Precise Wireless Camera Localization Leveraging Traffic-Aided Spatial Analysis

Yan He, Qiuye He, Song Fang, and Yao Liu

***Abstract*—Wireless cameras nowadays commonly employ motion sensors to identify that something is occurring in their fields of vision before starting to record and notifying the property owner of the activity. In this paper, we discover that the motion sensing action can disclose the location of the camera through a novel wireless camera localization technique we call *MotionCompass*. By creating motion stimuli and sniffing wireless traffic for a response to that stimuli, a user can obtain the motion trajectories within the motion detection zone and then use them to calculate the camera's location. We also extend the camera localization algorithm to pinpoint cameras in always-active mode. We develop an Android app to implement *MotionCompass*. Our extensive experiments using the developed app and 18 popular wireless cameras demonstrate that for cameras with one motion sensor, *MotionCompass* can attain a mean localization error of around 5 cm with less than 140 seconds. We also discuss defenses against *MotionCompass*. Our localization technique builds upon existing work that detects the existence of hidden cameras, to pinpoint their exact location.**

***Index Terms*—Motion sensor, wireless traffic analysis, variable bit rate, hidden camera, localization.**

I. Introduction

DUE to their flexibility and greatly simplified installation, wireless security cameras are becoming more widely deployed than traditional wired cameras to monitor and report trespassing or other unauthorized activity. It is forecasted that the global wireless video monitoring and surveillance market will increase at a high annual growth rate of 16.85% over 2017 to 2023 in a market survey [2].

Some wireless security cameras are made visible as a deterrence measure, but that visibility may mean (1) they are more susceptible to damage or theft [3]; (2) burglars may become more interested in breaking in as they think the camera signals that there are valuables inside the property [4]; (3) it is easier to avoid being recorded, e.g., an adversary may find the blind spots (i.e., areas not within the vision of the camera) and leverage them to evade being recorded [5]. For these reasons, people may install wireless cameras inconspicuously. They are thus naturally attractive targets to adversaries who want to bypass the surveillance.

The rapid proliferation of wireless cameras also brings privacy concerns associated with unauthorized video recording [6], [7]. These cameras can be kept hidden easily such that their targets are unaware of their existence. According to a survey of 2,023 Airbnb guests that was conducted in 2019, 58% of them were concerned that property owners might install hidden cameras inside their rooms, and meanwhile as high as 11% said that they had discovered a hidden camera in their Airbnb [8]. As thus, the detection of wireless cameras is drawing increasing attention for privacy protection [9], [10], [11], [12], [13], [14].

Traditional ways to detect a wireless camera mainly include Radio Frequency (RF) scanning, lens detection, and physical search. The latter two methods are cumbersome as they require inspecting every corner of the target area. RF scanning works when the camera is actively transmitting, but existing work (e.g., [15]) can only detect the existence of wireless cameras but cannot tell their exact locations.

Personal privacy is improved by identifying if a wireless camera exists in various locations, such as hotel rooms, Airbnb rentals, and office buildings. However, detection is not sufficient on its own, as the camera owner may claim it is somewhere outside of the room or installed by another. We argue that it is significantly important to pinpoint the locations of hidden wireless cameras. For example, a victim whose privacy is violated can obtain direct and solid evidence by finding the physical camera that records. We also realize that this localization technique is a two-edged sword in that it can also be utilized by malicious users such as a burglar localizing a home's security camera in order to avoid its field of view or otherwise physically disarming it.

Many wireless cameras are equipped with built-in motion sensors, such as best-selling ones – Amazon Blink XT2 [16] and Arlo Pro 2 [17]. Because of the volume of collected data, wireless cameras remain in standby mode until movement is detected, at which point the camera turns on and starts recording, uploading captured video to the cloud backend server, and sending a notification to the property owner. The network correspondingly exhibits sudden high wireless traffic. The camera will then continue to record until the motion stops. After that, it reverts to standby mode. As wireless security cameras installed at different locations have different coverage areas, we can then determine the camera's coverage area to find the location of the camera. Specifically, we artificially induce motion at a spot (e.g., asking a helper to walk or utilizing a moving robot/drone); if we observe a

Manuscript received 24 January 2023; revised 6 November 2023; accepted 7 November 2023. Date of publication 15 November 2023; date of current version 7 May 2024. This work was supported in part by the National Science Foundation under Grants 1948547 and 2155181. An earlier version of the work was presented in ACM MobiSys'21. [DOI: 10.1145/3458864.3467683]. Recommended for acceptance by P. Casari. *(Corresponding author: Song Fang.)*

Yan He, Qiuye He, and Song Fang are with the School of Computer Science, University of Oklahoma, Norman, OK 73019 USA (e-mail: heyan@ou.edu; qiuye.he@ou.edu; songf@ou.edu).

Yao Liu is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: yliu@cse.usf.edu).

Digital Object Identifier 10.1109/TMC.2023.3333272

correspondingly high wireless traffic, we know that this spot may be monitored by a camera, and may determine the camera's possible area accordingly. With customized motion trajectories, we can then narrow down until pinpoint the location of the camera.

In practice, there exist other types of wireless traffic flows generated by non-camera devices, such as laptops, smartphones, or tablets. The wireless local area networks (WLANs) employ encryption techniques such as WEP, WPA, and WPA2 to encrypt transmitted data packets [18]. However, the header information is not, which exposes link-layer Media Access Control (MAC) addresses to eavesdroppers [19]. Each MAC address, which is a persistent and globally unique identifier, discloses the device manufacturer information via the first three most significant MAC bytes, i.e., the Organizationally Unique Identifier (OUI). We thus utilize the OUI of MAC addresses to detect the existence of wireless camera traffic. Specifically, as OUI relies on the camera manufacturers and the mainstream manufacturers of wireless cameras are limited [11], we can first pre-build a table of OUIs associated with wireless cameras. Next, with a captured MAC, we compare its OUI with each entry in the table. If a match is found, we assume that the wireless traffic with this specific MAC is generated by a wireless camera. MAC address eavesdropping and analysis can reveal the existence of wireless cameras while it is unable to reveal their exact locations. Thus, to pinpoint the camera, we design the motion stimulation and correlate its range with the camera location. In particular, we design novel strategies to first set up a coordinate system and then compute the camera's coordinates for determining its location.

Our major contributions are summarized as follows:

- Unlike previous extensive research in hidden wireless camera detection, this paper is the first to provide a practical approach to pinpoint a motion-activated wireless camera. The proposed technique can be carried out with a single smartphone, and it needs neither professional equipment nor connecting to the same network with the target camera.
- We exploit how a motion sensor can act as a compass to guide us to pinpoint the wireless camera by correlating the manipulated motion with the resultant wireless traffic generated by the camera.
- We explore the cases when the camera is in always-active mode, and craft corresponding methods to achieve the camera localization under such cases.
- We implement *MotionCompass* and develop an Android application for validating effectiveness and efficiency. We also discuss defense mechanisms against the proposed camera localization technique.

II. Preliminaries

Motion Sensing of Wireless Cameras: We usually cannot keep our eyes glued to our camera's feed on a phone or computer. To get rid of this limitation, a wireless camera, incorporating a motion sensor, provides a practical solution. There are various types of motion sensors, such as passive infrared (PIR), ultrasonic, microwave, and tomographic. A PIR sensor includes a pyroelectric film material, which is sensitive to radiated heat power fluctuation [20] and converts infrared radiation into electrical signals. It can thus detect the presence of humans or other warm-blooded living beings from the radiation of their body heat [21]. Due to its properties of small size, low cost, power efficiency, and being able to work in dark environments, PIR sensors are widely used in wireless cameras. Without loss of generality, in this paper, we explore the localization of wireless cameras equipped with this type of motion sensor.

Video Encoding: Video compression (i.e., coding) algorithms attempt to reduce redundancy and store information more compactly. There are two common types of video encoding schemes, Constant Bit Rate (CBR) and Variable Bit Rate (VBR). Bit rate stands for the amount of data that is transferred in a file over a period of time. CBR means that the rate at which the codec's output data is constant. As opposed to CBR, VBR varies the bitrate of an encoded video with its content. Considering streaming efficiency, VBR has been adopted by all popular streaming services [22], [23], [24]. Accordingly, VBR-encoded videos may leak information about the content [22], [23], [24]. When an activated wireless camera (e.g., in always-activate mode) is monitoring a static scene, the generated traffic is fairly constant due to the constant content, while if we create motion disturbance in the monitored area, the traffic varies accordingly. In this paper, such correlation between the change of the wireless traffic and the location of the artificial motion will be utilized to determine the monitored area of the camera and further localize the camera.

III. Attack Model and Assumptions

We consider a general scenario, where a user deploys a motion-activated wireless security camera to monitor a target area. The user aims to keep the camera hidden to avoid being noticed. An adversary aims to pinpoint the location of the camera with the *MotionCompass* technique. *MotionCompass* can also be utilized to find hidden cameras in untrustworthy hotels or Airbnb rooms, in which cases the roles of "attacker" and "legitimate user" are reversed, but for consistency and to prevent confusion, we will use these roles as just introduced.

We assume the adversary has the capability to sniff the wireless traffic, and can also safely perform some motion around the camera without being caught. For example, the adversary can ask a helper to walk or use a moving robot/drone to introduce manipulated movement. We also assume that the adversary can move at a known speed and record the time elapsed so that she can measure the movement distance. This can be achieved for example by using an Android app to log all the accelerometer readings for calculating the speed.

IV. Camera Localization

A. Overview

MotionCompass includes three important phases, i.e., camera traffic finder, camera MAC extraction, and camera traffic manipulation. The first one determines wireless traffic associated with wireless cameras. When the user introduces motion activity within an interested area, if there is a camera monitoring this area, the user would observe a wireless traffic pattern that is

highly correlated with the movement trajectory. To eliminate the interference of motion-activated non-camera devices (e.g., smart WiFi motion sensor [25]), we utilize the second phase, which first collects all MACs embedded in each traffic flow and then searches the OUI of each MAC in a table consisting of all OUIs assigned to cameras. If a match is found, the MAC would be regarded as belonging to a camera. The extracted MACs would be the input of the final phase, and all traffic flows with them would be monitored. The user then performs motion along specifically designed paths, and pinpoints the camera's location by correlating the manipulated motion paths with the wireless traffic that the monitored camera generates.

B. Camera Traffic Finder

As there are numerous wireless traffic flows in the air, we need to shrink the candidates for wireless camera traffic.

1) Coarse-Grained Activation: Wireless cameras are usually battery-powered and sit in standby mode to conserve battery. In standby mode, only a "heartbeat" signal of small size is sent out at a regular interval in the order of seconds, indicating normal operation or synchronization with the other party. They begin to record and send the owner push notifications when detecting motion (i.e., activation signals). These videos are then sent to the cloud backend server for secure storage in the owners' library. Thus, an attacker can manually generate the activation signals in a target area. If the motion happens to be performed in the motion activity zone, the camera recording will be then triggered [26], and abnormally high wireless traffic would be generated accordingly.

2) Traffic Candidates Determination: When the camera is in standby mode, the microcontroller unit (MCU) consumes less power and only processes data monitored by the built-in microphone or motion sensor. Once the activation signal is detected, the MCU awakens the Complementary Metal Oxide Semiconductor (CMOS) sensor to start recording until motion stops, and meanwhile enables wireless networking module to send out the recorded video. The generated traffic thus exhibits a distinguishable pattern, i.e., the volume of camera traffic depends on whether the camera is activated or not.

The specific pattern of camera traffic provides an adversary an opportunity to correlate the intentional activation with the existence of a wireless camera. If monitored wireless traffic suddenly becomes faster when a motion is performed and slower when the motion stops, this traffic flow can be determined as a candidate for the camera traffic.

C. Camera MAC Extraction

For any six-byte MAC address, the first half is the Organizationally Unique Identifier (OUI) [27], indicating the device manufacturer; and the second half represents the unique device ID. MAC address intends to be permanent and unique. Thus, the prefix (i.e., OUI) of a wireless camera's MAC address is fixed. The phase of camera MAC extraction aims to extract the MAC address of the target camera. To achieve this goal, we can compare the prefix of each MAC extracted from candidate camera traffic flows with those publicly available prefixes (e.g., [27]) defined by SoC suppliers to determine whether the monitored traffic belongs to a wireless camera.

1) Collection of MAC Identifiers: IEEE 802.11 wireless protocols are utilized in almost all commodity network devices [28]. The use of 802.11, however, may cause exposure of MAC addresses [19], and an eavesdropper within the radio range of a wireless transmitter can capture the corresponding MAC address. Though the data packets generated by the wireless camera are encrypted [29], the raw wireless frames are broadcasted over the air and the camera transmits its unencrypted MAC (i.e., Source Address) in the header of the 802.11 MAC frame.

To capture the raw wireless frames of the camera, we should first know the channels that the camera operates on. Wireless sniffing tools (e.g., Airmon-ng toolkit [30] which is open source) can capture raw 802.11 MAC frames and thus help determine all active channels nearby. The problem then becomes how to sort out the data frames generated by the camera from packets generated by various other devices that pass the first phase.

2) Camera MAC Match: We build a table containing OUIs of all cameras on the market, called *camera-labeled OUI table*, and utilize it to determine whether the monitored traffic belongs to a wireless camera. If the OUI of a MAC extracted from a monitored packet can be found in the OUI table, this corresponding traffic is regarded as being generated by a camera.

MAC Spoofing: Some devices may enable the user to change the MAC arbitrarily in software [31], [32]. Thus, the user may use a non-camera-manufacturer-based OUI for the camera to bypass the camera traffic detection, or use a camera-manufacturer-based OUI for a non-camera device to slow down the localization process.

Since the packets generated by wireless cameras are encrypted and the network of the hidden devices is inaccessible, traditional traffic flow classification methods (e.g., IP address, protocol) using 5-tuple (source IP and port, destination IP and port, and IP protocol) or a 3-tuple (source IP, destination IP, and IP protocol) [33] do not apply. Recent studies have proposed techniques using a unique identifier called the Universally Unique IDentifier-Enrollee (UUID-E), which is derived from a device's original MAC and can thus help recover the device's true MAC from the spoofed one [19], [32], [34]. However, the UUID-E based method only works for devices that support Wi-Fi Protected Setup (WPS) transmit a UUID-E in probe requests, and it requires significant processing time to pre-compute a lookup table of the device's UUID-E [34]. Also, a recent study [35] reveals that many latest devices do not send WPS elements, including UUID-E, leading to the failure of the UUID-E based method.

Alternatively, we can utilize another general and robust solution to handle MAC spoofing. Systems-on-a-chip (SoCs) provide video encoding and data transfer functionality for wireless cameras. Thus, the traffic patterns of wireless cameras highly depend on the corresponding SoCs. As most SoCs take similar encoding methods (e.g., H.264, H.265, and MJPEG), which may affect data packet size and processing overhead over a period of time, the resultant traffic patterns are similar as well [11]. This observation motivates us to train a Support Vector Machine (SVM) model to classify traffic patterns and

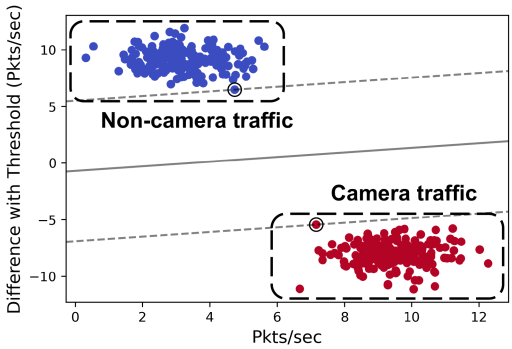


Fig. 1. SVM training result for wireless traffic flows.

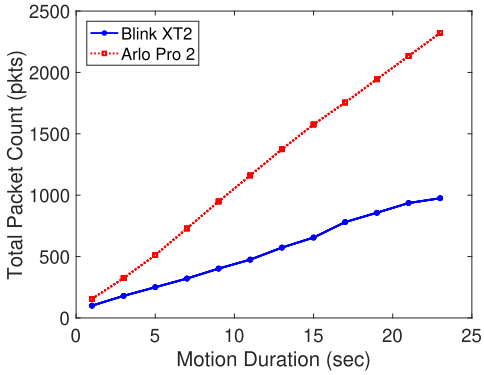


Fig. 2. Total packet count versus motion duration.

utilize it to determine whether each captured traffic belongs to a wireless camera.

The SVM classifier model is formed using the Scikit-learn libraries with Python 3.8.1. We set a threshold based on the average value of data transmission rates of various wireless devices in the environment. For each traffic flow, we extract its data transmission rate along with the difference between this rate and the threshold. Fig. 1 shows the result of running SVM on 400 traffic flows coming from both wireless cameras and non-camera devices, demonstrating the success of distinguishing traffic flows generated by wireless cameras.

D. Camera Traffic Manipulation

1) Correlation Between Traffic and Motion: We first set up a listener with existing tools to monitor the traffic transmitted from all candidate cameras and then observe the traffic change of each channel when we provoke the system with manipulated motion. If a camera generates traffic volume corresponding to the time that the manipulated activity lasts, we know that the camera is monitoring the area where the activity is performed. Otherwise, if the monitored traffic has no change, we can determine that the candidate camera is not monitoring the area. Motivated by this observation, we develop a customized algorithm to shrink the possible camera candidates and localize the target camera by feeding manipulated stimuli to the motion sensor and observing resultant traffic volume variation.

Empirically, we find the longer the motion duration is, the more (cumulative) packets the camera generates. We install an Amazon Blink XT2 camera and Arlo Pro 2 camera on the wall with a downward angle and monitor the activity in the detection area, respectively. For each scenario, we monitor the traffic generated by the camera and record the corresponding amount of the transmitted packets when a user passes nearby within different durations (i.e., manually producing activity within the coverage range of the motion sensor). Fig. 2 presents the variation of total packet count with the motion duration for the two different cameras. The obtained packet count shows a nearly linear correlation with the motion duration.

Camera Activation Detection: The discovered correlation between exposure time (the duration when the camera is activated) and total packet count can be then explored to determine whether the camera is activated by a user when she is able to sniff wireless traffic and obtain the total packet count. Specifically, in this work, we consider a *linear function approximation* architecture where the count N of network packets generated by a wireless camera is approximated as

$$N = a + R \cdot \Delta t, \tag{1}$$

where a is a constant, R is the throughput (i.e., the rate at which the activated camera generates packets), and Δt denotes exposure time. We can then determine whether the performed motion is still in the detection range of the motion sensor: if the observed total packet count does not fit the linear model with a significant deviation, the current motion will be determined as out of the detection range.

Typically, the field of view of a PIR sensor is at 105° or 110° horizontally and 80° vertically. If more PIR sensors are utilized simultaneously, the corresponding detection range can be wider. For example, the Arlo Ultra camera has dual PIR sensors and has a horizontal angle of view of 150° [36]. We consider a camera deployed on a vertical wall (which aligns with most practical scenarios). Note for other cases, we can regard that the camera is deployed on a virtual wall (i.e., a plane perpendicular to the floor). Thus, the camera localization problem can be converted to computing the coordinates of the camera, when the bottom left corner of the wall is regarded as the Origin.

2) Coordinates Calculation. Special Case: To obtain the maximum horizontal breadth, the camera body is often mounted perpendicular to the wall. Also, in the general case, the camera can be swiveled in any direction and mounted at any angle to the wall as long as its view is not obstructed by the wall. We first address the special case when the camera is mounted perpendicular to the wall.

We propose a two-step procedure to pinpoint the camera. As shown in Fig. 3, a user can perform motion along two paths with an average speed of v and simultaneously monitor the wireless traffic, including,

1. Moving parallel to the wall from left to right (or in the opposite way), as shown in Fig. 3(a): when the traffic indicates that the user enters and leaves the detection range, the respective locations are marked as A and B; the user also tracks the corresponding time t_1 and t_2 for calculating

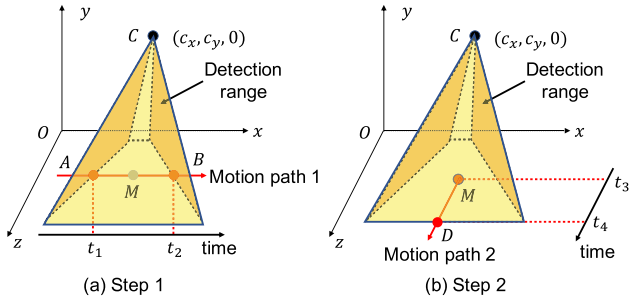


Fig. 3. Two-step procedure for the special case.

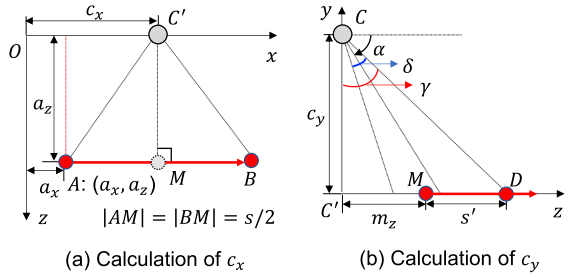


Fig. 4. Coordinates calculation for the special case.

the walking distance s within the detection range, i.e., $s = |AB| = v \cdot (t_2 - t_1)$.

2. Vertically getting out of the detection range at time t_3 with the start location at the midpoint M of the line segment $\overline{AB}$, as shown in Fig. 3(b): when it is determined that the user leaves the detection range at time t_4, the location is marked as D; the new walking distance s' within the detection range can be computed as $s' = |MD| = v \cdot (t_4 - t_3)$.

With step 1, we can obtain the x-axis coordinate c_x of the camera location (i.e., point C). For better understanding the calculation process, we plot the motion path 1 in the xz-plane, as shown in Fig. 4(a), where C' denotes the projection of the camera location onto the x-axis. Assume that the horizontal distance between location A and the z-axis is a_x, and the distance between location A and the wall is a_z. Both a_x and a_z can be easily measured by the user. Thus, we can calculate the camera's x-coordinate as

$$c_x = a_x + s/2. \tag{2}$$

With only motion path 1, we cannot determine the height c_y of the camera location. Thus, we perform motion path 2 beginning with M towards the outer edge of the detection range (i.e., the line MD is perpendicular to the x-axis). To demonstrate how to calculate c_y, similarly, we plot the plane through the points C, M, and D, as shown in Fig. 4(b).

A general rule of thumb is to install the camera at a downward angle for better monitoring the target area. Let α denote the camera installation angle, which is the angle between the camera optical axis (i.e., the direction that the camera faces) and the ground. Also, we use δ to represent the vertical field of view of the camera's motion sensor. The camera optical axis divides δ into two equal angles. With step 1, we can obtain the z-coordinate m_z of point M, which equals a_z. Let γ denote angle $\angle DCC'$. We thus have

$$\begin{cases} \gamma = \left(\frac{\pi}{2} - \alpha\right) + \frac{\delta}{2} \\ \tan\gamma = \frac{m_z + s'}{c_y} \end{cases}. \tag{3}$$

We can then compute the camera's y-coordinate as

$$c_y = \frac{a_z + s'}{\tan[(\pi - 2\alpha + \delta)/2]}. \tag{4}$$

Impact of a Time Delay for Traffic State Transition: Let τ denote the time delay for the user (receiver) to realize the transition of the camera's traffic state.

Due to the existence of the time delay, the real locations where the user enters and leaves the detection range should be A' and B', as shown in Fig. 5, and the corresponding time points for traffic to reach the activated and standby states are $t_1' = t_1 - \tau$ and $t_2' = t_2 - \tau$, respectively. Accordingly, the distance when the user moves during the time delay is $v\tau$, and thus the $x-$coordinate of A' is $a_x' = a_x - v\tau$. The walking distance s is unchanged since $s = v \cdot (t_2' - t_1') = v \cdot (t_2 - t_1)$. In this way, we can calculate the camera's x-coordinate as

$$c_x' = a_x' + s/2 = c_x - v\tau. \tag{5}$$

The "motion path 2" starts from the location M, where the traffic has already reached the activated state and there is thus no time delay for traffic state transition. However, when the user leaves the detection range, the time delay exists since the traffic may not immediately reach the standby state. The distance of path $|MD'| = v(t_4 - \tau - t_3) = s' - v\tau$. Similarly, M' is the midpoint of $|A'B'|$. As the line $M'D_1'$ is parallel to the line MD' (i.e., $M'D_1' \,//\, MD'$), we have $|M'D_1'| = |MD'| = s'$. Note that the boundary of the real motion detection range along the x-axis may not be a straight line segment, and this approximation could introduce localization errors.

With $\angle D_1'C_1C_1' = (\pi - 2\alpha + \delta)/2$ and $\tan\angle D_1'C_1C_1' = \frac{|D_1'C_1'|}{|C_1'C_1|}$, we further obtain $\tan[(\pi/2 - \alpha) + \delta/2] = (m_z' + s' - v\tau)/c_y$, we can then compute the camera's y-coordinate as

$$c_y' = \frac{a_z + s' - v\tau}{\tan[(\pi - 2\alpha + \delta)/2]} = c_y - \frac{v\tau}{\tan[(\pi - 2\alpha + \delta)/2]}. \tag{6}$$

Usually, when the delay is small, the resultant localization error would be small. For example, suppose the time delay is $\tau = 10$ ms and the user moving speed is $v = 1$ m/s. We consider $\alpha =$

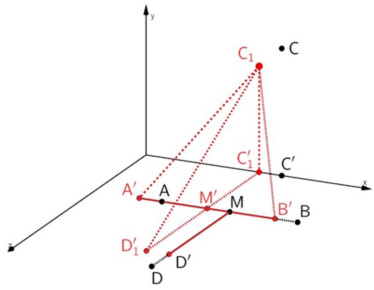


Fig. 5. Scenario with a time delay for traffic state transition.

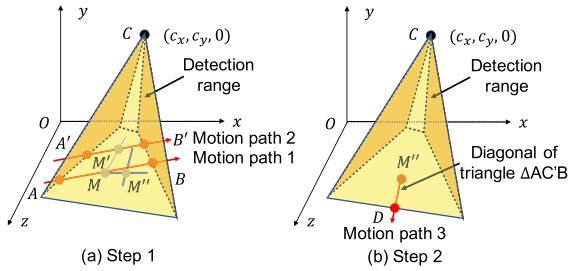


Fig. 6. Improved two-step procedure for the general case.

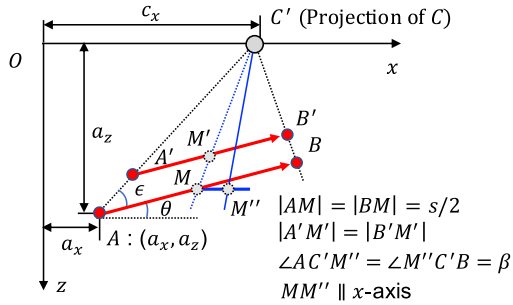


Fig. 7. Calculation of c_x for the general case.

50° and $\delta = 80^\circ$. Under such settings, the localization errors in x-coordinate, y-coordinate, and absolute location equal 0.01, 0.0018, and 0.0102 m, respectively.

Delay Removal: The corresponding localization error increases with the time delay τ. When τ is large, its impact on the localization error is non-negligible. Fortunately, such a delay can be easily verified and compensated. Specifically, after performing "motion path 1" that goes through locations A and B in sequence, the user can go through the same path in the opposite direction, which goes through location B first and then location A. Similarly, the locations where the monitored traffic indicates that the user enters and leaves the detection range should be B_0 and A_0. If B_0 and B (or A_0 and A) share the same location, there is no time delay (i.e., $\tau = 0$); otherwise, we have $\tau = \frac{|B_0B|}{2v} = \frac{|A_0A|}{2v}$. By considering τ, we can then update the real coordinates of points A and B and continue the procedure; for "motion path 2", we can then find the real coordinate of point D. Consequently, we can still apply Equations (2) and (4) to reveal the camera's location.

3) Coordinates Calculation. General Case: As aforementioned, the camera may not be necessarily mounted perpendicular to the wall, and can be pivoted to the left or right. As a result, with the above two-step procedure, the camera may not have the same x-axis coordinate with the midpoint M of motion path 1. This is because the line MC' (C' is the projection of the camera location C onto the x-axis) is not necessarily perpendicular to the x-axis.

We then propose an improved two-step procedure for determining the coordinates of the camera. Specifically,

1. As shown in Fig. 6(a), the user moves from left to right (or vice versa) twice and generates two parallel motion paths. Similarly, the user records the locations when she enters and leaves the detection range for each motion path i.e., A, B, A' and B'. The midpoints of the line segments $\overline{AB}$ and $\overline{A'B'}$ are denoted with M and M'. The walking distance $|AB|$ is denoted as s. The user then draws a line through M and M', and it intersects the x-axis at the point C' (i.e., the projection of point C onto the x-axis). The user can then measure angle $\angle AC'B$ and find its angle bisector. The user draws another line parallel to the x-axis and through the point M, and it will intersect the above angle bisector at a point, denoted with M''.
2. As shown in Fig. 6(b), the user then gets out of the detection range at time t_5 with the start location at point M'' along the angle bisector (i.e., line $C'M''$) of angle $\angle AC'B$. When it is determined that the user leaves the detection range at time t_6, the location is marked as D. The walking distance s' in this step can be measured as $s' = |M''D| = v \cdot (t_6 - t_5)$.

With the first step, we can calculate the x-axis coordinate c_x of the camera. As shown in Fig. 7, the user can measure the angle θ between the first or second motion path and the x-axis, and the angle ϵ between $C'A$ and AB. Thus, we have $\tan(\epsilon + \theta) = \frac{a_z}{c_x - a_x}$, and obtain

$$c_x = a_x + \frac{a_z}{\tan(\epsilon + \theta)}. \tag{7}$$

Meanwhile, we can calculate the z-coordinate m_z of point M as $a_z - \frac{s}{2} \cdot \sin\theta$. As the line MM'' is parallel to the x-axis, the z-coordinate m_z'' of point M'' equals m_z. Let $\beta = \frac{\angle AC'B}{2}$, and we then have $\angle OC'M'' = \angle OC'A + \angle AC'M'' = \epsilon + \theta + \beta$. As a result, we obtain $|C'M''| = \frac{m_z}{\sin(\epsilon+\theta+\beta)}$.

The calculation process of the camera's y-coordinate c_y is similar to that in the special case. $\triangle CC'D$ is a right triangle where angle $\angle CC'D$ is the right angle, and the angle $\angle DCC'$ (i.e., γ) is the same for both cases, i.e., $\gamma = (\frac{\pi}{2} - \alpha) + \frac{\delta}{2}$. We then have $\tan\gamma = \frac{|C'M''| + |M''D|}{|C'C|}$, and thus obtain

$$c_y = \frac{(2a_z - s \cdot \sin\theta)/(2 \cdot \sin(\epsilon + \theta + \beta)) + s'}{\tan[(\pi - 2\alpha + \delta)/2]}. \tag{8}$$

Similarly, with the method described in Section IV-D2, we can easily determine whether there is a time delay for traffic state transition and further remove its impact if yes.

4) Dealing With Always-Active Mode: When the hidden camera is switched to always-active mode, the traffic volume is no longer able to indicate whether the camera is activated or on standby. Thus, except for using the camera status, a new method is needed to determine whether the motion is in the detection zone of the camera.

Most wireless cameras utilize VBR encoding (as described in Section II), where slow scenes use fewer bits than fast-moving, action-packed scenes [37]. We then create artificial motion and compare the bitrate streams at moments with and without motion. Action scenes (e.g., when the camera captures the crafted motion) are encoded with a higher bitrate than slower scenes (e.g., when the camera does not record the crafted motion). Thus, if the motion does happen within the field of view of the camera, the user would obtain a higher burst than normal. This

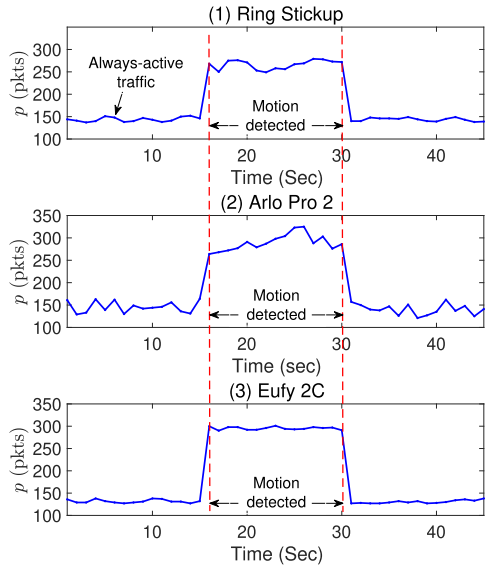


Fig. 8. Identifying action scenes via VBR video traffic for three different wireless cameras (Ring Stickup, Arlo Pro 2, and EufyCam 2 C).

observation enables the user to successfully infer whether the motion is within the detection zone of the camera. The motion trajectory and VBR leakage are then correlated to determine the location of the wireless camera based on the built model.

To demonstrate the impact of VBR encoding, we monitor the traffic generated by three different wireless cameras, which support VBR encoding. Specifically, we install a Ring Stickup camera, an Arlo Pro 2 camera, and a Eufy 2 C camera to monitor the same area. The motion sensors of the three cameras are all turned off. Instead, we turn on the live monitoring view to see real-time camera footage over the Internet, and thus all cameras are in always-active mode. In the beginning, the monitored area is static, i.e., without motion. Then, we let the user walk within the cameras' field of view for 15 seconds. The user then leaves. Fig. 8 presents the variation of the count (p) of captured new packets over time for the three selected cameras. We observe that even when no motion is captured by the wireless cameras, the measured amount of traffic for each camera maintains at a high level (i.e., a normal state). However, when a user continuously walks in the field of view of the cameras (from the 16th to 30th second), an obvious traffic burst can be observed. When the motion stops, each camera's traffic throughput returns to its normal state.

To localize the camera, we manipulate motion (i.e., simulate action scenes) to trigger variable bitrate streams along specified routes, and reuse the proposed two-step procedure for the special case (i.e., when the camera is mounted perpendicular to the wall) in Section IV-D2 together with the improved two-step procedure for the general case (i.e., when the camera body may be pivoted to the left or right) in Section IV-D3. As the field of view of a camera is often larger than that of a PIR sensor (e.g., [38]), we thus need to change the field of view of the motion sensor into that of the camera within both algorithms for accurate localization results. Meanwhile, to avoid inference to the correlation between the generated motion and the observed traffic, we also need to make sure that when performing specified activities, there is no other motion that cannot be controlled in the monitored area.

TABLE I
LIST OF WIRELESS SECURITY CAMERAS WE TEST

ID	Model
1-15	AIVIO Cam, Arlo Essential, Arlo Pro 2, Arlo Pro 3, Blink Indoor, Blink XT2, Blue by ADT, Canary Flex, Conico Cam, EufyCam 2C, Reolink Argus 2, Reolink Argus Pro, Ring Door View, Simplisafe Cam, Swann Wireless
16-18	Arlo Ultra, Ring Spotlight, Ring Stickup Cam

V. EXPERIMENTAL EVALUATION

We develop a phone (Android) app to implement *MotionCompass*, and run the app on rooted Android platforms. The default mode for a smartphone's NIC is managed mode, in which it only listens to the traffic that comes for it and discards other packets. *MotionCompass* needs the NIC to be in monitor mode. We thus achieve the monitor mode function based on Airmon-ng tools and the Android device running the Kali NetHunter, which is a popular open-source Android ROM penetration testing platform [39].

A. Evaluation Setup

The adversary first scans the possible MACs for wireless cameras and then performs motion to stimulate the camera. By measuring the distances of performed motion paths, as well as the initial parameters, such as the coordinates (a_x, a_z) of the start point within the camera detection range when the attacker introduces motion along the first motion path, and the angle θ between the first motion path and the wall, the adversary can then calculate the camera's location.

Testing Cameras and Scenarios: We test 18 most popular wireless cameras, as shown in Table I. Those cameras can be divided into two groups: G1 consisting of cameras (ID 1-15) with one motion sensor, and G2 including cameras (ID 16-18) with two motion sensors. Two scenarios are tested:

- *Outdoor:* We conduct the experiment outside a typical American single-family house. The camera is installed at five different locations on the front outside wall (with a width of 10 m and a height of 5.5 m).
- *Indoor:* We select a bedroom and place the camera at five different locations: two in the top-left and top-right corners of an inside wall (with a width of 5.5 m and a height of 2 m), one on top of the headboard, and two sitting on the nightstands beside the bed.

Fig. 9 shows the selected positions for the camera in the respective environment. The camera can be mounted at different angles along the wall. We do not consider the cases when most areas in the camera's field of view are obstructed by the wall, as the recording capability of the camera is highly restricted under these circumstances.

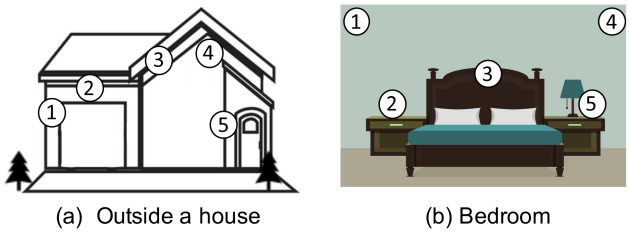


Fig. 9. Layout of the experimental environments.

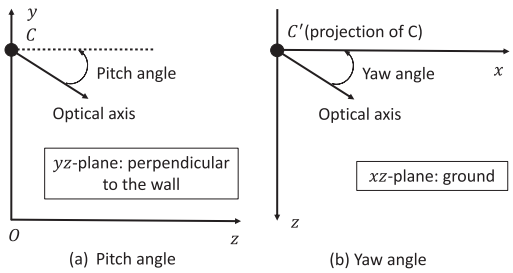


Fig. 10. Pitch and yaw angles (ϕ and ψ).

Metrics: We use the following two metrics.

- *Localization error:* This is measured as the Euclidean distance between the camera's estimated position and its corresponding true location.
- *Localization time:* This is the amount of time spent on obtaining the exact location of the camera.

B. Influential Factors

A Blink XT2 camera is installed at Location 2 of the house, as shown in Fig. 9(a). In this section, we discuss the impact of each influential factor.

Impact of the Camera's Pitch and Yaw Angles: The camera may have a non-right angle along the wall in the xz-plane (i.e., ground), and it may be pointed in different directions to adjust its monitoring area. We refer to the camera installation angle (discussed in Section IV-D), i.e., the angle between the ground and the camera as the *pitch angle*, denoted with ϕ, and the angles between the wall and the camera as the *yaw angle*, denoted with ψ, as shown in Fig. 10. The motion sensor usually has about 80° vertical field of view. The pitch angle can be thus adjusted from 0° to 50°; otherwise, the field of view would be obstructed by the wall. We vary ϕ from 0° to 50°, with increments of 5°. We maintain the yaw angle ψ fixed and perform 80 attempts to localize the camera for each ϕ. Similarly, considering that the motion sensor of the Blink XT2 camera has about 105° horizontal angle of view, we can adjust the yaw angle from 52.5° to 90°. We then vary ψ from 90° to 55°, with decrements of 5°. With a fixed ϕ, we perform 10 attempts to localize the camera for each ψ.

Figs. 11 and 12 present the corresponding estimation errors in x-coordinate, y-coordinate, and absolute location. We see that for a fixed yaw angle, these three errors have average values of 5.0, 5.7, and 7.8 cm, respectively; the location error remains

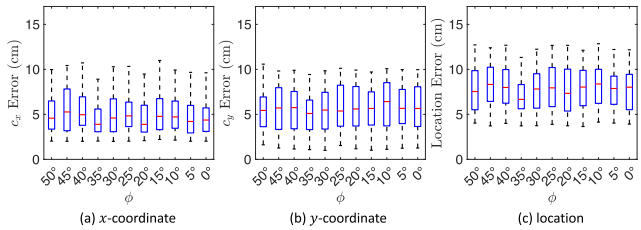


Fig. 11. Location errors versus pitch angle (ϕ).

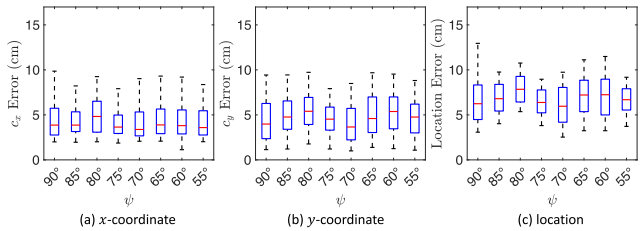


Fig. 12. Location errors versus yaw angle (ψ).

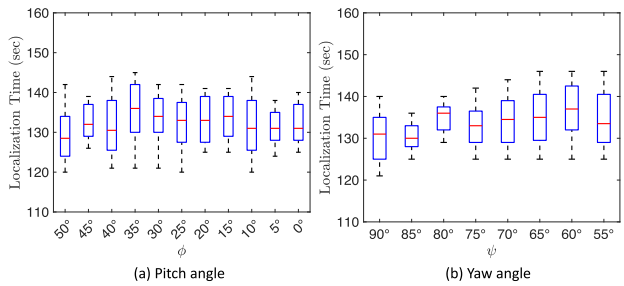


Fig. 13. Localization time versus pitch/yaw angle (ϕ/ψ).

below 12.8 cm regardless of the pitch angle. Meanwhile, for a fixed pitch angle, those three errors then become 4.4, 4.8, and 6.8 cm, and the location error is always less than 12.9 cm for all chosen yaw angles. Fig. 13 illustrates the resultant localization time. We observe that the median localization time ranges from 128.5 to 136 seconds for varying pitch angles, as shown in Fig. 13(a), while it ranges from 130 to 137 seconds for different yaw angles, as indicated in Fig. 13(b). These results demonstrate that *MotionCompass* is robust to variations in pitch and yaw angles.

Impact of Movement Speed: We change the user's movement speed v from 0.2 m/s to 1.0 m/s, with increments of 0.2. For each v, we perform 10 attempts of *MotionCompasss*. Fig. 14 illustrates the localization errors when the movement speed varies. We observe the localization error slightly increases with the value of v, demonstrating the robustness of *MotionCompass* to the speed variation. When $v = 0.2$ m/s, the mean localization error is 3.6 cm, while it increases to 10.7 cm for $v = 1.0$ m/s. This is because a higher speed would naturally result in a larger error in distance measurement. On the other hand, with a higher speed, the localization can be finished in a shorter time. Fig. 15 shows the relationship between the localization time and the movement speed. We observe that the median localization time

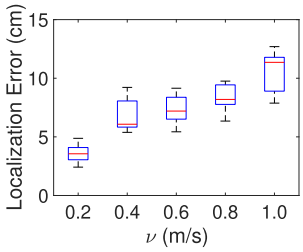


Fig. 14. Localization error versus v.

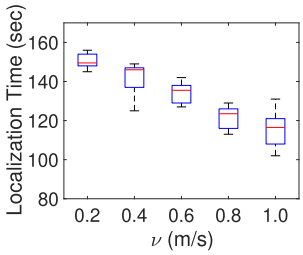


Fig. 15. Localization time versus v.

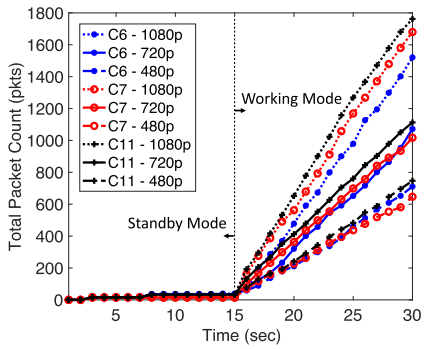


Fig. 16. Total packet count for cameras with different video resolutions.

equals 150 seconds when v is 0.2 m/s, and it drops to 117 seconds when v is increased to 1.0 m/s.

C. Varying Power Management Settings

Many battery-powered wireless cameras allow users to manually adjust the power management settings according to personal preference in terms of video quality or battery life, such as Blink [40], Blue by ADT [41], and Reolink [42]. Usually, three options are provided, including (i) *Best Video* for seeking the best video quality, (ii) *Best Battery Life or Saver* for extending the battery life, and (iii) *Optimized or Standard* for balance both video quality and battery balance. Take Blink cameras as an example. The above three options imply video resolutions of 1080p, 720p, and 480p, respectively [43].

Fig. 16 plots the captured 30-second traffic for three different cameras, i.e., C6 (Blink XT2), C7 (Blue by ADT), and C11 (Reolink Argus Pro) with varying video resolutions, where each camera maintains in standby mode until being triggered by continuous motion starting at the 16th second. We can see that in standby mode, the total count of packets generated by each camera increases slowly, as they only transmit heartbeat packets; in working mode (i.e., when motion is detected), the amount of packets generated by each camera jumps regardless of its video resolution. Also, we observe that the throughput (i.e., the rate of transmitted packets) increases with the video resolution for cameras. These results demonstrate that under varying video resolutions, the traffic patterns when the cameras are activated are consistently distinguishable from when they are in standby mode. For each video resolution of C6, C7, or C11, we further perform 25 trials of *MotionCompass*. Figs. 17 and 18 present the mean localization error and time. We see that different video resolutions for each camera achieve quite similar mean localization errors and time, indicating the robustness of *MotionCompass* against power management settings.

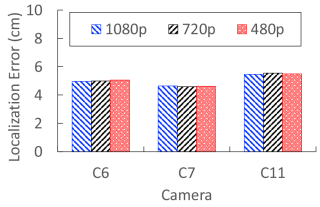


Fig. 17. Average localization error versus video resolution.

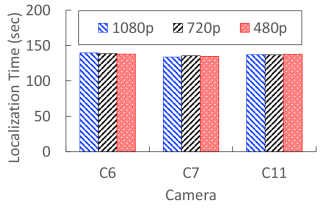


Fig. 18. Average localization time versus video resolution.

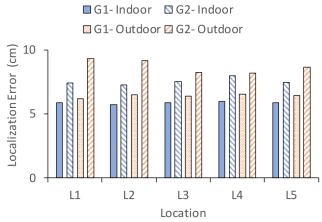


Fig. 19. Mean localization error.

D. Overall Localization Performance

We test all cameras in both environments. We perform 25 trials for each camera at every selected location, and thus have $18 \times 2 \times 5 \times 25 = 4,500$ attempts in total.

We compute the mean localization error and time for a camera in G1 (with one motion sensor) or G2 (with two motion sensors), as shown in Figs. 19 and 20. We see three tendencies. First, the performance is consistent across different locations in each environment. The mean localization error is always below 9.2 cm and the mean localization time stays less than 178 seconds. Second, in both indoor and outdoor environments, on average,

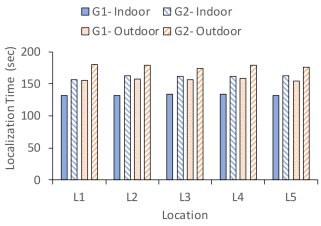


Fig. 20. Mean localization time.

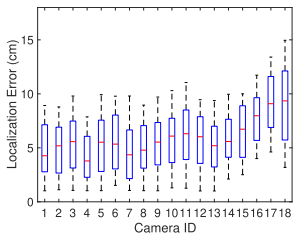


Fig. 21. Outdoor localization error.

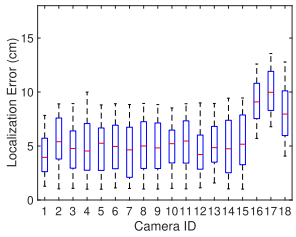


Fig. 22. Indoor localization error.

a camera in G2 causes a larger localization error and requires a longer localization time than a camera in G1. This is because a camera in G2 has a larger motion detection zone. The attacker thus has to walk longer to create the simulating motion, and also a larger localization error may be introduced. Finally, for each group of cameras, the mean localization error is larger and the mean localization time is longer in the outdoor environment compared with indoor. This appears due to the fact that the outdoor environment provides a wider space and the user may spend a longer time generating the simulating motion.

Figs. 21 and 22 show the localization errors for different cameras in the indoor and outdoor environments. We can see that for all cameras under both scenarios, a high localization accuracy can be always achieved. *MotionCompass* is able to achieve a minimum localization error ranging from 1.0 to 2.3 cm for cameras 1-15, while for cameras 16-18, the achieved minimum localization error varies from 3.1 to 4.5 cm. In the indoor environment, the localization error is slightly smaller than that in the outdoor environment overall. In both environments, cameras with two motion sensors cause slightly higher mean localization errors than cameras with one motion sensor.

Pinpointing Multiple Cameras: Note that *MotionCompass* monitors the wireless traffic based on MAC. We can thus simultaneously monitor multiple traffic flows, each of which

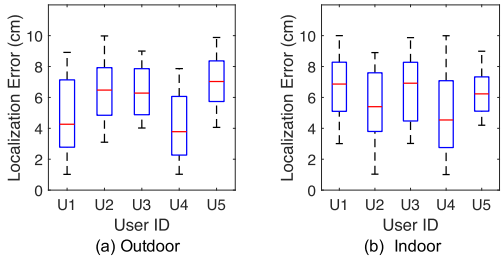


Fig. 23. Localization errors for different users.

TABLE II
LOCALIZATION TIME FOR DIFFERENT USERS

User ID	Localization time (seconds)		
	Average	Minimum	Maximum
U1	135	129	146
U2	141	130	147
U3	143	133	152
U4	137	128	145
U5	140	127	151

belongs to a wireless camera, and different cameras will not interfere with each other's localization.

E. User Study

We recruited 5 volunteers and asked each of them to perform *MotionCompass* to pinpoint a hidden wireless camera randomly selected and deployed in the aforementioned outdoor or indoor environment. Every participant performed 25 attempts for each environment. We make sure that the camera's field of view is not obstructed by the wall and it monitors an area that the participant can arrive at.

Fig. 23 shows the obtained localization errors. We can observe that the maximum localization error for each user is always below 10.0 cm, while for some users (e.g., user 2 in the indoor environment), they can achieve a localization error of as small as 1.0 cm. Meanwhile, in the outdoor environment, the mean localization error ranges from 3.8 to 7.0 cm for all users; and such a range becomes 4.5 to 6.9 cm in the indoor environment. These results demonstrate that the localization accuracy is quite consistent among different users. Table II presents the mean, minimum, and maximum localization time for different users. We also see a consistent average localization time for all users varying between 135 and 143 seconds. This verifies the practicality of the proposed camera localization strategy.

F. Localizing a Camera in Always-Active Mode

As discussed in Section IV-D4, occasionally, the motion sensor may be turned off and the camera is in the always-active mode, continuously monitoring the target area. When the camera supports VBR encoding, its generated traffic still has a correlation with the motion occurring in different areas of the camera's field of view. We can thus achieve pinpointing such cameras in always-active mode by leveraging the captured VBR video traffic. We select three different popular cameras for testing, including a Ring Stickup camera (with a 115-degree field of

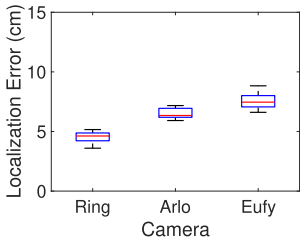


Fig. 24. Localization errors for cameras in always-active mode.

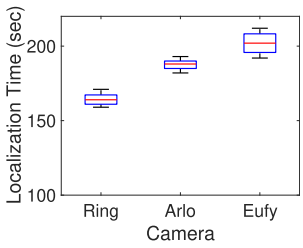


Fig. 25. Localization time for cameras in always-active mode.

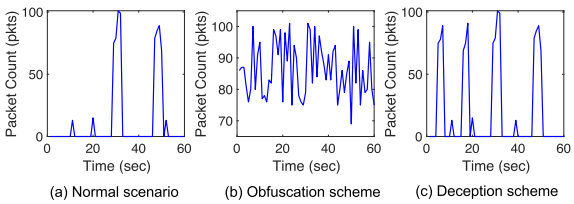


Fig. 26. Impact of traffic padding.

view), an Arlo Pro 2 camera (with a 130-degree field of view), and a Eufy 2 C camera (with a 135-degree field of view). For each camera, we perform 25 trials of the localization.

Fig. 24 presents the localization errors for three different cameras. We observe that the localization error is always less than 8.8 cm regardless of the camera type. Specifically, the mean localization errors for the three cameras are 4.6, 6.5, and 7.5 cm, respectively, demonstrating that the mean localization error slightly increases with the camera's field of view. Fig. 25 shows the corresponding localization time, which ranges from 159 to 212 seconds. With the camera's field of view increasing, the mean localization time slightly increases accordingly. The values of the mean location time for Ring Stickup, Arlo Pro 2, and Eufy 2 C are 164, 187, and 202 seconds, respectively. These results convincingly demonstrate that *MotionCompass* is able to pinpoint an always-active wireless camera supporting VBR encoding with a high localization accuracy and a short localization duration.

VI. Discussions

A. Limitations

The Requirements of WiFi and Motion Stimuli: MotionCompass sniffs in 802.11 (WiFi) networks and does not work for cameras using cellular connections. Also, it requires generating motion in the target area to activate the wireless camera. The attacker can walk in disguise or ask a helper to perform motion. Alternatively, she can utilize a moving robot/drone to introduce motion. Such methods, however, inevitably bring extra hardware costs.

Customized Activity Zone or Occlusion Effects: Our experiment is performed with cameras in default and recommended settings (i.e., with the maximum activity zone). In practice, some wireless cameras do not support activity zone customization (e.g., Canary Flex, Conico Cam, and AIVIO Cam), and some (e.g., Arlo Pro 2) allow users to create one or multiple activity zones under certain circumstances. For example, Arlo cameras allow users who subscribe to Arlo Smart plans or who connect the camera to continuous power to create up to 3 zones [44]. Additionally, there may be obstacles occluding the camera's vision; this is actually quite similar conceptually to activity zone customization as it involves reducing the space surveyed by the camera.

We discuss two scenarios. First, the owner only creates one activity zone: if it is slightly smaller compared with the default one (or if there is a small obstacle), *MotionCompass* still works with a small sacrifice in localization accuracy, as the attacker can estimate the default activity zone with the measured one; when the customized activity zone is too small (or there is a large obstacle blocking most of the zone), *MotionCompass* may fail, however, such a setting leaves a significant security risk as the camera can only alert motion within a small area. Second, the owner creates multiple activity zones: the attacker can determine all separated activity zones through stimulating motion and resultant wireless traffic, and then utilize such information to estimate the default activity zone and further pinpoint the camera.

Cameras Without Uploading Video: Some cameras, but not all, support local storage, enabling storing recordings in cameras or their base stations. *MotionCompass* still works in the latter case as there is still real-time traffic between the camera and the base station when the camera is activated. In the first case, cameras may not upload any videos, generating no real-time wireless traffic and thus leading to the failure of *MotionCompass*. However, such cameras may prevent their owners from receiving the recording notifications in time. Meanwhile, local storage (microSD card or USB drive) incurs extra costs and is vulnerable to theft/damage.

Wide-Angle Cameras: If a wide-angle camera covers the whole room, the relationship between the wireless traffic and the motion path cannot be obtained. Thus, *MotionCompass* fails. However, such a camera would be triggered frequently, and the battery may be depleted quite quickly.

B. Defense Strategies

MotionCompass explores the relationship between motion and wireless traffic to localize the camera. An intuitive solution is thus to disrupt the attacker from obtaining this relationship. The camera owner may manually turn off the motion sensor (e.g., [45]) so that the camera goes into standby mode and will not respond to any motion. However, this solution is

TABLE III
COMPARISON OF *MOTIONCOMPASS* AND EXISTING TECHNIQUES

Method	Equipment	Human Efforts	Localization Time
LM-8 Bug Detector [55]	special device	**High**: turn off all environmental wireless devices; check every corner	depending on the area size
SNOOPDOG [14]	a smartphone and a laptop	**High**: perform specific motion; walk around the perimeter of the room; stand, point a laptop in particular directions, and play videos	40 sec for detection and 30 sec per trial for localization
LAPD [56]	a smartphone equipped with a time-of-flight (ToF) sensor	**High**: stand at a short and ideal distance to the camera, with specific scan speeds; scan every corner	requiring to pre-select suspicious objects; depending on the area size
Lumos [57]	a smartphone or a tablet	**Moderate**: walk around the perimeter of the space	depending on the space size (under 30 minutes for a 1000 sq. ft. space)
MotionCompass	a smartphone	**Low**: walk along two or three paths	varying between 135 and 143 sec

impractical as it will make the camera lose the capability of timely sensing intrusion and sending an alert. Also, if the camera always maintains active, the battery will be drained quickly, and *MotionCompass* can still achieve the camera localization by leveraging the capture VBR video traffic, as demonstrated in Section IV-D4.

Recording Time Randomization: A practical defense is to randomize the recording length. As we cannot predict when the motion occurs, and need to make sure that the owner receives the alert in time, we cannot add extra recording or delay the recording at the beginning of the motion. Instead, we can continue to record for an extra random period once the motion stops, causing the attacker to obtain inaccurate localization results. This technique, however, will speed up the power consumption.

Uploading Time Randomization: Alternatively, we can postpone uploading motion-induced video. With this defense, the camera is still activated by any motion in the activation zone, while it only sends the alert and stores the recording locally. The motion detection capability is thus not affected. Meanwhile, the attacker only observes short wireless traffic for the introduced motion in the activation zone. As there is no longer a determined relationship between the motion trajectory and the resultant wireless traffic, *MotionCompass* fails. However, this defense requires the camera to be equipped with a large local memory.

Traffic Padding: Also, we can leverage traffic padding (i.e., increasing traffic volume) to defend against the proposed camera localization scheme. Previous research has shown that traffic padding can re-shape traffic patterns of IoT devices to prevent eavesdroppers from inferring user activity via traffic analysis [46], [47]. Specifically, we develop two strategies, *obfuscation* and *deception*. The idea behind the obfuscation is to make the disclosed traffic bursts featureless so that the traffic shows a consistent pattern over time regardless of whether stimuli are fed or not. Different from obfuscation, which makes the eavesdropper unable to perform device localization, the strategy of deception intentionally generates bogus traffic bursts (i.e., patterns) and thus may make the eavesdropper obtain inaccurate localization results. Fig. 26 presents the impact of padding the traffic of a motion-activated wireless camera (EufyCam 2 C). We see that in Fig. 26(a), the traffic burst happens with the motion appearing in the detection range of the camera with no traffic padding; in Fig. 26(b), packet counts at different time are similar, making it difficult for the eavesdropper to associate them with the fact of whether the camera detects motion. Also, in Fig. 26(c), the camera crafts extra traffic bursts that may mislead the eavesdropper to obtain a fake and incorrect camera's location. Traffic padding, however, will introduce extra bandwidth overhead.

VII. RELATED WORK

A. Traffic Analysis

Traffic analysis can achieve various applications, including detecting drones [48], [49], inferring apps [50], [51], monitoring misbehaving apps [52], enforcing network access restrictions [53], identifying actions on apps [54], and detecting hidden wireless cameras [11]. Our work also uses traffic analysis to determine whether the wireless camera is activated. Unlike existing traffic analysis based approaches (e.g., [11], [50], [51], [52]), which utilize the inherent traffic patterns to detect devices or apps which generate them, our work correlates the traffic pattern with human activities. Specifically, we exploit the association between the traffic variation and the location where the motion is introduced, and compute the camera's location by obtaining the edge information of the camera's motion detection range.

B. Hidden Wireless Camera Detection

There are emerging research efforts performing hidden wireless camera detection due to their popularity and the privacy concerns associated with unauthorized videotaping [9], [10], [11], [12], [58]. For example, [11] proposes a hidden wireless camera detection approach by utilizing the intrinsic traffic patterns of flows from wireless cameras; [10] investigates the responsive traffic variation corresponding to the light condition change to determine whether the traffic is produced by a wireless camera; [58] proposes to detect wireless cameras by monitoring network traffic that indicates the characteristics of corresponding audio transmission. All those traffic pattern based techniques, however, can only detect the existence of traffic flows belonging to wireless cameras, and they cannot tell the exact location of the camera. In contrast, our work not only detects wireless camera traffic but also pinpoints the location of the camera.

There are some other recent studies (e.g., [14], [56], [57]) that propose approaches to localize wireless cameras. Particularly, like how *MotionCompass* works in localizing cameras in always-active mode, [14] leverages the cause-effect relationship between dynamic scenes and the traffic of cameras supporting VBR. It further proposes a trial-based localization algorithm to pinpoint cameras, while *MotionCompass* achieves localization using spatial analysis. Meanwhile, the localization performance for [14] depends on the number of trials, and more trials would increase the time to finish the localization. The work [56] leverages the laser time-of-flight (ToF) depth sensors on commodity smartphones to localize hidden cameras. However, many current smartphones do not equip ToF sensors. Also, it requires the user first to determine a suspicious object that must be near (less than 1 m) to the user and scan every suspicious item within a room, which is time-consuming. As a result, it may not work if the camera is installed somewhere several meters away from the user (e.g., in the corner of walls). Nevertheless, such a laser-based method and the traffic analysis-based camera detection/localization approaches can be complementary from two perspectives. First, traffic analysis-based methods exclusively work for wireless cameras while the laser-based method [56] can function for cameras that do not emit wireless signals. Second, as the traffic analysis-based methods can either only detect the existence of cameras or suffer from errors in localizing cameras, they thus can be initially employed to narrow down the possible area where the camera may be located, and subsequently, the laser-based method [56] can be then applied to confirm the precise camera location. Table III illustrates the comparison between the proposed technique and other existing camera localization techniques.

VIII. Conclusion

We propose *MotionCompass*, a lightweight technique for pinpointing wireless cameras. Its novelty stems from identifying and proving that the motion activation property of wireless cameras may disclose the camera's location. By generating customized movement which stimulates the camera to emit wireless traffic, and correlating the motion trajectory with observed wireless traffic, *MotionCompass* can achieve robust camera localization. We also extend *MotionCompass* to handle the cases when the hidden camera is in always-active mode by analyzing the sniffed VBR video traffic. Extensive real-world experiments demonstrate that *MotionCompass* can achieve an average localization error of about 5 cm in both indoor and outdoor environments.

References

[1] Y. He, Q. He, S. Fang, and Y. Liu, "MotionCompass: Pinpointing wireless camera via motion-activated traffic," in *Proc. 19th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 215–227.

[2] Market Research Future, "Global wireless monitoring and surveillance market," 2020. [Online]. Available: https://www.marketresearchfuture.com/reports/wireless-monitoring-surveillance-market-975

[3] G. Fleishman, *Take Control of Home Security Cameras*. San Diego, CA, USA: Take Control Books, 2020.

[4] K. Iboshi, "We asked 86 burglars how they broke into homes," 2017. [Online]. Available: https://www.ktvb.com/article/news/crime/we-asked-86-burglars-how-they-broke-into-homes/277--344333696

[5] A. Li, "Security camera blind spots: How to find and avoid them," Nov. 2018. [Online]. Available: https://reolink.com/find-and-avoid-security-camera-blind-spots/

[6] Y. Ye, S. Ci, A. K. Katsaggelos, Y. Liu, and Y. Qian, "Wireless video surveillance: A survey," *IEEE Access*, vol. 1, pp. 646–660, 2013.

[7] S. Mare, F. Roesner, and T. Kohno, "Smart devices in AirBnBs: Considering privacy and security for both guests and hosts," in *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 2, pp. 436–458, 2020.

[8] IPX1031, "Survey: Do AirBnB guests trust their hosts?," Apr. 2019. [Online]. Available: https://www.ipx1031.com/airbnb-guests-trust-hosts/

[9] Y. Cheng, X. Ji, T. Lu, and W. Xu, "DeWiCam: Detecting hidden wireless cameras via smartphones," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 1–13.

[10] T. Liu, Z. Liu, J. Huang, R. Tan, and Z. Tan, "Detecting wireless spy cameras via stimulating and probing," in *Proc. 16th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2018, pp. 243–255.

[11] Y. Cheng, X. Ji, T. Lu, and W. Xu, "On detecting hidden wireless cameras: A traffic pattern-based approach," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 907–921, Apr. 2020.

[12] K. Wu and B. Lagesse, "Do you see what I see? Detecting hidden streaming cameras through similarity of simultaneous observation," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2019, pp. 1–10.

[13] N. Lakshmanan, I. Bang, M. S. Kang, J. Han, and J. T. Lee, "SurFi: Detecting surveillance camera looping attacks with Wi-Fi channel state information," in *Proc. 12th Conf. Secur. Privacy Wireless Mobile Netw.*, 2019, pp. 239–244.

[14] A. D. Singh, L. Garcia, J. Noor, and M. Srivastava, "I always feel like somebody's sensing me! A framework to detect, identify, and localize clandestine wireless sensors," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1829–1846.

[15] X. Ji, Y. Cheng, W. Xu, and X. Zhou, "User presence inference via encrypted traffic of wireless camera in smart homes," *Secur. Commun. Netw.*, vol. 2018, pp. 1–10, Sep. 2018.

[16] Blink XT2 system bundles, 2020. [Online]. Available: https://blinkforhome.com/collections/blink-xt2-outdoor-cameras

[17] Arlo Pro 2, 2020. [Online]. Available: https://www.arlo.com/en-us/products/arlo-pro-2/default.aspx

[18] K. Scarfone, D. Dicoi, M. Sexton, and C. Tibbs, "Guide to securing legacy IEEE 802.11 wireless networks," *Nat. Inst. Standards Technol. Special Publication*, vol. 800, 2008.

[19] J. Martin, E. Rye, and R. Beverly, "Decomposition of MAC address structure for granular device inference," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, 2016, pp. 78–88.

[20] H. Liu, Y. Wang, K. Wang, and H. Lin, "Turning a pyroelectric infrared motion sensor into a high-accuracy presence detector by using a narrow semi-transparent chopper," *Appl. Phys. Lett.*, vol. 111, no. 24, 2017, Art. no. 243901.

[21] S. Narayana, R. V. Prasad, V. S. Rao, T. V. Prabhakar, S. S. Kowshik, and M. S. Iyer, "PIR sensors: Characterization and novel localization technique," in *Proc. 14th Int. Conf. Inf. Process. Sensor Netw.*, 2015, pp. 142–153.

[22] T. S. Saponas et al., "Devices that tell on you: Privacy trends in consumer ubiquitous computing," in *Proc. USENIX Secur. Symp.*, 2007, pp. 55–70.

[23] Y. Liu, A.-R. Sadeghi, D. Ghosal, and B. Mukherjee, "Video streaming forensic–content identification with traffic snooping," in *Proc. Int. Conf. Inf. Secur.*, 2010, pp. 129–135.

[24] J. Gu, J. Wang, Z. Yu, and K. Shen, "Walls have ears: Traffic-based side-channel attack in video streaming," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1538–1546.

[25] Smart WiFi motion sensor, 2022. [Online]. Available: https://www.bazzsmarthome.com/products/smart-wifi-motion-sensor

[26] Arlo Technologies, Inc., "Arlo Pro 2 HD security camera system user manual," Feb. 2019. [Online]. Available: https://www.arlo.com/en-us/images/Documents/ArloPro2/arlo_pro_2_um.pdf

[27] IEEE, "OUI public listing," 2020. [Online]. Available: http://standards-oui.ieee.org/oui/oui.txt

[28] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The IEEE 802.11 universe," *IEEE Commun. Mag.*, vol. 48, no. 1, pp. 62–70, Jan. 2010.

[29] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, "IEEE 802.11 wireless local area networks," *IEEE Commun. Mag.*, vol. 35, no. 9, pp. 116–126, Sep. 1997.

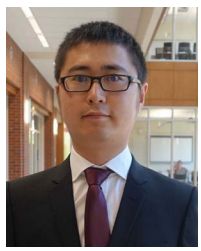
[30] Aircrack-ng, "Main documentation–aircrack-ng suite," 2022. [Online]. Available: https://www.aircrack-ng.org/documentation.html

[31] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 41–49, Sep. 2018.

[32] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2016, pp. 413–424.
[33] M. Bagnulo, P. Matthews, and I. van Beijnum, "RFC 6146: Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers," *RFC Editor*, USA, 2011.
[34] J. Martin et al., "A study of MAC address randomization in mobile devices and when it fails," in *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 365–383, 2017.
[35] E. Fenske, D. Brown, J. Martin, T. Mayberry, P. Ryan, and E. C. Rye, "Three years later: A study of MAC address randomization in mobile devices and when it succeeds," in *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 3, pp. 164–181, 2021.
[36] Arlo Ultra: 4K security camera: 4K wireless camera system, 2020. [Online]. Available: https://www.arlo.com/en-us/products/arlo-ultra/default.aspx
[37] What is variable bit rate? 2022. [Online]. Available: https://www.adobe.com/creativecloud/video/hub/guides/what-is-variable-bit-rate.html
[38] Motion detection, 2022. [Online]. Available: https://support.simplisafe.com/hc/en-us/articles/360035333332-Motion-Detection
[39] Offensive Security, "Kali Linux NetHunter," 2022. [Online]. Available: https://www.kali.org/kali-linux-nethunter/
[40] Camera video quality, 2023. [Online]. Available: https://support.blinkforhome.com/en_US/issues-with-your-camera/camera-video-quality
[41] Blue by ADT wireless outdoor camera," 2023. [Online]. Available: https://support.bluebyadt.com/s/article/Blue-by-ADT-Outdoor-Cameras
[42] How to set up the quality settings via reolink app, 2023. [Online]. Available: https://support.reolink.com/hc/en-us/articles/360006937654/
[43] Best settings for blink outdoor camera, 2023. [Online]. Available: https://smarthomeways.com/best-settings-for-blink-outdoor-camera/
[44] What are activity zones and how do I create them? 2020. [Online]. Available: https://kb.arlo.com/1001908/What-are-activity-zones-and-how-do-I-create-them
[45] Reolink, "How to turn on/off the PIR sensor," 2020. [Online]. Available: https://support.reolink.com/hc/en-us/articles/360004379493-Turn-on-off-the-PIR-sensor
[46] T. Datta, N. Apthorpe, and N. Feamster, "A developer-friendly library for smart home IoT privacy-preserving traffic obfuscation," in *Proc. Workshop IoT Secur. Privacy*, 2018, pp. 43–48.
[47] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) IoT traffic shaping," in *Proc. Privacy Enhancing Technol.*, vol. 3, pp. 128–148, 2019.
[48] P. Nguyen, H. Truong, M. Ravindranathan, A. Nguyen, R. Han, and T. Vu, "Matthan: Drone presence detection by identifying physical signatures in the drone's RF communication," in *Proc. 15th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 211–224.
[49] S. Sciancalepore, O. A. Ibrahim, G. Oligeri, and R. Di Pietro, "PiNcH: An effective, efficient, and robust solution to drone detection via network traffic analysis," *Comput. Netw.*, vol. 168, 2020, Art. no. 107044.
[50] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2015, pp. 433–441.
[51] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 1, pp. 63–78, Jan. 2018.
[52] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring smart home apps from encrypted traffic," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 1074–1088.
[53] I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-keeper: Detecting malicious IoT network activity using online traffic analysis at the edge," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 45–59, Mar. 2020.
[54] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Can't you hear me knocking: Identification of user actions on Android apps via traffic analysis," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy*, 2015, pp. 297–304.
[55] Lm-8 hidden camera & bug detector, 2023. [Online]. Available: https://www.spyguy.com/products/lm-8-hidden-camera-bug-detector
[56] S. Sami, S. R. X. Tan, B. Sun, and J. Han, "LAPD: Hidden spy camera detection using smartphone time-of-flight sensors," in *Proc. 19th ACM Conf. Embedded Netw. Sensor Syst.*, 2021, pp. 288–301.
[57] R. A. Sharma, E. Soltanaghaei, A. Rowe, and V. Sekar, "Lumos: Identifying and localizing diverse hidden IoT devices in an unfamiliar environment," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1095–1112.
[58] R. Mitev, A. Pazii, M. Miettinen, W. Enck, and A.-R. Sadeghi, "LeakyPick: IoT audio spy detector," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2020, pp. 694–705.

Yan He received the BS and MS degrees both from the University of Oklahoma, in 2019 and 2022, respectively. He is currently working toward the PhD degree in computer science with the University of Oklahoma. His research interests include mobile computing and Internet-of-Things (IoT) security.

Qiuye He received the MS degree from Xidian University, Xi'an, China, in 2019. She is currently working toward the PhD degree in computer science with the University of Oklahoma. Her research interests are in the area of wireless security and mobile computing.



Song Fang received the PhD degree in computer science from the University of South Florida, in 2018. He is now an assistant professor with the School of Computer Science, University of Oklahoma. His research interests include wireless and mobile system security, cyber-physical systems and IoT security, and mobile computing. He is also interested in applying machine learning in cybersecurity. He received the NSF CISE CRII award, in 2020.

Yao Liu received the PhD degree in computer science from North Carolina State University, in 2012. She is now an associate professor with the Department of Computer Science and Engineering, University of South Florida. Her research is related to computer and network security, with an emphasis on designing and implementing defense approaches that protect mobile, network and computer technologies from being undermined by adversaries. Her research interests also lie in the security applications for cyber-physical systems, Internet of Things, and machine learning. She was an NSF CAREER Award recipient, in 2016. She also received the ACM CCS Test-of-Time Award by ACM SIGSAC, in 2019.