedings of the 17th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'15)*. Springer, 2015, pp. 207–228.

[8] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current events: Identifying webpages by tapping the electrical outlet," in *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13)*. Springer, 2013, pp. 700–717.

[9] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Transactions on Information and System Security*, vol. 13, no. 1, p. 3, 2009.

[10] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion." in *Proceedings of the 6th USENIX conference on Hot Topics in Security (HotSec'11)*, vol. 11, 2011, pp. 9–9.

[11] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14)*. ACM, 2014, pp. 453–464.

[12] S. Jana and V. Shmatikov, "Memento: Learning secrets from process footprints," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy (S&P'12)*. IEEE, 2012, pp. 143–157.

[13] C. Vaucelle, H. Ishii, and J. A. Paradiso, "Cost-effective wearable sensor to detect emf," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, 2009, pp. 4309–4314.

[14] Y. Cheng, X. Ji, W. Xu, H. Pan, Z. Zhu, C.-W. You, Y.-C. Chen, and L. Qiu, "Magattack: Guessing application launching and operation via smartphone," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 283–294.

[15] A. Singh, *Mac OS X internals: a systems approach*. Addison-Wesley Professional, 2006.

[16] dtrace.org, "About dtrace," http://dtrace.org/blogs/about, 2017.

[17] Github, "Dtrace-win32," https://github.com/prash-wghats/DTrace-win32, 2016.

[18] Gregose, "Syscall-table," http://syscalls.kernelgrok.com/, 2016.

[19] Wikipedia, "Cmos," https://en.wikipedia.org/wiki/CMOS, 2017.

[20] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of cmos combinational logic networks," in *Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD'92)*. IEEE, 1992, pp. 402–407.

[21] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, "Demicpu: Device fingerprinting with magnetic signals radiated by cpu," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1149–1170.

[22] S. L. Marple, *Digital spectral analysis: with applications*. Prentice-Hall Englewood Cliffs, NJ, 1987, vol. 5.

[23] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.

[24] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, "Dynamic time warping averaging of time series allows faster and more accurate classification," in *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM'14)*. IEEE, 2014, pp. 470–479.

[25] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[26] A. Das, N. Borisov, and M. Caesar, "Tracking mobile web users through motion sensors: Attacks and defenses," in *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS'16)*, 2016.

[27] T. Hupperich, H. Hosseini, and T. Holz, "Leveraging sensor fingerprinting for mobile device authentication," in *Proceedings of the*

13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'16). Springer, 2016, pp. 377–396.

[28] A. Das, N. Borisov, and E. Chou, "Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 1, pp. 88–108, 2018.

[29] N. Matyunin, Y. Wang, T. Arul, K. Kullmann, J. Szefer, and S. Katzenbeisser, "Magneticspy: Exploiting magnetometer in mobile devices for website and application fingerprinting," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, 2019, pp. 135–149.

[30] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition (ICDAR'95)*, vol. 1. IEEE, 1995, pp. 278–282.

[31] A. N. Akansu and R. A. Haddad, *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic Press, 2001.

[32] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Transactions on information theory*, vol. 38, no. 2, pp. 713–718, 1992.

[33] Alexa, "Top sites in china," http://www.alexa.com/topsites/countries/CN, 2017.

[34] 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*, vol. 12, pp. 2825–2830, 2011.

[35] S. Biedermann, S. Katzenbeisser, and J. Szefer, "Hard drive side-channel attacks using smartphone magnetic field sensors," in *Proceedings of the 19th International Conference on Financial Cryptography and Data Security (FC'15)*. Springer, 2015, pp. 489–496.

[36] M. Guri, A. Daidakulov, and Y. Elovici, "Magneto: Covert channel between air-gapped systems and nearby smartphones via cpu-generated magnetic fields," *arXiv preprint arXiv:1802.02317*, 2018.

[37] R. Ning, C. Wang, C. Xin, J. Li, and H. Wu, "Deepmag: Sniffing mobile apps in magnetic field through deep convolutional neural networks," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2018, pp. 1–10.

[38] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, "Stealthy video capturer: a new video-based spyware in 3g smartphones," in *Proceedings of the 2rd ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec'09)*. ACM, 2009, pp. 69–78.

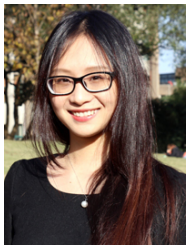
[39] R. Schlegel, K. Zhang, X.-y. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11)*, vol. 11, 2011, pp. 17–33.

[40] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC'12)*. ACM, 2012, pp. 41–50.

[41] M. Zhang and A. A. Sawchuk, "A preliminary study of sensing appliance usage for human activity recognition using mobile magnetometer," in *Proceedings of the 14th ACM international conference on Ubiquitous computing (Ubicomp'12)*. ACM, 2012, pp. 745–748.

[42] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*. Springer, 2010, pp. 199–214.

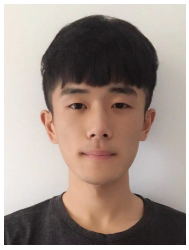
Xiaoyu Ji received his BS degree in Electronic Information & Technology and Instrumentation Science from Zhejiang University, Hangzhou, China, in 2010. He received his Ph.D. degree in Department of Computer Science from Hong Kong University of Science and Technology in 2015. From 2015 to 2016, he was a researcher at Huawei Future Networking Theory Lab in Hong Kong. He is now an associate professor with the Department of Electrical Engineering of Zhejiang University. His research interests include IoT security, including sensor, network, and AI security. He won the best paper award of ACM CCS 2017, ACM AsiaCCS 2018. He is a member of IEEE.



Yushi Cheng is currently a Ph.D. student in the College of Electrical Engineering at Zhejiang University. She received her B.S. degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 2016. Her research interests include IoT security, AI security, and mobile & ubiquitous computing. She received a WST best paper runner-up award in 2017, and an ASIACCS best paper award in 2018.

Wenyuan Xu is currently a professor in the College of Electrical Engineering at Zhejiang University. She received her B.S. degree in Electrical Engineering from Zhejiang University in 1998, an M.S. degree in Computer Science and Engineering from Zhejiang University in 2001, and the Ph.D. degree in Electrical and Computer Engineering from Rutgers University in 2007. Her research interests include wireless networking, network security, and IoT security. Dr. Xu received the NSF Career Award in 2009, a CCS best paper award in 2017, and an ASIACCS best paper award in 2018. She was granted tenure (an associated professor) in the Department of Computer Science and Engineering at the University of South Carolina in the U.S. She has served on the technical program committees for several IEEE/ACM conferences on wireless networking and security, and she is an associated editor of TOSN.

Yuehan Chi is currently a Master sudent in the College of Electrical Engineering at Zhejiang University. He received his B.S. degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 2019. His research intersests include side channel attack, system security and reliability.



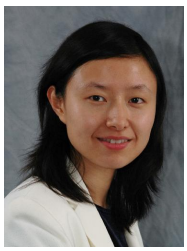
Hao Pan is a Ph.D. student from the Department of Computer Science and Engineering at Shanghai Jiao Tong University. He has received his bachelor's degree from the Yingcai Honors College at University of Electronic Science and Technology of China in 2016. His current research interests reside in mobile computing, with special focuses on wireless communication and sensing, security for mobile systems and mobile human-computer interaction.

Zhuangdi Zhu is a Ph.D. student from the Computer Science department at Michigan State University. She has received her bachelor's degree from the College of Elite Education at Nanjing University of Science and Technology in 2015. She has worked as a Ph.D. intern with Google and Facebook. Her current research interests reside in Machine Learning. Her previous research experience includes systems and wireless networking.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3092209, IEEE Transactions on Mobile Computing

Chuang-Wen You got his Ph.D. degree from the Department of Computer Science and Information Engineering at the National Taiwan University. Currently, he is an assistant professor of the Interdisciplinary Program of Technology and Art (IPTA) and a joint assistant professor of the Institute of Information Systems and Applications (ISA) at the National Tsing Hua University. Before joining NTHU in 2020, he was an assistant researcher of the NTU IoX Center and also a joint assistant professor of the Department of Information Management at the National Taiwan University. His current research directions are human-computer interaction, ubiquitous computing, and sensor networking systems, with a research focus on mobile & wearable healthcare systems, mobile sensing platforms, and IoT-based smart parking systems.

Yi-Chao Chen joined Shanghai Jiao Tong University as a tenure-track Assistant Professor in the Department of Computer Science and Engineering in 2018. He received the B.S. and M.S. in the Department of Computer Science and Information Engineering at National Taiwan University in 2004 and 2006, respectively. He got his Ph.D. in Computer Science at the University of Texas at Austin in 2015. Prior to joining SJTU, he spent a year as a Researcher in Huawei Future Network Theory Lab in Hong Kong and then worked as a Co-founder in Hauoli LLC. His research interests focus on networked systems and span the areas of wireless networking, network measurement and analytics, and mobile computing.



Lili Qiu received the Ph.D. degree in computer science from Cornell University, Ithaca, NY, USA, in 2001. She is currently a professor in the Department of Computer Sciences, University of Texas (UT), Austin, TX, USA. Her research interests include Internet and wireless networking, with special focuses on wireless network performance and management. Before joining UT Austin in 2005, she was a researcher at Microsoft Research for four years. She received the US NSF Career Award (2006), Google Faculty Research Award, ACM Distinguished Scientist. She is a fellow of the IEEE.