Eavesdropping Mobile App Activity via Radio-Frequency Energy Harvesting

Tao Ni, *Shenzhen Research Institute, City University of Hong Kong, and Department of Computer Science, City University of Hong Kong;* Guohao Lan, *Department of Software Technology, Delft University of Technology;* Jia Wang, *College of Computer Science and Software Engineering, Shenzhen University;* Qingchuan Zhao, *Department of Computer Science, City University of Hong Kong;* Weitao Xu, *Shenzhen Research Institute, City University of Hong Kong, and Department of Computer Science, City University of Hong Kong*

https://www.usenix.org/conference/usenixsecurity23/presentation/ni

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

Eavesdropping Mobile App Activity via Radio-Frequency Energy Harvesting

Tao Ni[1,2], Guohao Lan[3], Jia Wang[4], Qingchuan Zhao[2], Weitao Xu[1,2]*

[1]*Shenzhen Research Institute, City University of Hong Kong*

[2]*Department of Computer Science, City University of Hong Kong*

[3]*Department of Software Technology, Delft University of Technology*

[4]*College of Computer Science and Software Engineering, Shenzhen University*

Abstract

Radio-frequency (RF) energy harvesting is a promising technology for Internet-of-Things (IoT) devices to power sensors and prolong battery life. In this paper, we present a novel side-channel attack that leverages RF energy harvesting signals to eavesdrop mobile app activities. To demonstrate this novel attack, we propose *AppListener*, an automated attack framework that recognizes fine-grained mobile app activities from harvested RF energy. The RF energy is harvested from a custom-built RF energy harvester which generates voltage signals from ambient Wi-Fi transmissions, and app activities are recognized from a three-tier classification algorithm. We evaluate *AppListener* with four mobile devices running 40 common mobile apps (e.g., YouTube, Facebook, and WhatsApp) belonging to five categories (i.e., video, music, social media, communication, and game); each category contains five application-specific activities. Experiment results show that *AppListener* achieves over 99% accuracy in differentiating four different mobile devices, over 98% accuracy in classifying 40 different apps, and 86.7% accuracy in recognizing five sets of application-specific activities. Moreover, a comprehensive study is conducted to show *AppListener* is robust to a number of impact factors, such as distance, environment, and non-target connected devices. Practices of integrating *AppListener* into commercial IoT devices also demonstrate that it is easy to deploy. Finally, countermeasures are presented as the first step to defend against this novel attack.

1 Introduction

The last decade has witnessed a surge of Internet-of-Things (IoT) devices ranging from personal wearable smart devices to industrial smart lightening systems. Recently, radio-frequency (RF) energy harvesting, which scavenges electromagnetic energy radiated by ambient Wi-Fi routers [1] and cellular base stations [2] from the remote (e.g., 30 meters [3]) to generate electrical energy, has become promising in daily IoT devices, such as RF-powered sensors [4] and LED lamps [5], to prolong their battery lives. It also holds the promise to bring us one step closer to Nikola Tesla's vision of *powering every device through the air* [6]. Specifically, RF energy harvesting captures ambient radio signals and converts them into electrical power signals that can reveal the wireless data transmission pattern. However, considering the data transmission pattern can leak the activities performed by the mobile device [7–10], it is unclear whether this emerging technique actually constitutes a new attack surface leaking mobile device activities and user privacy due to the lack of investigation.

To take the first step toward filling this gap, this paper aims to demonstrate the feasibility of leveraging this technique to launch a side-channel attack on smartphones to violate user privacy. While there is a line of similar prior researches, they have either exploited different side-channel information, such as embedded sensors [11–13], microphone [14, 15], wireless traffic [9, 16–22], and electromagnetic emanation [7, 23, 24], or studied energy side channels with a number of limitations. For example, some of them assume a prior malicious app injection (e.g., crack battery power profiles [8, 25]), some require physical connection (e.g., USB power cable [10, 26]), and some have to intercept network traces [9, 21].

Compared to the aforementioned side-channel attacks, launching an RF energy harvesting-based side-channel attack toward smartphones has fewer limitations. Since this attack only depends on the RF energy between the victim and the wireless transmitter (i.e., Wi-Fi router) that is entirely captured in the air, the launch of this attack has (i) no need to inject malicious apps, (ii) no requirement for locally physical access to smartphones, and (iii) no intention to intercept network traces. In addition, energy harvesters can also charge themselves when harvesting RF energy, which makes them have limited requirements on the power supply, to launch much more stealthy attacks.

This paper, therefore, presents *AppListener*, the first automated system that demonstrates the feasibility of leveraging RF energy harvesting as a side channel to compromise the privacy of a victim's smartphone activities. In particular, *AppLis-*

*The corresponding author.

tener can recognize which device is running and which app is performing which activity. Specifically, to further alleviate the distance limitation and hide its presence, *AppListener* can recognize a victim device by harvesting RF energy patterns from the Wi-Fi router to which such a device connects. To this end, this paper designs and builds a custom RF energy harvesting module that *AppListener* uses to capture the energy patterns from the Wi-Fi router. Having harvested the energy patterns, it is also non-trivial to recognize a victim's activities in fine-grained with high accuracy. *AppListener* proposes a three-tier classification framework based on Random Forest classifiers to distinguish heterogeneous devices, different apps, and various app activities, respectively.

AppListener is trained over a dataset consisting of 40,000 data samples and evaluated with 40 popular mobile apps (e.g., YouTube, Facebook, and WhatsApp). The training data samples are collected from four users using four different smartphones and four different Wi-Fi routers in four typical environments (i.e., home, office, hallway, and cafe), and apps used for evaluation belong to five categories (i.e., video, music, social media, communication, and game), each of which contains five application-specific activities and running in four mobile devices of different brands. Experiment results show a series of promising capabilities of *AppListener*. First, *AppListener* achieves high accuracy in distinguishing four mobile devices (99.8%). Second, it can recognize various mobile apps (98.8% accuracy in recognizing 40 apps). Third, it can also identify a number of different application-specific activities (86.7% accuracy on 25 activities). In addition, we conducted a comprehensive study to analyze the practical impact factors, such as different sample frequencies and power consumption, noise levels in different environments, and distances between the router and attacker devices in multi-victim scenarios. Our results show that *AppListener* well balances the effectiveness and energy consumption and is resilient to many impact factors to a certain degree.

Ethical Consideration. We have taken ethical considerations seriously in every step of our research with the highest priority. In particular, this study has been approved by the Institutional Review Board (*IRB*) allowing us to recruit volunteers to participate in our experiment and collect their data for analysis.

Contributions. In short, our core contributions are summarized as follows:

- **Novel Side-channel Attack.** This paper presents and demonstrates the feasibility of a novel side-channel attack that leverages RF energy harvesting signal from the Wi-Fi router to which a victim's mobile connects to eavesdrop its fine-grained mobile app activities.
- **Novel Techniques.** This paper proposes a novel automated system, *AppListener*, that analyzes the harvested voltage signal to eavesdrop the fine-grained mobile app activities from a certain recognized mobile device using a novel three-tier classification framework.

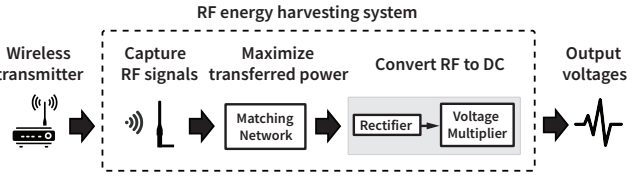


Figure 1: Illustration of an RF energy harvesting system.

- **Comprehensive Evaluation.** *AppListener* is evaluated with 40 popular mobile apps, four smartphones, four Wi-Fi routers in four different environments having multiple victims. It is also evaluated with a set of impact factors. Our results show its high effectiveness and resilience to many practical impact factors.

2 Motivation and Threat Model

2.1 Primer on RF Energy Harvesting

RF energy harvesting is the process that converts electromagnetic energy, e.g., energy that is radiated by Wi-Fi routers [1] and cellular base stations [2], into electrical energy, e.g., direct current (DC) voltage, that can be used to power batteries and electrical devices [27]. As an example, Figure 1 shows the workflow of a typical RF energy harvesting system. When transmitting wireless signals, the wireless transmitter, e.g., the Wi-Fi router, radiates electromagnetic energy into the ambient wireless channel [28]. The receiving antenna on the RF energy harvester captures the RF signal and feeds it into a matching network that maximizes the harvesting efficiency. Then, an RF-to-DC circuit, which is made up of a rectifier and a voltage multiplier, converts the RF signal into DC voltage that can be used to charge batteries or power electrical devices. Theoretically, if we assume the transmitting power of a wireless transmitter is P_t, according to the Friis' transmission equation [29], the received power P_r at the RF energy harvester follows:

$$P_r = P_t \frac{G_r G_t \lambda^2}{(4\pi d)^2} |cos(\theta)|^2, \quad (1)$$

where λ is the signal wavelength, d is the distance between transmitting antenna and receiving antenna, θ is the angle between the two antennas, G_t and G_r represent the gain of transmitting antenna and receiving antenna, respectively. We can notice that the impact of different factors, such as P_t, G_t, and G_r, on the harvested power P_r is linear. Therefore, we can use normalization to mitigate the impact of these factors (details are discussed in §4.3).

2.2 A Motivating Example

Below, we present an example to motivate our work. We consider the scenario where a user connects his/her smartphone to a Wi-Fi network, and opens an app that requires service from the server. We select a set of popular apps and build an RF energy harvester (design details are shown in §3.4) to conduct

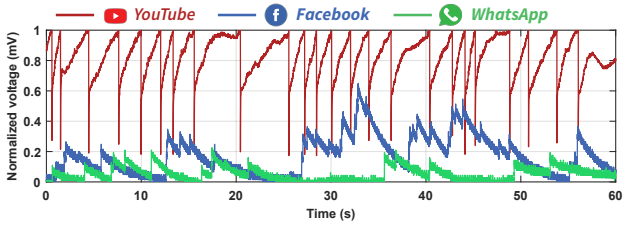


Figure 2: Harvested voltage signals of YouTube (video), Facebook (social), and WhatsApp (communication).

a preliminary study. Using the harvested voltage signal, we examine if it is possible to (i) classify the category of the app, (ii) recognize apps that belong to the same category, and (iii) classify different activities of a certain app.

(i) Classifying app category. First, we consider three categories of apps and examine their energy harvesting patterns. We use an iPhone 11 to run YouTube, Facebook, and WhatsApp, which represents video, social, and communication applications, respectively. More specifically, we run the three apps separately: we play videos continuously on YouTube; scroll and comment posts on Facebook; and type and send chat messages on WhatsApp. Figure 2 shows the harvested voltage signals of the three apps. The signals are distinct from each other, which indicates that *the newly discovered side channel could be used to differentiate app categories*.

(ii) Recognizing apps from the same category. We then select three popular apps, i.e., YouTube, Netflix, and TikTok, from the video category to investigate the feasibility to recognize apps from the same category. We run the apps with three common activities, i.e., playing video, fast forwarding, and switching to next. Figure 3 shows the associated harvested voltage signals when playing these three apps, respectively, where we can observe their overall voltage patterns are significantly different. This is because the three apps differ in the characteristic of the media content, data streaming algorithm used, and how the algorithm is implemented by the service provider. Thus, as shown in Figure 3, YouTube has the highest harvested voltage level among the three examined apps. *This observation demonstrates the feasibility of the side channel to classify apps in the same category*.

(iii) Classifying different activities of a certain app. Finally, Figure 3 also indicates that different activities of a certain app have different voltage patterns, which can be used to perform *fine-grained app activity recognition.*

Key insight. Our key insight is that apps differ in their data traffic behaviors, i.e., what and how the multimedia content is delivered from the content/service provider to the mobile client, which further affects the down-link wireless transmission patterns of the Wi-Fi router and the energy harvested by the RF energy harvester. For instance, video apps, such as YouTube, Netflix, and TikTok, employ the Adaptive BitRate (ABR) streaming for content delivery [30, 31]. The ABR algorithm uses either concurrent TCP or QUIC/UDP flows to deliver multiple data chunks simultaneously. In the

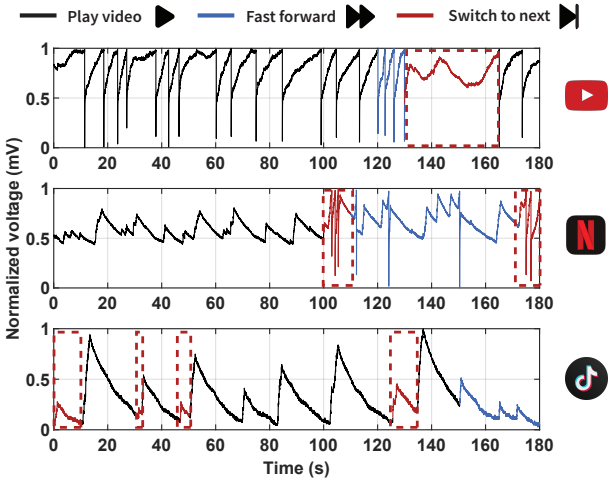


Figure 3: The harvested voltage of three apps in the video category.

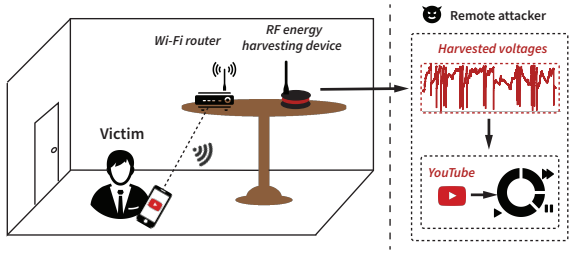


Figure 4: An eavesdropping attack example of *AppListener*.

meantime, the mobile client uses a buffer to store the received chunks [32]. When the buffer level is low, the client requests data chunks as fast as the network can deliver to increase the buffer level, and thus, lead to heavy data traffic and frequent wireless transmission from the Wi-Fi router. On the contrary, video and communication apps, such as WhatsApp and WeChat, generate light data traffic to send text and image messages between users, which lead to a modest wireless transmission frequency. Similar principles are also applied in different activities of a certain app.

2.3 Threat Model

Figure 4 illustrates a typical scenario of our proposed RF energy harvesting-based eavesdropping attack. Specifically, when a victim plays an app on his/her smartphone connecting to a Wi-Fi router, an attacker can remotely eavesdrop his/her mobile app activities by placing an RF energy harvesting-equipped IoT device (e.g., smart power-switch) near the Wi-Fi router in a close proximity (e.g., 1.5 m) and analyzing the harvested voltage signals. This scenario is rational because, considering the app activities are monitored from the corresponding energy radiation from Wi-Fi routers, (1) people usually prefer Wi-Fi to cellular if a Wi-Fi network is available in the ambient environment; (2) the vast majority of mobile apps today require network interactions with the server during operation; and (3) a small IoT device locating near the router is highly likely to be neglected. In addition, we assume the routers adopt fixed transmission power to transfer data.

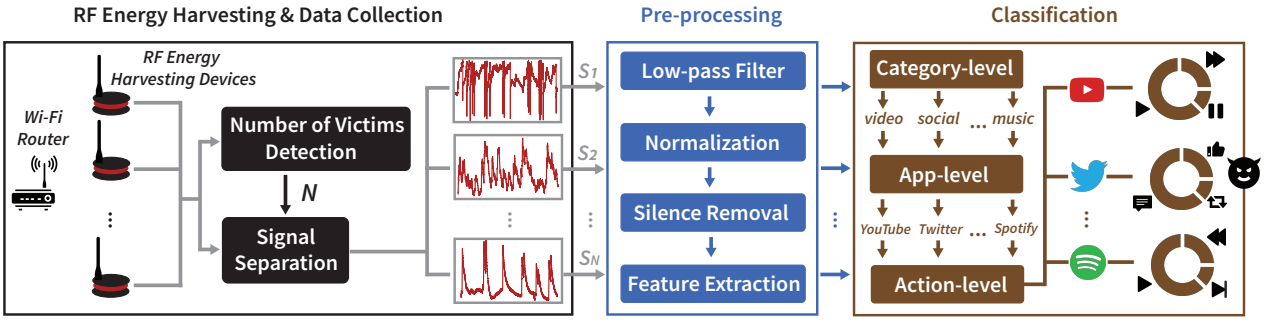


Figure 5: Overview of *AppListener*.

Moreover, we assume attackers leverage M RF energy harvesters to monitor smartphone app activities of N victims ($M \geq N$), and these attackers are either malicious IoT device manufacturers or individuals who have interests in monitoring victims' app activities and can to use RF energy harvesters. For example, malicious manufacturers can build their own RF energy harvesters running malicious firmware or malware. Individual attackers can physically modify commercial IoT devices to flash malicious firmware or trigger the firmware update process, if applicable, e.g., update over-the-air mechanism (OTA) [33–35], from the companion app of that IoT device to replace the firmware with a malicious one.

3 Attack Framework

Figure 5 shows the overview of *AppListener*. We consider the scenario where the attacker leverages M RF energy harvesters to monitor N victims' smartphone app usage ($M \geq N$). After receiving the harvested signals, the first step of *AppListener* is to determine the number of victims N. Since the recorded signal by each energy harvester is a mixture of radio signals produced by N victims, *AppListener* applies a source separation technique to separate the harvested signals into N independent signals, where each signal corresponds to one victim. Then, each separated signal will be fed into the same pre-processing, feature extraction, and classification module to recognize the label of the app. If there are N victims, N labels will be generated by the system. Finally, after obtaining the app labels, *AppListener* will further analyze the specific activities a victim has performed in the corresponding app.

3.1 Determine the Number of Victims

There may exist multiple victims accessing the same Wi-Fi at the same time in the real-world scenarios. Therefore, the first step of *AppListener* is to detect the number of victims in the environment. If multiple victims exist, the voltage signal generated by the RF energy harvester is incurred by multiple radio transmissions (i.e., the down-link wireless transmissions between the Wi-Fi router and multiple victims' smartphones). We propose a detection method based on principal component analysis (PCA) [36]. Specifically, suppose the harvested voltage signal is a time series $S = [s_1, s_2, ..., s_l]$, where l is the signal length, and s_n is the harvested voltage signal at time n, we calculate the Hankel-form matrix of S as follows [37]:

$$H_{harvested} = \begin{pmatrix} s_1 & s_2 & \cdots & s_n \\ s_2 & s_3 & \cdots & s_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_m & s_{m+1} & \cdots & s_{m+n-1} \end{pmatrix}_{m \times n} \quad (2)$$

The dimension of $H_{harvested}$ is $m \times n$, where $l = m + n - 1$ and $m \geq n$. Since the harvested voltage signal is a mixture of signals harvested from multiple RF sources, we aim to analyze the contributions of different RF sources on the harvested voltage signal. To this end, we decompose the matrix $H_{harvested}$ into two orthogonal sub-spaces $U = [u_1, u_2, ..., u_m] \in \mathbb{R}^{m \times m}$ and $V = [v_1, v_2, ..., v_n] \in \mathbb{R}^{n \times n}$ using singular value decomposition (SVD) [38]:

$$H_{harvested} = U \sum V^T = \sum_{i=1}^{r} \delta_i u_i v_i^T, \quad (3)$$

where $\sum \in \mathbb{R}^{m \times n}$ is a diagonal matrix composed of singular values arranged in a descending order $\delta_1 \geq \delta_2 \cdots \geq \delta_r \geq 0$.

According to PCA, the original signal, i.e., $H_{harvested}$, is projected into different sub-spaces, and the singular values represent the weights of different sub-components after the projection. Theoretically, if there is one single victim, the weight of the first singular value should be significantly larger than the others. Similarly, if there are two victims, the weights of the first two singular values should be significantly larger than the others. Accordingly, we train a Random Forest (RForest)-based classifier and leverage it to detect the number of victims based on the weights of the first few largest singular values.

3.2 Signal Separation

After determining the number of victims N, *AppListener* separates the harvested signal into N independent signals where each signal corresponds to one victim. To achieve this, we propose a method based on blind source separation (BSS) [39], which is a widely used technique that separates a set of source signals from a set of mixed signals, without the requirements of prior knowledge about the source signals or mixing process [40].

We consider the scenario where N victims access the same Wi-Fi network simultaneously, and M RF energy harvesters are deployed near the Wi-Fi router ($M \geq N$). Let

Algorithm 1: Signal Separation Algorithm

Input: N: Number of desired components (victims). $Y \in \mathbb{R}^{N\times L}$: Observed L-length voltage signals from N devices.

Output: $A^{-1} \in \mathbb{R}^{N\times N}$: Inverse mixing matrix. $X \in \mathbb{R}^{N\times L}$: Independent voltage signals.

1 Initialize an empty array A^{-1}
2 **for** $i \leftarrow 1$ **to** N **do**
3 Initialize a random N-length vector a_i
4 **while** a_i *is not converged* **do**
5 $a_i^* = \frac{1}{L} Y g(a_i^T Y)^T - \frac{1}{L} g'(a_i^T Y) 1_L a_i$ // 1_L is a L-dimension column vector of 1's
6 $a_i^* = a_i - \sum_{j=1}^{i-1} (a_i^T a_j) a_j$
7 $a_i = \frac{a_i^*}{\|a_i^*\|}$
8 **end while**
9 $A^{-1} = [a_1, a_2, \ldots, a_i]$, if converged, add to A^{-1}
10 **end for**
11 $A^{-1} = [a_1, a_2, \ldots, a_N]$, obtain inverse mixing matrix.
12 $X = A^{-1}Y$, calculate independent voltage signals.

$Y = [y_1, y_2, \ldots, y_N]^T$ denote the observed time-series voltage signals from N energy harvesters (randomly chosen from all M harvesters), our goal is to recover the independent voltage signals $X = [x_1, x_2, \ldots, x_N]^T$ incurred by each victim. If we assume A denote the unknown mixture process, the original independent signals X can be obtained by $X = A^{-1}Y$.

To obtain the well-conditioned inverse mixing matrix A^{-1} and the independent voltage signals X, we design a signal separation algorithm based on FastICA [41]. As shown in Algorithm 1, it first defines an empty array for recording the unmixing vectors in A^{-1} (Line 1). In the subsequent iterations, it finds the converged unmixing vector a_i by implementing an non-quadratic function $g(u)$ such as $tanh(u)$ (Line 4-8). Then, the unmixing vectors form the inverse mixing matrix A^{-1} (Line 9). In the end, we use the obtained A^{-1} to calculate the matrix X that contains independent components (Line 12).

3.3 Signal Pre-processing and Classification

Pre-processing. After obtaining the independent voltage signal induced by each of the victims, we apply a low-pass Savitzky-Golay (S-G) filter [42] to remove the high-frequency noise and smooth the voltage signal. We choose the S-G filter because it ensures a high signal-to-noise ratio while maintaining the original time and frequency domain patterns of the signal [43]. As mentioned in §2.1, the distance and the angle between two antennas largely affect the harvested voltage. As such, we normalize the filtered signals to mitigate these two impacting factors. Moreover, considering the existence of inactive wireless connection periods when using an app, such as the time a victim is reading messages in WhatsApp or a long post in Facebook, we use a threshold to detect and remove the corresponding inactive periods from the voltage signal to clean the data for next steps.

Feature selection and extraction. After the signal pre-processing, we apply a four seconds sliding window [38, 44] with 50% overlapping on the voltage signal. For each sliding window, we extract a total of 31 features from both time-domain (28 features) and frequency-domain (3 features) as shown in Table 6 (in Appendix). The extracted features are used to train machine learning classifiers. Furthermore, we exploit the Recursive Feature Elimination (RFE) [45] as the feature selection algorithm to reduce model complexity and improve recognition accuracy. The features selected for classifier training are also shown in Table 6 (in Appendix).

Three-tier classification. To achieve fine-grained application monitoring, we design a three-tier classification framework. As shown in Figure 5, we first perform a coarse-grained category-level classification to determine the category of the app. Then, based on the output, we perform fine-grained app-level classification. After recognizing the app, the action-level classifier recognizes the specific actions of the victim when using the app. For all classifications, we choose Random Forest classifier (RForest) which achieves the highest accuracy among classifiers we examined (more details in §4.2).

3.4 Implementation

We leverage a set of tools including MATLAB Signal Process Toolbox (Version 8.6) and Python scikit-learn (Version 1.1) to implement the signal pre-processing, feature extraction, and the classification modules of *AppListener*. For the RForest classifier, we set the number of trees as 100 and the maximum depth as 32. In addition, we build a custom RF energy harvester and develop a portable data collection device.

3.4.1 Custom-built RF Energy Harvester

Our custom-built RF energy harvester consists of three parts: an antenna that captures the RF signals, a T-topology matching network that maximizes the efficiency of the energy harvesting, and an RF-DC circuit that converts RF energy to DC voltages. In particular, the matching network consists of two inductors (i.e., L_1 and L_2) and one capacitor (i.e., C_2), and the RF-DC circuit is composed of one RF-DC converter (i.e., PCC110) and one capacitor (i.e., C_1). The internal design of this harvester is shown in Figure 23a in Appendix and the total cost of our RF energy harvester is less than ten dollars (the inventory is provided in Table 7 in Appendix).

In the current prototype, we select a 2.3 dBi dipole antenna with the impedance of 50 Ω to capture the 2.4 GHz Wi-Fi signal and we choose to configure L_1=100 nH, L_2=470 nH, and C_2=0.01 pF for the T-topology matching network placed between the antenna and the RF-DC circuit. In respect of the on-chip RF-DC converter, we use the PCC110 produced by Powercast company due to its high efficiency. The output voltage of C_1 is used as the final output. To ensure a noticeable change in the harvested voltage signal, the capacitance of capacitor C_1 is 1 µF. Note that the RF energy harvester is designed to monitor the RF energy of all channels emitted by the target router at the same times.

Table 1: List of five categories, and 40 popular mobile apps with five app-specific user activities used for evaluation.

Video Apps				Activity		
YouTube	TikTok	Netflix	Vimeo	Play	Next	Pause
Hulu	TED Talk	Disney+	Twitch	Forward	Backward	
Music Apps				**Activity**		
Spotify	Apple Music	YouTube Music	SoundCloud	Play	Next	Pause
Shazam	Netease Cloud	Kugou Music	QQ Music	Forward	Backward	
Social Media Apps				**Activity**		
Facebook	Twitter	Instagram	LinkedIn	Repost	Refresh	Share
Reddit	Pinterest	Quora	Sina Weibo	Thumb-up	Comment	
Communication Apps				**Activity**		
WhatsApp	Line	Telegram	Messenger	Text	Images	Videos
WeChat	Snapchat	Hangouts	Discord	Send voice	Video call	
Game Apps				**Activity**		
PUBG	Minecraft	Arena of Valor	FIFA	Loading	Entering	Gaming
Genshin	Hearthstone	LoL Wild Rift	UNO	Matching	Exit game	

3.4.2 Portable Data Collection Device

We also design and develop a portable system called *Burger Model* to collect and record harvested voltages. The *Burger Model* consists of three layers. The top layer is a self-designed expansion board that is composed of the RF energy harvester, an amplifying circuit, and a Bluetooth module. The amplifier circuit achieves mV-level amplification and extends the eavesdropping distance. The Bluetooth module is used for data communication so the attacker can control the *Burger Model* remotely. The middle layer is a SD card shield with a 16 GB storage that can store harvested voltage signals. The bottom layer is an Arduino Uno board that controls the upper two layers. Note that the entire model can be made smaller by using tiny antennas and a system-on-chip board (i.e., Raspberry Pi Zero W) as shown in Figure 24 in Appendix.

4 Evaluation

4.1 Evaluation Setup

Using the Burger model, we conduct a comprehensive evaluation of *AppListener*. The setup of the experiment is shown in Figure 6 where we consider four common attacking environments: home, office, hallway, and cafe. For home and cafe, we assume there is only one Wi-Fi router and it is eavesdropped by the attacker. For office and hallway, we assume there are multiple Wi-Fi routers. The victims connect their mobile devices to the targeted router, while the other routers act as interfering devices. In addition, in all environments, three *Burger Model* devices ($M = 3$) are placed 0.5 m away from the targeted Wi-Fi router. There is no constraint on the locations of the victims.

App selection and data collection. As shown in Table 1, we select 40 commonly used smartphone apps from five popular categories: video, music, social, communication, and game, where each category has eight representative apps and is associated with a set of five user activities. We recruit four volunteers[1] (two females and two males; all subjects use their smartphones every day) who are asked to play certain selected apps in different circumstances for 1,500 s. The sampling rate is set to 50 Hz, and the collected voltage signal is stored in the SD card of *Burger Model*. In addition, during the data collection, target mobile devices only run the target app alone disabling background non-system services and apps.

[1]The study is approved by our institution's IRB (No. H002554).

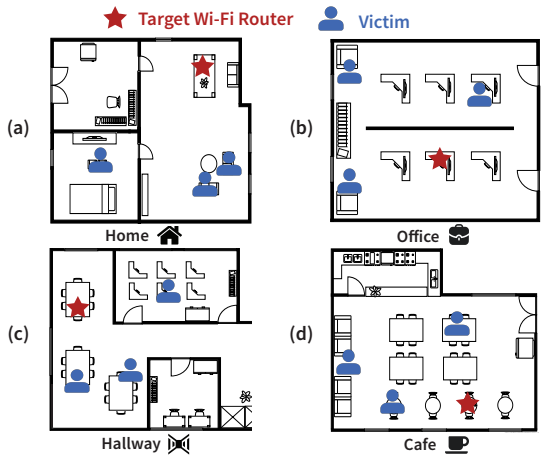


Figure 6: Experimental setups.

4.2 Effectiveness Evaluation

4.2.1 Effectiveness of Victim Detection

To evaluate the effectiveness of the victim detection module (§3.1), we investigate the relationship between the number of victims and the weights of the first few largest singular values. Figure 7 (1), (2), and (3) show the weights of the first three largest singular values for the case when there are one, two, and three victims accessing the Wi-Fi router, respectively. In particular, if there is only one victim, as shown in Figure 7 (1), the largest singular value δ_1 dominates the harvested signal. If there are two and three victims, as shown in Figure 7 (2) and (3), respectively, the weight of δ_1 decreases and the weights of δ_2 and δ_3 increase as more victims present. Specifically, for the single-victim case, we calculate δ_1 when the user is using different apps. We test 40 common apps for this case. Similarly, for the two-victim and three-victim cases, two/three participants use two/three apps simultaneously on their smartphones. We examine 40 combinations of common apps in each case. Figure 7 (4), (5), and (6) show the distributions of δ_1, δ_2 and δ_3 when different number of victims are involved, which indicate that we can use the weight of different singular values to detect the number of victims. Based on the above analysis, we design a Random Forest (RForest) based classifier to detect the number of victims, which achieves 99.8% accuracy in determining the number of victims when there are less than four victims in the same environment.

4.2.2 Effectiveness of Classification

Comparison of different classification algorithms. As mentioned in §3.3, we choose the Random Forest for clas-

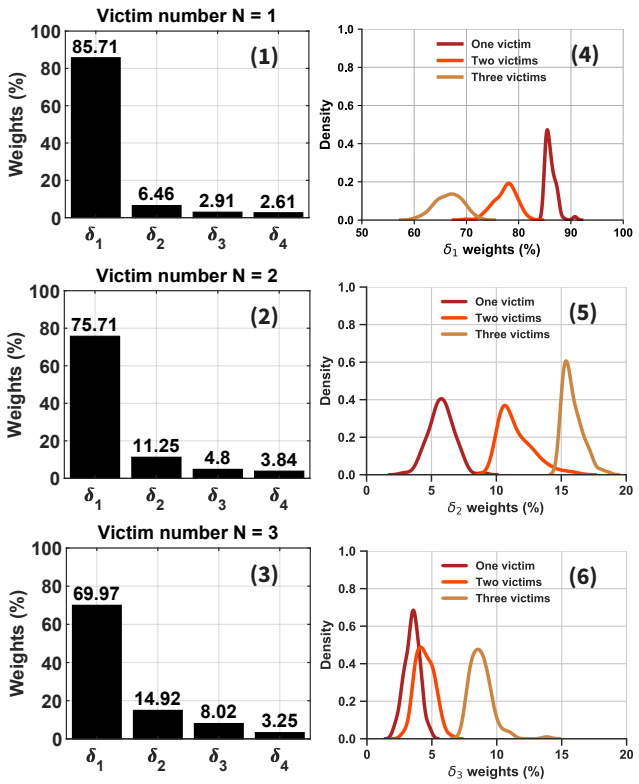


Figure 7: The singular values when there are different number of victims and the corresponding density distribution in the same environment.

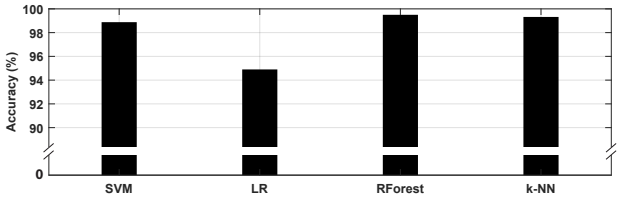


Figure 8: Accuracy (%) of different classifiers in identifying the category of the app.

sification. In this evaluation, we present the comparison of four widely used conventional machine learning classifiers, including Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RForest), and k-Nearest Neighbor (k-NN), to support our decision. The parameters of the classifiers are well-tuned to achieve the highest possible accuracy. Specifically, for the SVM classifier, we choose linear kernel function, and the soft margin constant is set to 10. For LR classifier, the range of penalty parameter C is $\{1, 10, 100, 1000\}$. For the RForest classifier, we set the number of trees as 100 and the maximum depth as 32. For the k-NN classifier, we set the number of nearest neighbors as 10. For each classifier, we perform the 10-fold cross-validation during the evaluation, where nine folds are used as the training data, and the remaining one fold is retained as the testing data.

Figure 8 shows the accuracy of *AppListener* in recognizing the category of the app (i.e., category-level classification).

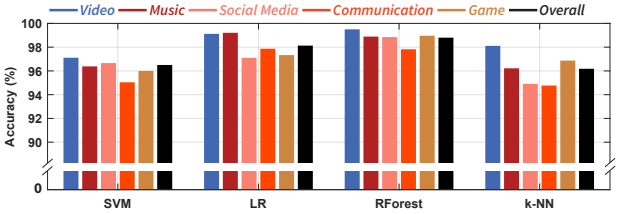


Figure 9: Accuracy (%) of different classifiers in identifying the app. The results are averaged over the apps in each of the five categories.

All the four examined classifiers achieve over 98% accuracy. Figure 9 shows the accuracy in recognizing the label of the app (i.e., app-level classification). Note that the results are averaged over the eight apps in each category. All four classifiers achieve over 96% accuracy in app-level classification. Moreover, for the best classifier, i.e., RForest, the overall accuracy averaged over the 40 apps is over 98%. *Therefore, we opt to use the RForest as our classifier.*

Effectiveness in the single-victim scenario. In this single victim scenario, the victim uses iPhone 11 as the mobile device and plays each of the 40 apps for 1,500 s. As mentioned above, we train the RForest classifier for each of the 40 examined apps to perform the fine-grained action-level classification to recognize the victim's activities for a specific app. The recognition results are shown in Figure 10. Overall, *AppListener* achieves over 85% accuracy in recognizing the user's activity. We also present the detailed recognition results for each of the 40 apps in Table 9 (in Appendix).

In particular, as can be seen in Figure 9, the classification accuracy of categories "Social" and "Communication" are lower than the others. To understand the reasons, we plot their corresponding confusion matrices in Figure 11. We find that the True Positive Rate (TPR) of "Communication" apps WhatsApp and Line is 86% and 89%, respectively, which is much lower than that of others ($\geq$ 96%). Similarly, "Social media" apps such as Facebook and Weibo also have relatively lower TPR. To find out the reason, we analyze the corresponding harvested signals of these apps. We find that "Communication" and "Social" apps exhibit similar patterns in the harvested voltage signals, which lead to classification errors. For example, Figure 12a shows the harvested voltages when two victims are using WhatsApp and Line, respectively. We can find similar patterns in the time windows from 10 s to 13 s and 24 s to 28 s. The same issue happens among "Social" apps as well (shown in Figure 12b), where the harvested voltage signals of Facebook and Weibo contain segments that also exhibit similar patterns. This is because that apps from the same category usually share similar implementation in content delivery (e.g., YouTube and Netflix employ the ABR algorithm for data streaming [30, 31]), which result in similar network communication behaviors making their energy harvesting patterns similar.

Effectiveness in the multi-victim scenario. In multiple victims scenario, *AppListener* first separates the signals into N signals $S_1, S_2 \ldots, S_N$ and use these signals to recognize the

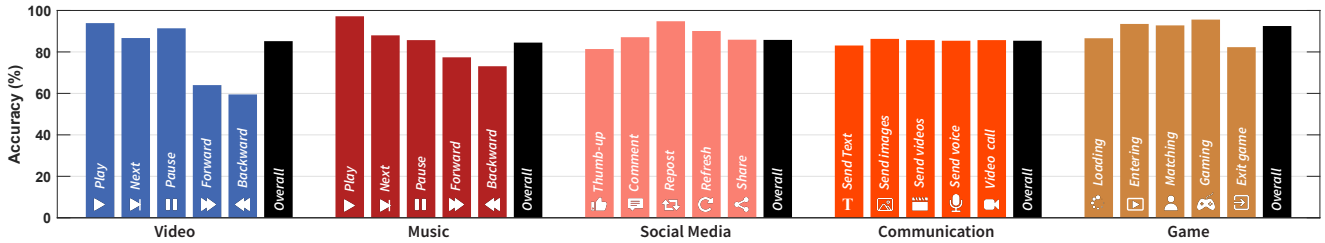
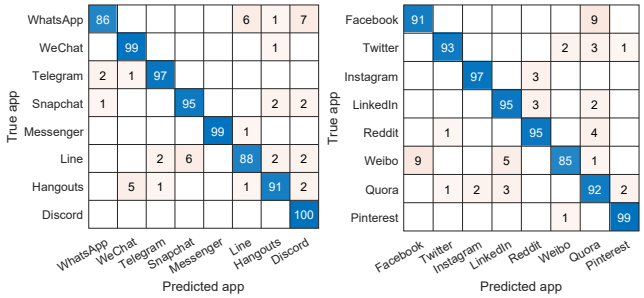
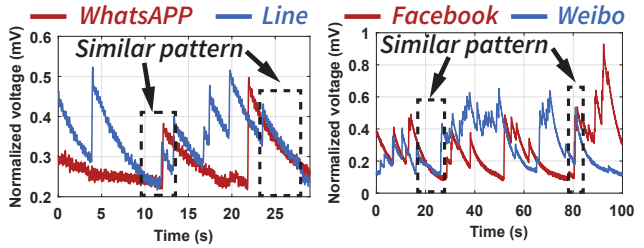


Figure 10: Accuracy (%) in recognizing the app activities. The results are averaged over the apps in each of the five categories.



(a) Communication apps. (b) Social Media apps

Figure 11: Confusion matrix of apps in two categories.



(a) WhatsApp and Line. (b) Facebook and Weibo.

Figure 12: Harvested signals patterns of apps in the same category.

apps. In other words, if N victims are detected, *AppListener* will produce N classification results. Specifically, we randomly generate 40 app combinations (e.g., YouTube and TikTok) from ten popular apps in the two-victim and three-victim scenario (each victim uses one app) to empirically evaluate the accuracy of recognizing different apps. For example, in the two-victim scenario, if they are using YouTube and TikTok, we can successfully recognize these two apps with the precision of 99.6%. The detailed app combination and the associated recognition accuracy are presented in Table 2. Overall, the above experiments show that *AppListener* can achieve an average classification accuracy of 93.3% in the two-victim scenario and 86.5% in the three-victim scenario.

Additionally, comparing the performance in the two-victim and three-victim scenarios, we can observe that the app-level classification accuracy decreases when the number of victims increases. Fundamentally, this is resulted from the limitations of the state-of-the-art signal separation methods, which cannot achieve perfect separation. Correspondingly, each separated signal is still contaminated by other victim's signals, which reduces the performance of *AppListener* in multi-victim scenarios.

Table 2: Evaluation results of multiple victims scenarios. Note "●" and "○" represent app in using or not.

	App Combinations	Acc. (%)	App Combinations	Acc. (%)
Two Victims Scenario	●●○○○○○○○○	99.6	○○●○○○●○○○	96.1
	●○●○○○○○○○	91.1	○○●○○○○○●○	95.8
	●○○●○○○○○○	89.2	○○●○○○○○○●	87.7
	●○○○●○○○○○	93.8	○○●○○○○○○●	90.6
	●○○○○●○○○○	98.9	○○○●●○○○○○	87.9
	●○○○○○●○○○	92.5	○○○●○●○○○○	88.5
	●○○○○○○●○○	93.1	○○○●○○●○○○	87.7
	●○○○○○○○●○	96.9	○○○●○○○●○○	89.5
	●○○○○○○○○●	92.4	○○○●○○○○●○	93.2
	○●●○○○○○○○	92.2	○○○●○○○○○●	92.6
	○●○●○○○○○○	95.2	○○○○●●○○○○	98.5
	○●○○●○○○○○	90.3	○○○○●○●○○○	96.1
	○●○○○●○○○○	97.5	○○○○●○○●○○	97.2
	○●○○○○●○○○	90.0	○○○○●○○○●○	95.6
	○●○○○○○●○○	87.7	○○○○●○○○○●	91.1
	○●○○○○○○●○	92.0	○○○○○●●○○○	92.6
	○●○○○○○○○●	89.0	○○○○○●○●○○	91.9
	○○●●○○○○○○	96.8	○○○○○●○○●○	95.1
	○○●○●○○○○○	95.6	○○○○○●○○○●	95.8
	○○●○○●○○○○	98.8	○○○○○○●●○○	96.5
Three Victims Scenario	●●●○○○○○○○	89.3	●○○○●○○○○●	87.4
	●●○●○○○○○○	87.4	●○○○○●●○○○	87.8
	●●○○●○○○○○	84.8	●○○○●○●○○○	85.6
	●●○○○●○○○○	86.5	●○○○●○○●○○	86.1
	●●○○○○●○○○	86.7	●○○○●○○○●○	84.6
	●●○○○○○●○○	87.9	●○○○●○○○○●	85.4
	●●○○○○○○●○	87.7	●○○○○●●○○○	86.9
	●●○○○○○○○●	84.7	●○○○○●○●○○	87.4
	●○●●○○○○○○	86.4	●○○○○●○○●○	86.3
	●○●○●○○○○○	90.2	●○○○○●○○○●	85.4
	●○●○○●○○○○	88.0	●○○○○○●●○○	89.0
	●○●○○○●○○○	85.3	●○○○○○●○●○	87.2
	●○●○○○○●○○	85.6	●○○○○○●○○●	86.9
	●○●○○○○○●○	87.2	●○○○○○○●●○	84.7
	●○●○○○○○○●	84.5	●○○○○○○●○●	84.2
	●○○●●○○○○○	86.5	●○○○○○○●○●	83.8
	●○○●○●○○○○	88.0	○●●●○○○○○○	86.5
	●○○●○○●○○○	87.5	○●●○●○○○○○	83.7
	●○○●○○○●○○	90.0	○●●○○●○○○○	85.6
	●○○●○○○○●○	86.9	○●●○○○●○○○	86.0

4.3 Impacts of Practical Factors

Impact of distance. As mentioned in §2.1, the distance between the RF energy harvester and the target device plays an important role in the reliability of harvesting. To evaluate its impact, we follow the same procedures as that in the single-victim scenario while placing our custom-built RF energy harvester at different distances to the target Wi-Fi router ranging from 0.3 m and 2.4 m with an interval of 0.3 m. As can be seen in Figure 13, the accuracy rates in recognizing fine-grained mobile app activities of different victims scenarios all decrease as distance increases. In particular, the accuracy

rates drop much faster after 1.5 m, and *the distance within 1.5m has a limited impact on the recognition accuracy.*

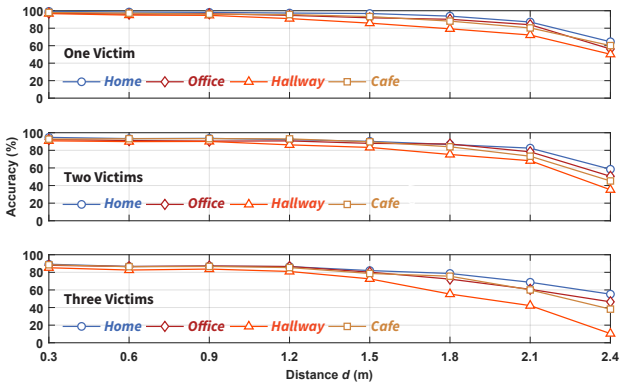


Figure 13: Analysis of the impact of distance.

In addition, though slightly, the accuracy rates of distances within 1.5 m also drop as distance increases. Fortunately, we can observe from Figure 14 that the voltage harvested at different distances exhibit similar patterns. Therefore, as can be seen in Figure 14, we can use normalization to mitigate the impact of distance if the distance is less than 1.5 m.

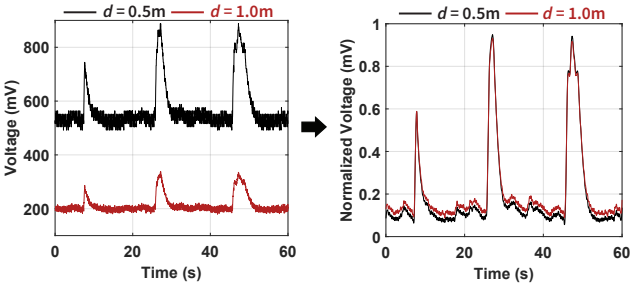


Figure 14: Harvested energy at 0.5 m and 1.0 m. The left figure shows the raw harvested energy while the right figure shows the harvested energy after normalization.

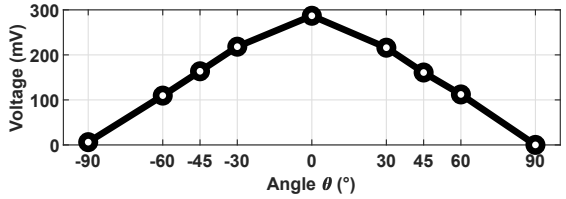


Figure 15: Analysis of the impact of antenna angles θ.

Impact of antenna angles. To evaluate the impact of antenna angle, we place the prototype 1 m away from the Wi-Fi router and change their angle θ from -90° to $+90^\circ$. As the result in Figure 15, the highest voltage is achieved when two antennas are parallel ($\theta = 0$) and decreases with the increment of θ. This result indicates that the attacker can actually find out the direction of the victim's Wi-Fi router by adjusting his own antenna to achieve the maximum harvested voltage. Similar to the impact of distance, although the relationship between angle and harvested voltage is not strictly linear, it can be approximated by a linear function. Therefore, normalization can also be used to reduce the impact of antenna angle.

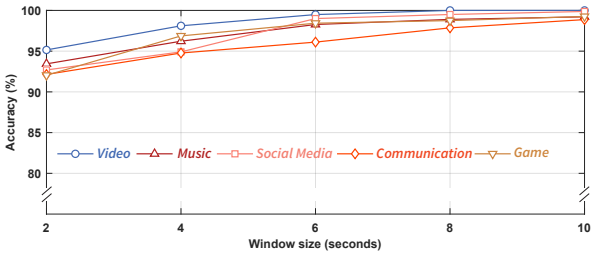


Figure 16: Analysis of the impact of sliding window size.

Impact of sliding window size. As mentioned in §3.3, we use a four-second sliding window to extract features from the harvested energy, which is a widely adopted parameter to extract signal features. In theory, the size of this sliding window affects the quality of the feature extraction because a larger sliding window allows a signal segment contains more information, which is beneficial to the classification accuracy. To understand its impact, we examine the impact of sliding window size on app-level classification accuracy following the settings of the single-victim scenario. The results are shown in Figure 16. The accuracy for apps in all five categories increases when a larger window size is used. Finally, it can achieve over 99% accuracy when a 10 s window is used. However, larger windows lead to a longer response time; therefore, we choose the four-second as our sliding window size following works in the related literature (e.g., [38,44]) to reach a reasonable trade-off between classification accuracy and response time.

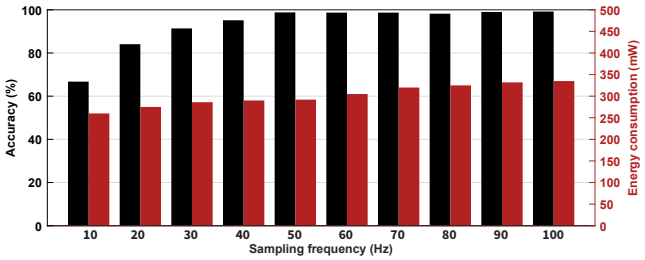


Figure 17: Analysis of the impact of sampling frequency.

Impact of sampling frequency. The sampling frequency of RF energy harvester may impact the quality of the harvested energy because higher frequency can capture more signal samples. However, higher frequency can also consume much more power. To understand the impact of the sampling frequency and reach a reasonable balance between accuracy, frequency, and energy consumption, we design and conduct an experiment by placing our energy harvester with different sampling frequencies, ranging from 10 Hz to 100 Hz with an interval of 10 Hz, at 0.5 m distance to the TP-Link Wi-Fi router. We then connect our iPhone 11 to the router and play all 40 mobile apps in the single-victim scenario setting. In addition, we use a Monsoon Power Monitor to measure the corresponding energy consumption of our energy harvester.

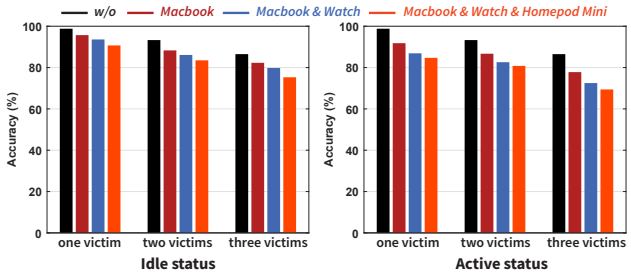


Figure 18: Analysis of the impact of non-target devices.

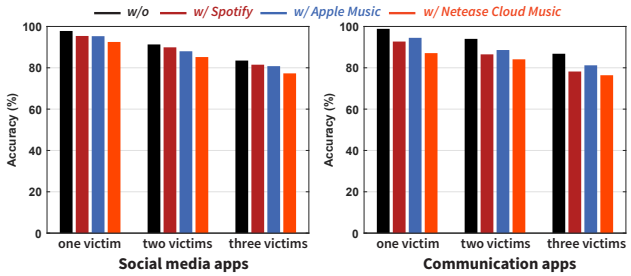


Figure 19: Analysis of the impact of background apps.

Figure 17 presents the app recognition accuracy at different sampling frequencies alongside the corresponding energy consumption. As can be seen, after 50 Hz, the accuracy reaches 98.8% and increases slightly to 99.2% at the frequency of 100 Hz; however, the energy consumption increases from 292 mW at 50 Hz to 335 mW at 100 Hz. As such, we believe using 50 Hz as the default sampling frequency can achieve a good balance between accuracy and energy consumption.

Impact of connected non-target devices. *AppListener* currently focuses on monitoring app activities from smartphones and tablets, other devices connecting to the same Wi-Fi router may result in interfering noises that may negatively affect the accuracy. To understand the impact of connected non-target devices, we conducted two experiments by connecting three non-target devices to our TP-Link router, including a Macbook Pro, an Apple Watch, and an Apple Homepod Mini, running random system services (idle status) and specific traffic-intensive activities (active status) at the background. Then, we used iPhone 11 to run 40 apps to evaluate the accuracy of *AppListener* in terms of app recognition in the single-victim scenario and use two additional smartphones (i.e., Samsung S10 and Nexus 6P) in the multi-victim scenario. Specifically, in the active status, we use Macbook to browse websites, Apple Watch to record exercises, and Homepod Mini to continuously play music. Figure 18 presents the evaluation results. In all scenarios, the accuracy could achieve 98.8% and drop as the number of connected non-target devices and victims increases. Specifically, in the idle status, the accuracy ranges from 98.8% when there is no non-target devices in the single-victim scenario to 75.3% if there are three non-target devices in the multi-victim scenario. In the active status, the accuracy ranges from 98.8% to 70.4% in the same setting as that in the idle status. The accuracy in the active status is lower than the idle status, 4.9%, in the three-victim scenario because traffic-intensive activities in non-target devices introduce extra noises to the harvested RF energy. As such, *it could bring non-significant impacts if there are connected non-target devices and they are running actively.*

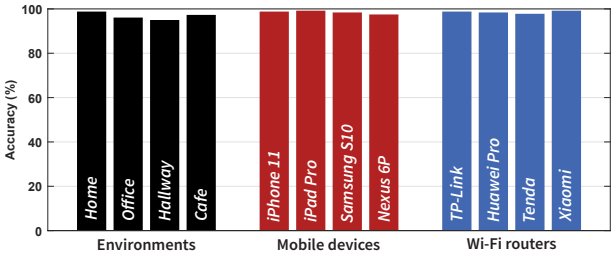


Figure 20: Analysis of the impact of environment and hardware.

Impact of background apps. To explore the impact of background apps, we evaluate the performance of *AppListener* in recognizing social media and communication apps when the target device is running different music apps (e.g., Spotify, Apple Music, and Netease Cloud Music) in the background. This setting is close to real-world scenarios because most video or game apps disable the background apps automatically (e.g., pausing the music). Figure 19 shows that when playing different background music apps, the accuracy ranges from 97.8% and 98.9% in the single-victim scenario to 77.3% and 76.4% in the multi-victim scenario in recognizing social media and communication apps. As such, *though non-significantly, background apps could impact the performance of AppListener.*

Impact of environment. Figure 20 shows the accuracy of *AppListener* in recognizing all 40 apps when it is deployed in the office, home, hallway, and cafe environment, respectively. As can be seen, it achieves a relatively lower accuracy in the hallway (95.0%), but reaches an accuracy of 96.1% in the office, 97.3% in the cafe, and 98.8% (the highest one) in the home environment. As such, this 3.8% difference indicates that *the impact of different environments is limited*.

Impact of hardware. To study the impact of hardware on *AppListener*'s performance, we use four different smart devices on which victims play apps (i.e., iPhone 11, iPad Pro, Samsung S10, and Nexus 6P) and four different Wi-Fi routers of four mainstream manufacturers, including TP-Link, Huawei Pro, Tenda, and Xiaomi. In particular, these routers differ in their wireless network adapters and the number of antennas on board. To evaluate the recognition accuracy of different hardware, we consider device-dependent models in this experiment, where data from the same device are used for training and testing. Specifically, we connect four smart devices to the TP-Link Wi-Fi router to collect data in evaluating the impact of these smart devices, and we use the iPhone 11 to connect to four Wi-Fi routers to evaluate the impact of different routers. Similarly, the data are collected by playing all 40 apps, and the results are also presented in Figure 20. We can see that *AppListener* achieves over 98.5% accuracy in single-victim app recognition regardless of the type of hardware used, indicating *the impact of different routers is limited*.

4.4 Generalization Analysis

Cross environments. We first evaluate the cross-site eavesdropping ability of *AppListener*, i.e., whether a model trained from one environment can be directly re-used in other environments. To this end, we connect an iPhone 11 to a TP-Link router playing 40 apps in four different environments, and split the collected into training set and testing set with the ratio of 80/20. Then, we train four classification models based on the training data of each of these environments. Next, we evaluate the accuracy of app recognition using each of these models on the testing data of all the four environments.

From the results in Table 3, we can observe that the classification accuracy drops at most by 4.5%, 3.8%, 3.5%, and 2.3% if applying the model trained in the home, office, hallway, and cafe environment, respectively, to any other scenarios. *The results demonstrate the strong generalization ability of AppListener in different environments*.

Cross mobile devices. To evaluate this generalization ability, we connect four mobile devices to the TP-Link router in the home environment, in the single-victim scenario, playing all 40 apps. Similarly, we split the data collected from these four smart mobile devices into training and testing set with the ratio of 80/20. In Table 3, we find that the accuracy of app recognition drops by 10.8%, 9%, 9.8%, and 9.5% if applying the model trained in the iPhone 11, iPad Pro, Samsung S10, and Nexus 6P, respectively, to other devices. We believe this level of accuracy decrease is acceptable and *it is practical for AppListener to conduct cross mobile device attacks.*

Cross Wi-Fi routers. In this evaluation, we connect iPhone 11 to four different Wi-Fi routers in the home environment playing all 40 apps, also in the single-victim scenario. We follow the 80/20 ratio to split the training and testing set for data collected from the four Wi-Fi routers and show the results in Table 3. Similar to the above two evaluations, the accuracy of app recognition slightly drops if applying the model trained on the data collected from one router to other routers. In this evaluation, the accuracy drops at most 4.7%, which presents *the competitive generalization ability of AppListener in cross Wi-Fi router scenarios*.

5 Discussion

Attack novelty and advantages. This paper reports a novel side-channel attack that leverages the RF energy harvesting technique to accurately recognize specific mobile apps and pinpoint fine-grained in-app activities. In the literature, similar works are often achieved by analyzing network traffics for app fingerprinting. Table 4 presents an in-depth comparison between our *AppListener* and the state-of-the-art works that use traffic analysis for app fingerprinting from the perspective of side channel, package inspection requirement, traffic encryption, in-app activity recognition, and supporting of multi-victim. In particular, most of these traffic-based works (except ActiveTracker [20]) require package inspection (e.g., seeking for the IP address or destination address), while *AppListener* eliminates this requirement by leveraging the energy information of the wireless traffic instead of the content of the wireless traffic. In addition, *AppListener* also has no requirement on the encryption of the traffic, which is the advantage claimed by many traffic-based works (except DECANTeR [16]). Moreover, *AppListener* enables fine-grained in-app activity recognition, which is not supported by all traffic-based works (except NetScope [17], FlowPrint [21], ActiveTracker [20] and FOAP [22]) and using relative fewer features in recognition in multi-victim scenarios that is only supported by FOAP [22], which is a concurrent work of *AppListener*. As a novel side-channel attack, *AppListener shares most advances of traffic-based works while supporting multi-victim scenarios with less model training efforts while achieving competitive performance.*

Limitations. We have implemented a prototype of *AppListener* to demonstrate the feasibility of our newly reported side channel. While our initial results are promising, there still exist several limitations in the current work. First, in multi-victim scenario, *AppListener* uses the combined energy harvesting signals to predict the app activities. However, due to the limitations of the state-of-the-art source separation algorithms, *AppListener* cannot understand which app is running on which smartphone if these smartphones are running different apps. Similarly, it cannot detect the exact number of smartphones if all victims run the same app.

Second, while *AppListener* has a great potential to be battery-free, which could improve its ability for long-time standing and become less noticeable, our current prototype still needs a battery to power its operations. Our current prototype takes 292 mW on average to capture and transmit RF energy through BLE to our remote data analysis device. In terms of energy harvesting, limited by the efficiency of the off-the-shelf RF-DC converter, our prototype can harvest power around 270 mW. In other words, two 3 V and 550 mAh button cell batteries can support the device to operate by approximately 120 hours. Therefore, one of our future works is to improve our prototype to be battery-free. Additionally, this study assumes the transmission power is fixed in Wi-Fi routers. If the Wi-Fi access point adopts dynamic transmit power, the harvested energy will also be changed dynamically according to Equation 1. As such, the harvested energy cannot reveal the real traffic patterns of the smartphone activities and our side-channel attack will not be launched successfully.

Finally, our current prototype performs the best if placed within 1.5 m to the target device (e.g., Wi-Fi router). Even if the RF energy could be captured in a much longer distance (e.g., 30 meters [3]), longer distance reduces the quality of the signal making it insufficient to launch our reported new side-channel attack. While we believe the distance of 1.5 m is sufficient to demonstrate the feasibility of this new side-channel attack with this configuration, such a distance could be further improved in the future work. For instance, more RF-DC converters can be added to improve the total amount of harvested

Table 3: Results of cross environments, cross mobile devices, and cross Wi-Fi routers.

Accuracy (%)		Test environments				Test mobile devices					Test Wi-Fi routers			
		Home	Office	Hallway	Cafe		iPhone 11	iPad Pro	Samsung	Nexus 6P		TP-Link	Huawei	Tenda	Xiaomi
Training model	Home	98.8	96.5	94.3	98.2	iPhone 11	98.8	96.5	90.2	88.0	TP-Link	98.8	95.4	97.7	98.0
	Office	95.3	96.1	92.3	95.8	iPad Pro	94.3	99.2	91.6	90.2	Huawei	95.8	98.4	95.4	96.2
	Hallway	94.3	91.5	95.0	92.1	Samsung	89.3	88.6	98.4	93.1	Tenda	94.3	96.5	97.8	95.5
	Cafe	96.6	96.1	95.0	97.3	Nexus 6P	87.7	88.0	92.9	97.5	Xiaomi	94.5	95.2	97.3	99.2

Table 4: Comparison with prior works. Note "●" and "○" represent "Yes" and "No". W/O IP/Des.: Without IP/Destination address.

Works	Network Traffic	RF Energy	W/O IP/Des.	Encrypted	In-app Activity	# of Features	Multi-victim
DECANTeR [16]	●	○	○	○	○	6	○
AppScanner [9]	●	○	○	●	○	54	○
NetScope [17]	●	○	○	●	●	N/A	○
MIMETIC [18]	●	○	○	●	○	N/A	○
Liu et al. [19]	●	○	○	●	●	30	○
ActiveTracker [20]	●	○	●	●	●	N/A	○
FlowPrint [21]	●	○	○	●	●	110	○
FOAP [22]	●	○	●	●	●	123	●
AppListener (Ours)	○	●	●	●	●	31	●

energy, and thus improve the attacking distance. For concept proof, we prototype a new RF energy harvester with two original harvesters by parallel connecting their RF-DC circuits to enhance the harvesting efficiency. Then, we follow the same procedure to collect data of 40 apps at different attacking distances from 0.3 m to 3.6 m and evaluate the accuracy of the app recognition in the single-victim scenario when using the upgraded harvester. As shown in Figure 21, comparing to our proposed *Burger Model* which uses one RF-DC converter, the attacking distance for our new two-converter harvester is extended from 2 m to 3.2 m while maintaining over 90% accuracy in pinpointing different mobile apps. On the contrary, using more RF-DC converters in implementing the harvester also increases hardware cost and battery consumption.

Extending attacks. *AppListener* is proposed as a general attack framework and its ability could go beyond the presented attacks in this paper. For example, we believe it could be extended to device fingerprinting attacks. To demonstrate the feasibility, we have conducted a preliminary study by analyzing the harvested signals of different smartphones when their users are playing the same app. In particular, Figure 22a shows the signal traces of an iPhone 11 and Samsung S10 when their users are switching the videos in TikTok. Although the overall signal patterns of different smartphones are similar, there still exist minor differences, due to the difference in hardware design and manufacturing. The results in Figure 22b show that we can achieve over 99% accuracy in identifying four different mobile devices (i.e., iPhone 11, iPad Pro, Samsung S10, and Nexus 6P), which makes the device fingerprinting attack feasible and practical. Moreover, *AppListener* can be extended to support monitoring more than three victim devices presented in this paper. Due to the nature of this new side channel, we can place more energy harvesters and tune the algorithm to support more victim devices. Considering the small size and the potential self-powering of custom-built harvester as well as the unnoticeable outlook of compromised commodity IoT device (discussed in the following paragraphs), it could be practical to place or hide more harvesters nearby a commodity Wi-Fi router.

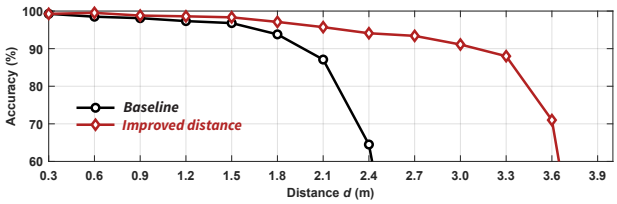
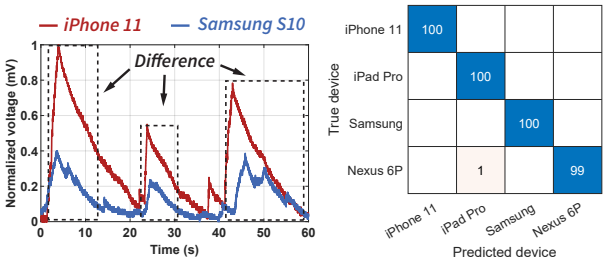


Figure 21: Improving attack distance with two RF-DC converters harvester comparing to one-converter harvester (*Burger Model*).



(a) (b)

Figure 22: (a) Harvested signals of iPhone 11 and Samsung S10 of switching videos in TikTok. (b) Results of device fingerprinting.

Countermeasures. Since *AppListener* leverages harvested signals from wireless traffic transmission between smartphones and routers to eavesdrop mobile app activities, there are two basic countermeasures to defend against our reported novel side-channel attacks: traffic obfuscation and dynamic transmission power adaptation. Specifically, a traffic obfuscation approach, such as transmitting redundant packets from the Wi-Fi router, is effective because these redundant packets can interfere the voltage signals harvested by *AppListener*, leading to performance degradation. In respect of the dynamic transmission power, it can mitigate this new attack because *AppListener* assumes a fixed transmitting power on the Wi-Fi router side. As is shown in Equation 1, the received power is affected by the transmission power. If the Wi-Fi router adopts dynamic transmit power, the harvested energy will changed dynamically and cannot reflect the real traffic loads of the smartphone activities. To investigate the prediction accuracy under dynamic transmit power control, we enable this mode in the Wi-Fi router (Huawei Pro) and repeat the same experiments in the single and multi-victim scenarios. Results show the accuracy dropped by 29.7%, 38.6% and 45.3% in one-, two-, and three-victim scenarios. Therefore, if the Wi-Fi

Table 5: Results of compromising commodity IoT devices. Note "●" and "○" represent "Yes" and "No".

Commodity Product	Energy Harvester	Antenna	Gain	BLE	Acc. (%) of Multi-Victim Scenarios			Max. Distance (m) (One Acc. > 90%)
					One	Two	Three	
ZF Energy Harvesting BLE Push-button [46]	●	●	N/A	●	93.6	89.5	82.2	∼ 1.05
Xiaomi Mi Bedside Smart Lamp [47]	○	●	N/A	●	90.9	83.0	77.1	∼ 0.60
GHome Smart LED Bulb [48]	○	●	N/A	○	91.8	86.2	80.7	∼ 1.50
Tuya Wi-Fi Temperature and Humidity Sensor [49]	○	●	1.3 dBi	●	86.9	83.1	78.1	∼ 0.45
Tuya Smart Plug (With Metering) [50]	○	●	1.0 dBi	○	89.4	83.6	76.9	∼ 0.45
Zinguo Wi-Fi Smart Switch [51]	○	●	3.0 dBi	○	91.8	85.8	81.6	∼ 0.90

router applies a dynamic transmission power deliberately, i.e., bursting transmission in low-power mode while transmitting small packets in high-power mode, the radio patterns of each app will be changed accordingly. As such, *AppListener* will be deceived by the modified radio patterns. However, the dynamic transmit power scheme cannot guarantee the link quality because it is used to improve spectral efficiency and reduce interference by transmitting with the minimum power provided that the link is not overloaded.

Potential of through-wall attack. In the experiments above, we assume there exists line-of-sight between the Wi-Fi router and the RF energy harvester. However, in practice, there might be solid obstacles between the target Wi-Fi router and harvester. We now evaluate the classification accuracy when the target Wi-Fi router and energy harvester are blocked by different materials. Specifically, we consider partition board, wooden door, and thin/thick concrete walls. *AppListener* works if the Wi-Fi router and energy harvester are blocked by a partition board, a wooden door, and a thin concrete wall (8.0 cm), but fails to work when they are blocked by a thick concrete wall (27.4 cm) due to its higher attenuation [52]. The full results are presented in Table 8 in Appendix.

Integrating with commodity IoT devices. To demonstrate the feasibility of integrating our proposed side-channel attack framework with commodity IoT devices, we have purchased six IoT devices and manually modified them to be potential RF energy harvesters. These commodity IoT devices include push-button, smart lamp, bulb, plug, and switch, and they come with different hardware components. Table 5 shows their hardware components that are relevant to this study, including the energy harvester, the antenna, the antenna gain, and the BLE module. In particular, only one device, i.e., ZF Energy Harvesting BLE Push-button [46], comes with the harvesting module by default. For the other devices, we unpack the exterior, embed the designed RF energy harvester, and connect it to the antenna of the original devices as shown in Figure 25 in Appendix. It is worth noting that after installing the RF energy harvesters, these devices can be assembled as its original outlook because our designed RF energy harvesters are small.

To evaluate the effectiveness of integrating *AppListener* with these commodity IoT devices, we conduct an experiment using these modified devices in recognizing 40 mobile apps in both single-victim and multi-victim scenarios. As the results in Table 5, they perform differently due to different materials and hardware configurations (especially antennas). Overall, they can achieve over 75% accuracy in app recognition when placed at approximately 0.5 m from the Wi-Fi router in the worst case of the three-victim scenario. The results indicate *the practicality of launching this novel side-channel attack by integrating AppListener with commodity IoT devices*.

6 Related Work

Network Traffic-based Attacks. A majority of smartphone activity eavesdropping attacks are based on capturing and analyzing network traffics [53–56]. Taylor *et al.* introduced AppScanner [9], which used network data, such as IP addresses, DNS queries, and packet payloads, to identify mobile apps. Saltaformaggio *et al.* proposed NetScope [17] to recognize apps and fine-grained activities based on inspecting IP packet headers and metadata. Based on these two works, Aceto *et al.* [18] further proposed combining-decision and DNN methods to improve the performance of traffic classification on app fingerprinting. ActiveTracker [20] reveals fine-grained app activities without inspect the content of packets, FlowPrint [21] proposed a semi-supervised app fingerprinting method by capturing temporal correlations among destination-related features of an encrypted network traffic, and FOAP [22] presents an open-world app fingerprinting method that can deal with unsegmented encrypted traffic and handle app multiplexing. Different from these works, *AppListener* exploits a new side channel to recognize mobile apps and pinpoint fine-grained in-app activities with less assumptions and achieves competitive attacking performance.

Sensor-based Attacks. A number of studies have revealed the possibility of eavesdropping users' private information by exploiting the leaked information in various sensors. For example, AccelWord [11], AccelEve [13] and Spearphone [12] demonstrated the feasibility of performing side-channel attacks by using the motion sensors (i.e., accelerometer and gyroscope). PatternListener [14] and KeyListener [15] exploited the acoustic sensors (i.e., microphone) to crack the smartphone's lock pattern and eavesdrop the keystrokes on the touch screen. Unlike these works, *AppListener* leverages the harvested RF energy to uncover user privacy.

Power Analysis-based Attacks. Another line of attacks aims to eavesdrop users' private information by analyzing the battery profile of mobile devices during the charging process or normal usage. For instance, Yang *et al.* [26] demonstrated that a USB charging port leaked information about the webpages being loaded on the touchscreen. Cronin *et al.* [10] presented Charger-Surfing to leak touchscreen information from the

USB cable charging smartphone. POWERFUL [8] presented an attack to identify mobile apps via malware injection into the target smartphone to obtain battery consumption profile. PowerSpy [25] revealed that the power consumption data of a smartphone can leak the location privacy. Comparing to these works, *AppListener* is non-intrusive, stealthy, practical, and can work in multi-victim scenarios.

7 Conclusion

In this paper, we present a novel side-channel attack on mobile device activities based on RF energy harvesting. To validate the feasibility, we propose *AppListener*, the first automated system that leverages RF energy harvesting as a side-channel to eavesdrop a victim's smartphone app activities. Evaluation results show that *AppListener* achieves 98.8% prediction accuracy for a single victim, 93.3% accuracy for two victims, and 86.5% accuracy for three victims. Also, *AppListener* shows 86.7% accuracy in identifying specific app activities. We also evaluate the generalization ability and resilience of *AppListener* and the results show that *AppListener* maintains a competitive eavesdropping accuracy across different practical impact factors. We hope our findings can raise public awareness and spur research on detecting forthcoming attacks and new defense methods.

Acknowledgments

We sincerely thank our shepherd and anonymous reviewers for their constructive comments. This research was substantially supported by NFSC (Project 62101471) and was partially supported by the Shenzhen Research Institute, City University of Hong Kong, the Research Grants Council of the Hong Kong SAR, China (ECS Project CityU 21201420 and GRF Project CityU 11201422), CityU APRC grant 9610563, CityU SRG-Fd grant 7005853, Shenzhen Science and Technology Funding Fundamental Research Program (Project No. 2021Szvup126), NSF of Shandong Province (Project No. ZR2021LZH010), and a grant from Chow Sang Sang Group Research Fund sponsored by Chow Sang Sang Holdings International Limited (Project No. 9229062). Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily of supported organizations.

References

[1] Ugur Olgun, Chi-Chih Chen, and John L Volakis. Efficient ambient wifi energy harvesting technology and its applications. In *Proceedings of the IEEE International Symposium on Antennas and Propagation*, 2012.

[2] Chomora Mikeka and Hiroyuki Arai. Design of a cellular energy-harvesting radio. In *Proceedings of the European Wireless Technology Conference*, 2009.

[3] Xiaoqing Tang, Guihui Xie, and Yongqiang Cui. Self-sustainable long range backscattering communication using rf energy harvesting. *IEEE IoTJ*, 2021.

[4] Vamsi Talla, Bryce Kellogg, Benjamin Ransford, Saman Naderiparizi, Shyamnath Gollakota, and Joshua R Smith. Powering the next billion devices with wi-fi. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*, 2015.

[5] Gao-Ching Lin, Min-Wei Lee, and Yu-Cheng Hsu. An ac-dc rectifier for rf energy harvesting system. In *Proceedings of the Asia Pacific Microwave Conference*, pages 1052–1054. IEEE, 2012.

[6] Chris Horn. *Cutting the cord: when will Tesla's 100-year-old vision become reality?* THE IRISH TIMES, 2019. https://gbksoft.com/blog/mobile-app-market-analysis/.

[7] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. In *Proceedings of the ACM CCS*, 2016.

[8] Yimin Chen, Xiaocong Jin, Jingchao Sun, Rui Zhang, and Yanchao Zhang. Powerful: Mobile app fingerprinting via power analysis. In *Proceedings of the IEEE INFOCOM*, 2017.

[9] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proceedings of the IEEE EuroS&P*, 2016.

[10] Patrick Cronin, Xing Gao, Chengmo Yang, and Haining Wang. Charger-surfing: Exploiting a power line side-channel for smartphone information leakage. In *Proceedings of the USENIX Security Symposium*, 2021.

[11] Li Zhang, Parth H Pathak, Muchen Wu, Yixin Zhao, and Prasant Mohapatra. Accelword: Energy efficient hotword detection through accelerometer. In *Proceedings of ACM MobiSys*, 2015.

[12] S Abhishek Anand, Chen Wang, Jian Liu, Nitesh Saxena, and Yingying Chen. Spearphone: A speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers. *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2019.

[13] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren. Learning-based practical smartphone eavesdropping with built-in accelerometer. In *Proceedings of NDSS*, 2020.

[14] Man Zhou, Qian Wang, Jingxiao Yang, Qi Li, Feng Xiao, Zhibo Wang, and Xiaofeng Chen. Patternlistener: Cracking android pattern lock using acoustic signals. In *Proceedings of the ACM CCS*, 2018.

[15] Jiadi Yu, Li Lu, Yingying Chen, Yanmin Zhu, and Linghe Kong. An indirect eavesdropping attack of keystrokes on touch screen through acoustic sensing. *IEEE Transactions on Mobile Computing*, 2019.

[16] Riccardo Bortolameotti, Thijs van Ede, Marco Caselli, Maarten H Everts, Pieter Hartel, Rick Hofstede, Willem Jonker, and Andreas Peter. Decanter: Detection of anomalous outbound http traffic by passive application fingerprinting. In *Proceedings of ACSAC*, 2017.

[17] Brendan Saltaformaggio, Hongjun Choi, Kristen Johnson, Yonghwi Kwon, Qi Zhang, Xiangyu Zhang, Dongyan Xu, and John Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2016.

[18] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapè. Mimetic: Mobile encrypted traffic classification using multimodal deep learning. *Computer Networks*, 165:106944, 2019.

[19] Junming Liu, Yanjie Fu, Jingci Ming, Yong Ren, Leilei Sun, and Hui Xiong. Effective and real-time in-app activity analysis in encrypted internet traffic streams. In *Proceedings of the ACM KDD*, 2017.

[20] Ding Li, Wenzhong Li, Xiaoliang Wang, Cam-Tu Nguyen, and Sanglu Lu. Activetracker: Uncovering the trajectory of app activities over encrypted internet traffic streams. In *Proceedings of the IEEE SECON*, 2019.

[21] Thijs van Ede, Riccardo Bortolameotti, Andrea Continella, Jingjing Ren, Daniel J Dubois, Martina Lindorfer, David Choffnes, Maarten van Steen, and Andreas Peter. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Proceedings of NDSS*, 2020.

[22] Jianfeng Li, Hao Zhou, Shuohan Wu, Xiapu Luo, Ting Wang, Xian Zhan, and Xiaobo Ma. Foap: Fine-grained open-world android app fingerprinting. In *Proceedings of USENIX Security Symposium*, 2022.

[23] Yi Han, Sriharsha Etigowni, Hua Liu, Saman Zonouz, and Athina Petropulu. Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In *Proceedings of the ACM CCS*, 2017.

[24] Nader Sehatbakhsh, Baki Berkay Yilmaz, Alenka Zajic, and Milos Prvulovic. A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit. In *Proceedings of the IEEE HPCA*, 2020.

[25] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *Proceedings of the 24th USENIX Security Symposium*, pages 785–800, 2015.

[26] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, and Kiran S Balagani. On inferring browsing activity on smartphones via usb power analysis side-channel. *IEEE TIFS*, 2016.

[27] Hamid Jabbar, Young S Song, and Taikyeong Ted Jeong. Rf energy harvesting system and circuits for charging of mobile devices. *IEEE Transactions on Consumer Electronics*, 56(1):247–253, 2010.

[28] Sangkil Kim, Rushi Vyas, Jo Bito, Kyriaki Niotaki, Ana Collado, Apostolos Georgiadis, and Manos M Tentzeris. Ambient rf energy-harvesting technologies for self-sustainable standalone wireless sensor platforms. *Proceedings of the IEEE*, 102(11):1649–1666, 2014.

[29] Manish Wadhwa, Min Song, Vinay Rali, and Sachin Shetty. The impact of antenna orientation on wireless sensor network performance. In *Proceedings of the IEEE International Conference on Computer Science and Information Technology*, 2009.

[30] Thomas Stockhammer. Dynamic adaptive streaming over HTTP– standards and design principles. In *Proceedings of the ACM Conference on Multimedia Systems*, pages 133–144, 2011.

[31] Iraj Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE Multimedia*, 18(4):62–67, 2011.

[32] Craig Gutterman, Katherine Guo, Sarthak Arora, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman. Requet: Real-time QoE detection for encrypted YouTube traffic. In *Proceedings of the ACM Multimedia Systems Conference*, pages 48–59, 2019.

[33] Chi Zhang, Wonsun Ahn, Youtao Zhang, and Bruce R Childers. Live code update for iot devices in energy harvesting environments. In *Proceedings of the Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pages 1–6. IEEE, 2016.

[34] Nasif Bin Shafi, Kashif Ali, and Hossam S Hassanein. No-reboot and zero-flash over-the-air programming for wireless sensor networks. In *Proceedings of the IEEE SECON*, pages 371–379. IEEE, 2012.

[35] Rajesh Krishna Panta and Saurabh Bagchi. Hermes: Fast and energy efficient incremental code updates for wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, pages 639–647. IEEE, 2009.

[36] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[37] Kun-Chou Lee, Jhih-Sian Ou, and Ming-Chung Fang. Application of svd noise-reduction technique to pca based radar target recognition. *Progress In Electromagnetics Research*, 81:447–459, 2008.

[38] Weitao Xu, Guohao Lan, Qi Lin, Sara Khalifa, Mahbub Hassan, Neil Bergmann, and Wen Hu. Keh-gait: Using kinetic energy harvesting for gait-based user authentication systems. *IEEE TMC*, 2018.

[39] Xianchuan Yu, Dan Hu, and Jindong Xu. *Blind source separation: theory and applications*. John Wiley & Sons, 2013.

[40] Bin Gao. *Single channel blind source separation*. PhD thesis, Newcastle University, 2011.

[41] Dominic Langlois, Sylvain Chartier, and Dominique Gosselin. An introduction to independent component analysis: Infomax and fastica algorithms. *Tutorials in Quantitative Methods for Psychology*, 2010.

[42] Jianwen Luo, Kui Ying, and Jing Bai. Savitzky–golay smoothing and differentiation filter for even number data. *Signal processing*, 85(7):1429–1434, 2005.

[43] Deepshikha Acharya, Asha Rani, Shivangi Agarwal, and Vijander Singh. Application of adaptive savitzky–golay filter for eeg signal processing. *Perspectives in Science*, 8:677–679, 2016.

[44] Sara Khalifa, Guohao Lan, Mahbub Hassan, Aruna Seneviratne, and Sajal K Das. Harke: Human activity recognition from kinetic energy harvesting data in wearable devices. *IEEE TMC*, 2017.

[45] Ke Yan and David Zhang. Feature selection and analysis on correlated gas sensor data with recursive feature elimination. *Sensors and Actuators B: Chemical*, 212:353–363, 2015.

[46] ZF Group. Energy harvesting wireless switch. https://switches-sensors.zf.com/wp-content/uploads/2020/03/EN_Energy_Harvesting_Catalogue-1.pdf, pages 14-15.

[47] Xiaomi Global. Mi bedside lamp 2 specs. https://www.mi.com/global/mi-bedside-lamp-2/specs/.

[48] GHome Smart. Smart light bulbs, dimmable led light bulbs works with alexa and google home. https://www.amazon.com/google-home-smart-bulbs/s?k=google+home+smart+bulbs.

[49] Tuya Smart. Wi-fi temperature and humidity sensor. https://expo.tuya.com/product/596383.

[50] Tuya Smart. Standard 10a wifi smart plug-with metering version. https://expo.tuya.com/product/901091.

[51] ZINGUO Smart. Zinguo smart switch wifi remote control touch switch. https://www.aliexpress.com/item/32921129013.html.

[52] Weitao Xu, Jun Young Kim, Walter Huang, Salil S Kanhere, Sanjay K Jha, and Wen Hu. Measurement, characterization, and modeling of lora technology in multifloor buildings. *IEEE Internet of Things Journal*, 2019.

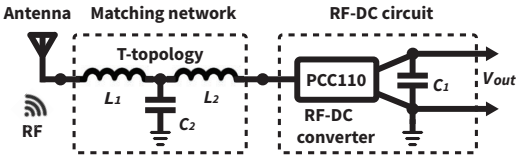
[53] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *Proceedings of the 26th USENIX Security Symposium*, pages 1357–1374, 2017.

[54] Anika Schwind, Florian Wamser, Thomas Gensler, Phuoc Tran-Gia, Michael Seufert, and Pedro Casas. Streaming characteristics of spotify sessions. In *Proceedings of the QoMEX*, 2018.

[55] Xin Su, Dafang Zhang, Wenjia Li, and Xiaofei Wang. Androgenerator: An automated and configurable android app network traffic generation system. *Security and Communication Networks*, 8(18):4273–4288, 2015.

[56] Zhen Liu, Ruoyu Wang, Nathalie Japkowicz, Yongming Cai, Deyu Tang, and Xianfa Cai. Mobile app traffic flow feature extraction and selection for improving classification robustness. *Journal of Network and Computer Applications*, 125:190–208, 2019.

Appendix



(a) Circuit diagram.

(b) Outlook.

Figure 23: Implementation of custom-built RF energy harvester.

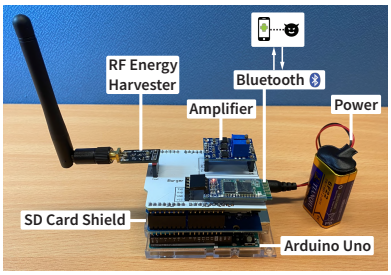
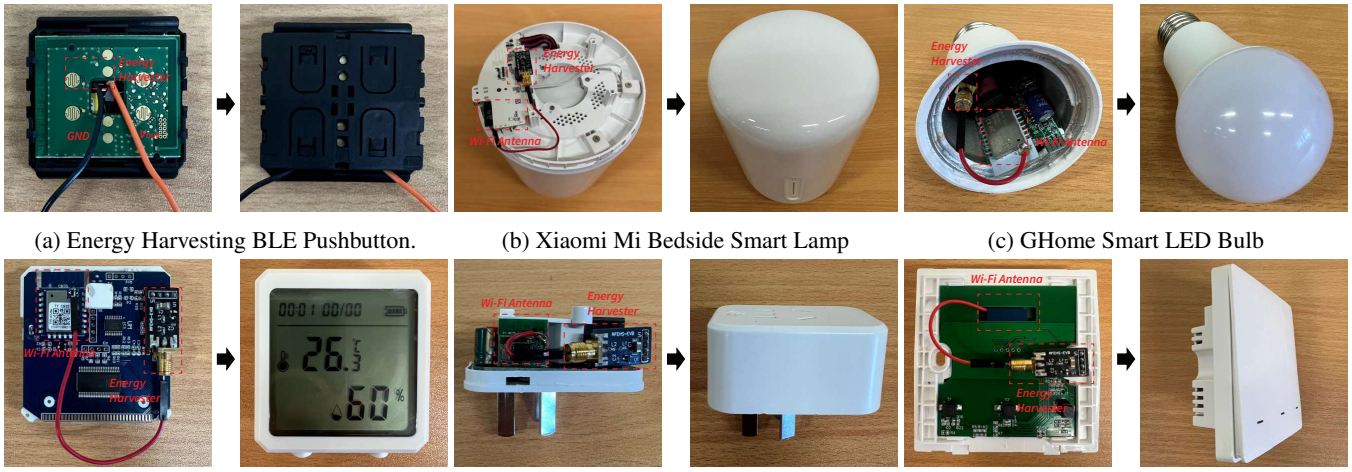


Figure 24: The *Burger Model*.



(a) Energy Harvesting BLE Pushbutton. (b) Xiaomi Mi Bedside Smart Lamp (c) GHome Smart LED Bulb

(d) Wi-Fi Temperature and Humidity Sensor (e) Tuya Smart Plug (With Metering) (f) Zinguo Wi-Fi Smart Switch

Figure 25: Integrating *AppListener* into commodity IoT device with energy harvesters and Wi-Fi antennas as is shown in Table 5

Table 6: Original feature set (Time and Frequency domains).

	Time-domain features (abbrv.)	**Description**	**Selected (✓/✗)**
T-domain basics	Mean (mean)	The average value of harvested voltage samples.	✓
	Standard Deviation (std)	Measures the variation of harvested voltage samples.	✓
	Maximum (max)	The maximum value of harvested voltage samples.	✓
	Minimum (min)	The minimum value of harvested voltage samples.	✓
	Range (range)	The difference of maximum and minimum voltages.	✓
	Absolute Mean (absMean)	The average of absolute harvested voltage values.	✓
	Coefficient of Variation (CV)	The ratio of Standard Deviation and Mean, $CV = \frac{std}{mean}$	✓
	Root Mean Square (RMS)	Measures the effective energy of harvested voltages	✓
	Mean Absolute Deviation (MAD)	Measures the asymmetry and peakedness of harvested voltage values distribution.	✓
	Skewness (skew)		✓
	Kurtosis (kurt)		✓
Quartile features	1st Quartile (Q1)	The first quartile, median, third quartile. They measure the overall distribution of the harvested voltage signals.	✓
	Median (med)		✓
	3rd Quartile (Q3)		✓
	Inter Quartile Range (IQR)	The difference of Q3 and Q1. It also indicates the dispersion.	✓
Crossing rates	Mean Cross Rate (MCR)	The number of voltage values that cross the mean, Q1, median and Q3 values. These features indicate the variation of harvested voltage signal.	✓
	Q1 Cross Rate (Q1CR)		✓
	Median Cross Rate (MedCR)		✓
	Q3 Cross Rate (Q3CR)		✓
Impulsive metrics	Shape Factor (SF)	Impulsive metrics are features that are relevant to the peaks of signals. $SF = \frac{RMS}{absMean}$, $IF = \frac{max}{absMean}$, $CF = \frac{max}{RMS}$	✓
	Impulsive Factor (IF)		✓
	Crest Factor (CF)		✓
Peak features	Distance between Peaks (davg, dmax, dmin)	Measures the average, maximum, minimum distance and height difference between two neighbor peaks.	✗
	Difference between Peaks (dfavg, dfmax, dfmin)		✗
	Frequency-domain features (abbrv.)	**Description**	**Selected (✓/✗)**
F-domain basics	Frequency-domain Mean (FDMean)	The average value of the magnitude of FFT coefficient.	✓
	Dominant Frequency Ratio (DFR)	Measures the ratio of the maximum harvested energy (2.4 GHz) among all radio frequencies in the spectrum.	✗
	Entropy (entropy)	Measures the spectral power distribution of the harvested voltage signal. $entropy = -\sum_{i=1}^{L} Pn_i log_2(Pn_i)$, Pn_i represents the normalized magnitude of FFT coefficient.	✓

Table 7: Cost of each component in the *Burger Model*

Component	Market Price (USD)
PCC110 RF-DC Converter	1.99
2.4GHz Dipole Antenna	4.99
Self-designed PCB Board	0.50
Arduino UNO Microcontroller (MCU)	14.99
16GB SD Card & SD Card Shield	3.95
AD620 Amplifier	2.55
BLE Communication Module	5.99
Other (Electronics, ADC)	1.50
Total Cost	**36.46**

Table 8: Harvested voltage & Accuracy vs. Blocking items.

Blocking item	Thickness (cm)	Harvested voltage (mV)	Acc. (%)
Non-blocking	–	429	98.4
Partition board	2.8	359	97.7
Wooden door	6.1	241	96.8
Thin wall	8.0	122	93.1
Thick wall	27.4	0	–

Table 9: Results of victim activity identification

Category	App name	Victim activity (# Identified actions / # All actions)					Acc. (%)
Video		Play	Next	Pause	Forward	Backward	
	YouTube	51 / 53	48 / 52	49 / 50	14 / 20	12 / 20	89.2
	TikTok	72 / 75	17 / 20	26 / 30	7 / 14	8 / 15	84.4
	Netflix	76 / 80	14 / 15	21 / 23	7 / 10	5 / 8	90.4
	Vimeo	60 / 65	24 / 28	10 / 11	8 / 12	7 / 11	81.6
	Hulu	72 / 74	15 / 21	19 / 19	9 / 16	10 / 15	86.2
	Disney+	67 / 72	15 / 18	22 / 25	6 / 9	6 / 10	86.5
	TED Talk	50 / 55	24 / 27	18 / 21	12 / 19	11 / 19	81.3
	Twitch	42 / 48	26 / 30	15 / 18	10 / 14	7 / 13	81.3
Music		Play	Next	Pause	Forward	Backward	
	Spotify	33 / 36	12 / 12	5 / 5	8 / 10	7 / 10	89.0
	Apple Music	43 / 46	4 / 6	5 / 6	11 / 11	10 / 12	90.1
	YouTube Music	36 / 40	5 / 8	6 / 6	10 / 10	6 / 9	86.3
	Shazam	29 / 33	17 / 20	6 / 7	9 / 11	7 / 10	84.0
	SoundCloud	42 / 50	7 / 7	3 / 3	6 / 10	4 / 8	79.5
	QQ Music	37 / 40	9 / 9	9 / 11	8 / 12	9 / 9	88.9
	Netease Cloud	28 / 35	13 / 15	8 / 10	7 / 11	7 / 10	77.8
	Kugou Music	27 / 33	14 / 15	6 / 8	6 / 9	7 / 10	80.0
Social Media		Thumb-up	Comment	Repost	Refresh	Share	
	Facebook	15 / 20	45 / 50	9 / 11	10 / 10	7 / 8	86.9
	Twitter	7 / 10	18 / 22	14 / 15	10 / 12	6 / 8	82.1
	Instagram	12 / 15	25 / 30	11 / 13	9 / 10	9 / 11	83.5
	LinkedIn	13 / 18	38 / 42	9 / 12	10 / 11	9 / 9	85.9
	Reddit	22 / 26	28 / 30	10 / 12	10 / 10	8 / 9	89.7
	Quora	21 / 23	20 / 25	16 / 18	14 / 17	13 / 15	85.7
	Sina Weibo	11 / 12	35 / 42	9 / 10	9 / 11	7 / 10	83.5
	Pinterest	13 / 16	40 / 45	13 / 15	10 / 10	8 / 8	89.4
Communication		Send text	Send images	Send videos	Send voice	Video call	
	WhatsApp	13 / 15	24 / 29	35 / 38	9 / 11	14 / 15	88.0
	Telegram	24 / 25	26 / 34	28 / 39	7 / 9	12 / 14	80.2
	Line	16 / 20	27 / 30	28 / 34	8 / 10	9 / 11	83.8
	WeChat	13 / 18	24 / 26	30 / 36	10 / 10	9 / 10	86.0
	Messenger	17 / 23	25 / 32	24 / 25	14 / 16	13 / 17	82.3
	Hangouts	12 / 20	22 / 24	22 / 30	11 / 13	13 / 16	77.7
	Snapchat	14 / 15	18 / 20	39 / 42	9 / 9	13 / 14	93.0
	Discord	23 / 23	29 / 31	28 / 29	14 / 18	13 / 15	92.2
Game		Loading	Entering	Matching	Gaming	Exit game	
	Arena of Valor	6 / 7	5 / 5	29 / 29	33 / 33	4 / 5	97.5
	Lol Wild Rift	6 / 8	5 / 5	29 / 31	41 / 41	5 / 6	94.5
	FIFA Soccer	9 / 10	6 / 6	30 / 33	36 / 36	5 / 7	93.5
	Hearthstone	8 / 9	5 / 6	29 / 30	40 / 40	6 / 8	94.6
	PUBG Mobile	8 / 8	4 / 6	23 / 25	30 / 34	10 / 10	90.4
	Genshin Impact	14 / 17	5 / 5	23 / 27	35 / 38	5 / 9	85.4
	Minecraft	10 / 12	6 / 6	30 / 32	42 / 45	9 / 9	93.3
	UNO	10 / 11	7 / 7	26 / 29	28 / 31	7 / 8	90.7