UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

SECURING SMART HOME IOT APPLICATIONS VIA WIRELESS TRAFFIC ANALYSIS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

Yan He
Norman, Oklahoma
2022

SECURING SMART HOME IOT APPLICATIONS VIA WIRELESS TRAFFIC ANALYSIS

A THESIS DEFENSE APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Fang Song, Chair

Dr. Anindya Maiti

Dr. Lan Chao

© Copyright by Yan He 2022
All Rights Reserved.

ACKNOWLEDGEMENTS

First, thank my academic advisor Dr. Fang Song, for his professional advice, patience, and supervision throughout my master's studies. Also, I would like to appreciate my dissertation committee members: Dr. Anindya Maiti and Dr. Lan Chao. I improve my thesis step by step from their valuable feedback and guidance.

I am expressing a special thanks to the National Science Foundation (NSF) and the school of computer science for their financial support for my studies.

Finally, thanks to all my family that supported me these years. Even though they are not here with me now, I still feel warm and supported by them all the time. COVID changes a lot and makes traveling harder. I am so happy that I have a strong and lovely family.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Householders have widely used IoT security systems with the development of smart home applications. Wireless security cameras are integral components of IoT security systems used by many private homes. These cameras commonly employ motion sensors to identify something occurring in their fields of vision before recording and notifying the property owner of the activity. In this thesis, we discover that the motion-sensing action can disclose the camera's location through a novel wireless camera localization technique we call *MotionCompass*. In short, a user who aims to avoid surveillance can find a hidden camera by creating motion stimuli and sniffing wireless traffic for a response to that stimulus. With the motion trajectories within the motion detection zone, the user can then compute the camera's exact location. We develop an Android app to implement *MotionCompass*. Our extensive experiments using the developed app and 18 popular wireless security cameras demonstrate that *MotionCompass* can attain a mean localization error of around 5 cm in less than 140 seconds for cameras with one motion sensor. This localization technique builds upon existing work that detects the existence of hidden cameras to pinpoint their exact location and area of surveillance.

CHAPTER I
INTRODUCTION TO REALWORLD ISSUES

1.1 AN OVERVIEW OF WIFI ENABLED IOT DEVICES

Many smart home applications are based on IoT (Internet-of-Things) devices. This helps the user execute certain tasks by controlling various household devices like lighting, air conditioning (AC), and garage door, or monitoring the home via camera or motion sensors. These network-enabled devices can receive and execute the command in seconds and report their working status. There are three main connection protocols widely used by IoT devices:

WiFi:

WiFi is an IEEE 802.11 b/g/n-based connection for IoT devices to establish a network connection with their controller/server.

Bluetooth:

Bluetooth is used by some IoT devices to communicate with their hub or user's smartphone.

Zigbee:

Zigbee hubs are widely used by IoT devices to handle the connection between the wireless sensors and their hub.

Due to the open nature of the wireless medium, wireless communications are especially vulnerable to eavesdropping and jamming attacks [45,46,47]. A passive eavesdropper usually intercepts packets that are transmitted between two legitimate devices. An active jammer often

uses a radio frequency (RF) device to transmit wireless signals. As the signals of the jammer and the sender collide at the receiver side, the signal reception process is disrupted.

An active jammer often uses a radio frequency (RF) device to transmit wireless signals. As the signals of the jammer and the sender collide at the receiver side, the signal reception process is disrupted [48]. For example, it's possible to use wireless signals to infer the keystrokes when you are typing [43,49,51]. It's also possible for an attacker to inject wireless channel characteristics into the wireless channels to hide the location [44,50]. Wireless connection method like WiFi protocols is widely used in IoT devices, and they are the devices purchased mainly by the users because they do not require any home hub and extra devices like smart controller to get installed. For example, an IoT light bulb can connect to WiFi, so you can turn on or off your light using your smartphone. An IoT thermostat allows you to access the central AC system of the home a thousand miles away. The IoT home wireless camera can help to prevent break-ins and burglars. The users can control as many IoT devices as they need to enhance the smart home experience, as the price of IoT devices is decreasing year to year. Those IoT devices may also bring security and privacy concerns. With passive traffic analysis, an adversary may steal sensitive information from connected IoT devices.

There are some existing works to utilize local network traffic analysis, which can detect IoT devices that are connected to WiFi networks [14]. The technique of traffic analysis can run packet-level analysis on captured local network traffic. If there is a target found that can be proven to be an IoT device, the working pattern and packet information can be extracted from the traffic.

A quick and simple example from the smart switch, we know it only contains two statuses: ON and OFF. It does not send any wireless traffic other than turning it on or off. Thus, we can easily see whether the switch is operated by someone by monitoring the traffic originating from it.

Similarly, we can apply a similar approach to other IoT devices. Obviously, the most malicious and attractive target is the camera. Wireless security cameras widely use WiFi connections. Also, they are typically battery-powered. A certain mechanism is applied, i.e., a wireless camera is in deep sleeping mode to save power, and it then uses sensors like motion sensors to detect human motion. Once motion is detected, the camera is then activated. This type of activation brings a chance for us to utilize this mechanism and correlate the camera's behavior with the network traffic generated by the camera. Intuitively, we know that the traffic represents the existence of a wireless camera.

The security camera market is growing rapidly. Wireless cameras protect and monitor homes by capturing images from trespassers. The aim of this thesis is to discover the possibility of finding and even localizing the hidden wireless cameras using a novel method of network traffic analysis, which can then be used to enhance the existing camera design to protect the privacy of the residents.

1.2 PROBLEMS OF WIRELESS CAMERAS

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 disarm it.

A great portion of wireless cameras are equipped with built-in motion sensors, such as the best-selling ones – Amazon Blink XT2 [3] and Arlo Pro 2 [1]. Because of the volume of data collected by wireless cameras, 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, as shown in Figure 1. 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 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.

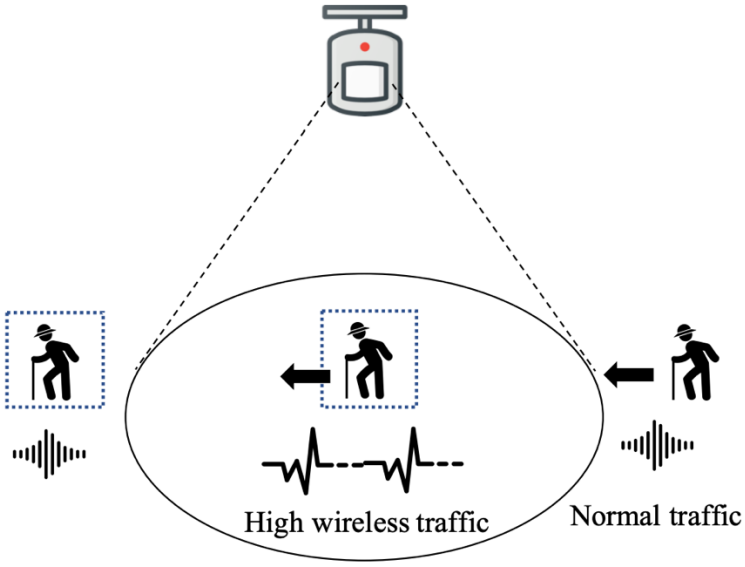


Figure 1: Motion-activated wireless camera.

In practice, there exist other different types of wireless traffic flows generated by non-camera devices, such as laptops, smartphones, or tablets. Thus, it is challenging to distinguish which traffic flows belong to wireless cameras, especially considering that the wireless local area networks (WLANs) employ encryption techniques such as WEP, WPA, and WPA2 to prevent information compromise from casual eavesdropping [34].

Almost all current commodity network devices deploy 802.11 wireless protocols, in which an inherent weakness has been found, i.e., exposure of link-layer Media Access Control (MAC) addresses [28]. Each MAC address, which is a persistent globally unique identifier, discloses the

device manufacturer information via the first three most significant MAC bytes, i.e., the Organizationally Unique Identifier (OUI). This thus motivates us to utilize OUI of MAC addresses to detect the existence of wireless camera traffic. Specifically, as OUI relies on the camera manufacturers and the main-stream manufacturers of wireless cameras are limited [10], 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, another challenge is how to design the motion stimulation so that the camera location can be found. To address this problem, we correlate the range of arbitrarily generated motion with the camera location. We design novel strategies to first set up a coordinate system and then compute the camera's coordinates for determining its location.

1.3 A NEW TECHNIQUE FOR WIRELESS CAMERA LOCALIZATION

As an example of an IoT product that is directly related to the home security. Wireless security cameras are integral components of security systems used by military installations, corporations, and, due to their increased affordability, many private homes. These cameras commonly employ motion sensors to identify that something is occurring in their fields of vision before starting to record and notify 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*. In short, a user who aims to avoid surveillance can find a hidden camera by creating motion stimuli and sniffing wireless traffic for a response to that stimulus. With the motion trajectories within the motion detection zone, the exact location of the camera can be then computed. We develop an Android app to implement *MotionCompass*. Our extensive experiments using the developed app and 18 popular wireless security cameras demonstrate that for cameras with one motion sensor. *MotionCompass* can attain a mean localization error of around 5 cm in less than 140 seconds. This localization technique builds upon existing work that detects the existence of hidden cameras, to pinpoint their exact location and area of surveillance.

In summary, the major contributions of this thesis are 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 connection to the same network as 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 implement MotionCompass and develop an Android application for validating its effectiveness and efficiency.*

CHAPTER II
LITERATURE REVIEW

Wireless Security Cameras: Growing awareness of safety and security has boosted the growth of the wireless camera markets [6]. Generally, wireless cameras process the video/audio streams (e.g., compressing them to a smaller size that facilitates transmission) and then upload them through a WLAN to a cloud backend server. Data transmission from the camera to the base station or router is regulated by wireless protocols.

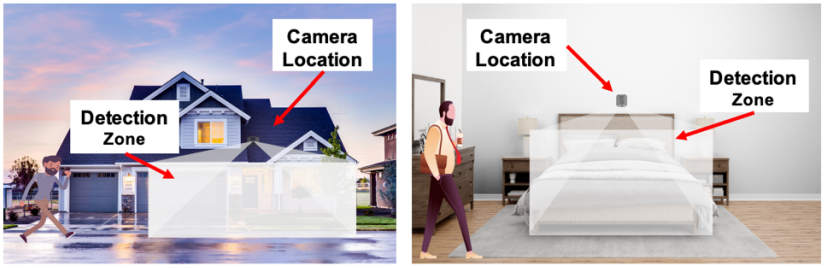
Wireless security cameras are usually equipped with motion or sound sensors for increased security, so that once motion or sound is detected, the camera turns on and starts recording, uploading captured video to the cloud backend server, and sending a notification to the property owner. Table 1 lists some technical parameters of popular wireless cameras. Sound-triggered systems often suffer from high false alarms via barking dogs, loud cars or other random noise. In this paper, we focus on localizing motion-activated wireless cameras, which have been widely adopted for security surveillance.

Table 1: Motion sensor equipped wireless cameras.

Model	Chipset	Alert	Protocol	Hub
AIVIO Cam	MediaTek	M*	WiFi	Yes
Arlo Essential	Broadcom	M/S**	WiFi/B+	No
Arlo Go	Broadcom	M	3G/4G-LTE	No
Arlo Pro 2	Broadcom	M/S	WiFi/B	Yes
Arlo Pro 3	Broadcom	M/S	WiFi/B	Yes
Arlo Ultra	Broadcom	M/S	WiFi/B	Yes
Blink Indoor	MediaTek	M/S	WiFi	No
Blink XT2	TI	M/S	WiFi	Yes
Blue by ADT	Broadcom	M	WiFi	No
Canary Flex	TI	M	WiFi	No
Conico Cam	TI	M	WiFi	No
EufyCam 2C	MediaTek	M/S	WiFi	Yes
Reolink Argus 2	MediaTek	M	WiFi	No
Reolink Argus Pro	MediaTek	M	WiFi	No
Reolink Go	TI	M	3G/4G-LTE	No
Ring Door View	TI	M	WiFi	No
Ring Spotlight	TI	M	WiFi	No
Ring Stickup Cam	TI	M	WiFi	No
Simplisafe Cam	Telit	M	WiFi	No
Swann Wireless	MediaTek	M	WiFi	No

* M: Motion ** S: Sound + B: Bluetooth

Motion Sensors: Usually, we cannot keep our eyes glued to our security camera's feed on phone or computer, especially when we have multiple cameras. To get rid of this limitation, a wireless security camera, incorporating a motion sensor, provides a practical solution. Figure 2 shows examples of camera detection zones for outdoor and indoor environments.



(a) Camera outside a house. (b) Camera in a room.

Figure 2: Camera detection zone examples.

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 [23] 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 [30]. Due to their properties of small size, low cost, power efficiency and being able to work in dark environments, PIR sensors are widely used in battery-powered wireless cameras. Without loss of generality, in this paper, we explore the localization of wireless cameras equipped with this type of motion sensors.

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. In the latest study conducted by Market Research Future in 2020, 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 [26].

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 [13]; (2) burglars may become more interested in breaking in as they think the camera signals that there are valuables inside the property [16]; (3) it is easier to avoid being recorded, e.g., an adversary may find the blind spots (i.e., areas not within the peripheral vision of the camera) and leverage them to evade being recorded [21]. 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 [25, 41], especially considering the progressively smaller size of spy cameras. These cameras can be kept hidden easily such that their targets are unaware of their existence. For example, 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 [18]. As thus, detection of wireless cameras is drawing increasing attention for privacy protection [9, 10, 20, 24, 39].

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 may work when the camera is actively transmitting, but existing work (e.g., [19]) can only detect the existence of wireless cameras but cannot tell their exact locations.

CHAPTER 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. With all information collected, we are able to obtain everything we need to calculate the camera's location in next steps.

CHAPTER IV
CAMERA LOCALIZATION

4.1 OVERVIEW

MotionCompass includes three important phases, i.e., camera traffic finder, camera MAC extraction, and camera traffic manipulation, as shown in Figure 3.

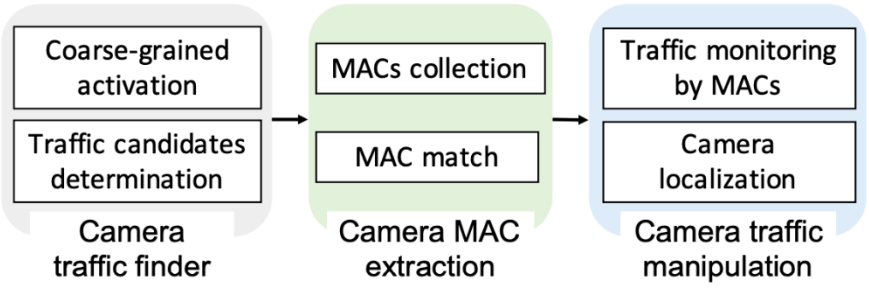


Figure 3: Three phases of the proposed scheme.

The first phase determines the 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 [4]), 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 wireless camera generates. We show the details for each

phase in the following discussion.

4.2 CAMERA TRAFFIC FINDER

Since there are numerous wireless traffic flows in the air, we need to shrink the candidates for wireless camera traffic flow. We propose to generate motion in the target area to stimulate potential cameras monitoring the area, and then utilize the resultant wireless traffic to determine candidate traffic flows that may be generated by a wireless security camera.

4.2.1 Coarse-grained Activation.

Wireless cameras are usually battery-powered. They sit in standby mode to conserve battery and begin to record when they detect motion or sound (i.e., activation signals). These videos are then sent to the cloud backend server for secure storage in owners' library. Also, when the video recording is activated, the camera can send the owner push notifications or email alerts.

Therefore, in order to observe the wireless traffic of the camera (i.e., enable the camera to send packets), an attacker can manually generate the activation signals in a target area. As a result, when the motion happens to be performed in the motion activity zone, the camera recording will be then triggered [8]. As shown in Figure 4, when the camera (Amazon Blink XT2) lies in standby mode, only a "heartbeat" signal of small size is sent out to the camera base station or the router at a regular interval in the order of seconds, indicating normal

operation or to synchronize with the other party; while if human activity is detected, an abnormally high wireless traffic would be generated accordingly.

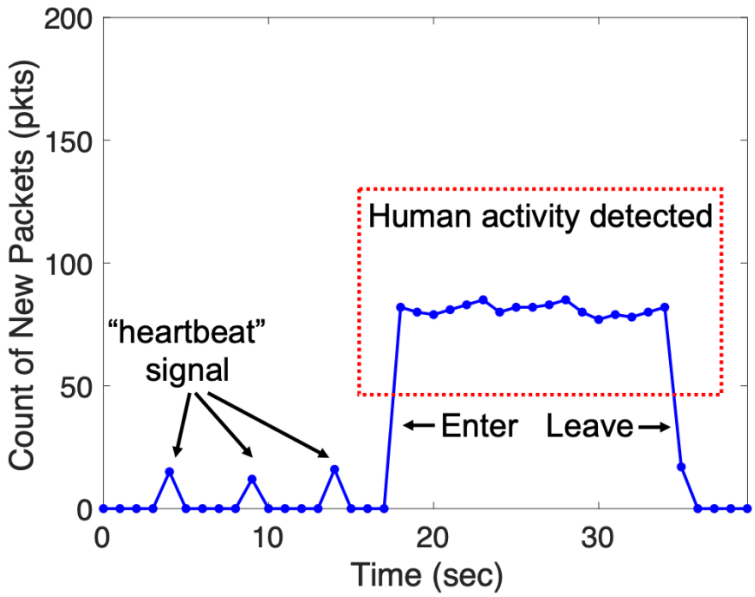


Figure 4: Count of newly captured packets when a user walks past the detection zone of the camera.

4.2.2 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. As a result, the traffic generated by the wireless camera 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 the 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 then determined as a candidate for the camera traffic.

4.3 CAMERA MAC EXTRACTION

Wireless cameras are powered by systems-on-a-chip (SoCs) from a few SoC manufacturers such as Broadcom, Ambarella and Texas Instruments (TI). SoCs typically bundle processors, memories, wireless networking capabilities, as well as power management circuits. As a result, the link-layer Media Access Control (MAC) address of a wireless camera is determined by corresponding SoC manufacturer and has the following format: for any six-byte MAC address, the first half is the Organizationally Unique Identifier (OUI) [17], indicating the device manufacturer; and the second half represents the unique device ID. MAC address intends to be permanent and globally unique identification. 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., [17]) defined by SoC suppliers to determine whether the monitored traffic belongs to a wireless security camera.

4.3.1 Collection of MAC Identifiers.

IEEE 802.11 wireless protocols are utilized in almost all commodity network devices [15], including wireless cameras. The use of 802.11, however, may cause exposure of MAC addresses [28], and a wireless eavesdropper within radio range of a wireless transmitter can capture the corresponding MAC address. Though the data packets generated by the wireless camera are encrypted [12], 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, as shown in Figure 5.

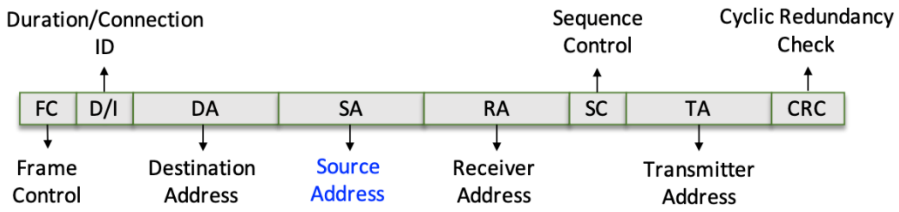


Figure 5: Header format of IEEE 802.11 MAC frame.

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 [7] 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.

4.3.2 Camera MAC Match.

As aforementioned, the 3-byte prefix (i.e., OUI) of a MAC is normally fixed and depends

on the manufacturer and the type of device. For example, Amazon Blink wireless cameras use HongHai's chipset for WiFi communication, and the OUI of their MACs starts with "1C-BF-CE", where "1C" denotes the HongHai company and "BF-CE" indicates the camera department. The uniqueness of OUI motivates us to firstly build a library containing OUIs of all cameras on the market, and then utilize it to determine whether the monitored traffic belongs to a wireless camera. We refer to such a table as camera labeled OUI table. Specifically, 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: Though manufacturers assign MAC addresses using the global standard, it may not be the case that devices will broadcast the OUI-based MAC that is originally flashed into the devices. Some devices may enable the user to change the MAC arbitrarily in software [37,40]. 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. However, recent studies have proposed techniques using a unique identifier called the Universally Unique Identifier-Enrollee (UUID-E), which can successfully recover the device's original, global MAC from the spoofed or randomized MAC [27,28,37]. Combining with such techniques to recover real MAC (if MAC spoofing occurs), the proposed OUI-based traffic analysis still works.

Besides, we can utilize another solution to handle MAC spoofing. 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), the resultant traffic patterns are similar as well [10]. This observation motivates us to first train a Support Vector Machine (SVM) model to classify traffic patterns, and then 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.

Figure 6 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.

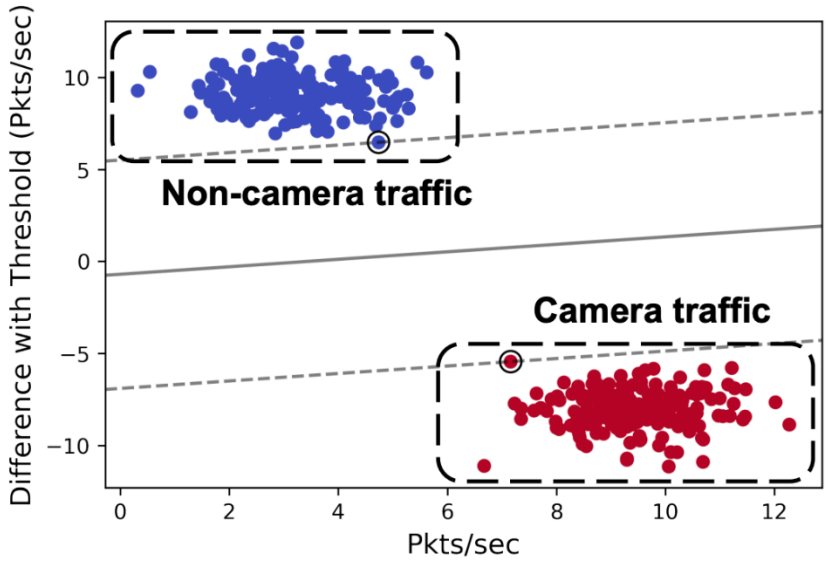


Figure 6: SVM training result for wireless traffic flows.

4.4 CAMERA TRAFFIC MANIPULATION

Camera traffic manipulation aims to shrink the possible candidates of the target camera determined in the previous phase into one and pinpoint its exact location. We begin by setting up a listener 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 environmental motion. We then build a model to correlate the camera location with the motion pattern, which directly affects the traffic generated by the camera.

4.4.1 Network Traffic Monitoring by MACs.

By setting up a packet monitor with existing tools, we can listen to the traffic coming from candidate cameras. Specifically, we detect if the traffic volume is changed and record the change of packet count.

If we purposely introduce human activity in a selected area, where a motion-activated camera happens to monitor, the camera will become awakened and generate traffic volume corresponding to the time that the manipulated activity lasts. On the other hand, if the monitored traffic has no change, we can determine that the candidate camera is not monitoring the area where the activity is performed. 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.

4.4.2 Camera Localization.

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).

Figure 7 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. For example, when the human activity within the detection zone lasts for 1 second, the packet counts for Blink XT2 camera and Arlo Pro 2 camera are 100 and 155 respectively,

and for every 2 seconds, the corresponding packet counts for the two cameras increase by an average of 80 and 197 (i.e., network throughput maintains almost constant).

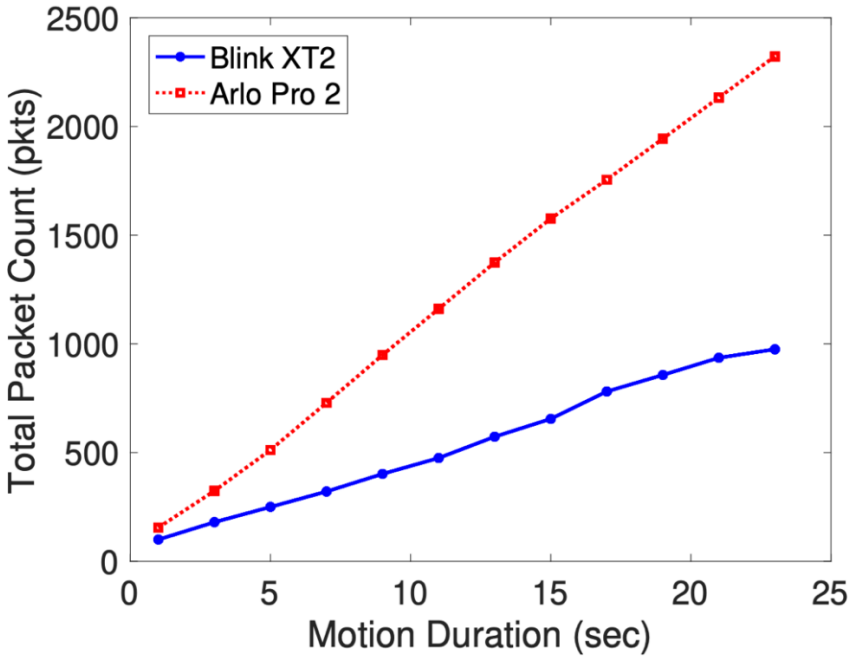


Figure 7: Total packet count vs. motion duration.

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

$$N = a + R \cdot \Delta t, \quad (1)$$

where a is a constant, R represents the throughput (i.e., the rate at which the activated camera generates packets), and Δt denotes exposure time.

With the linear model, we can determine whether the user performing the specified motion is still in the detection range of the motion sensor. Specifically, if the observed total packet count does not fit the linear model with a significant deviation, the performed motion at this time will be determined as out of the detection range of the motion sensor.

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, Arlo Ultra camera has dual PIR sensors and has a horizontal angle of view of 150° [2].

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.

4.4.3 Coordinates Calculation:

Special Case. In order 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 Figure 8, a user can perform motion along two paths with an average speed of υ and simultaneously monitor the wireless traffic, including two steps:

1. Moving parallel to the wall from left to right (or in the opposite way), as shown in Figure 8(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 the walking distance s within the detection range, i.e., $s = |AB| = v \cdot (t_2 - t_1)$.

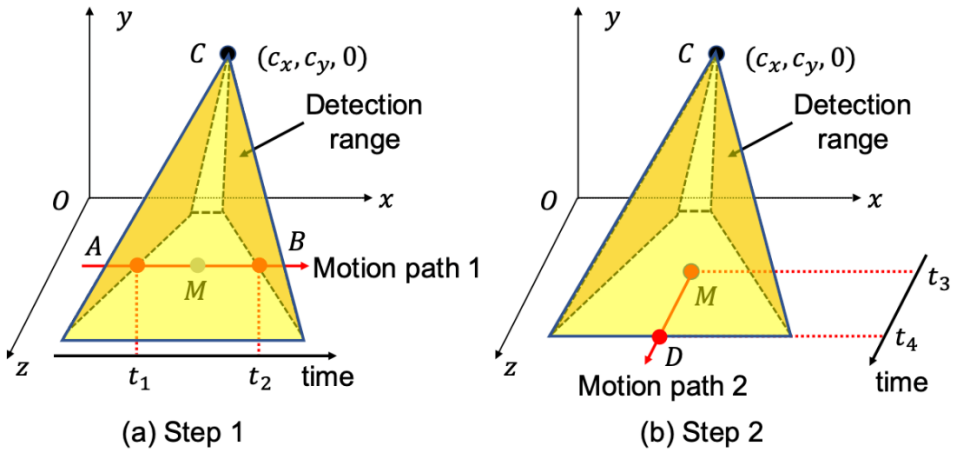


Figure 8: Two-step procedure for the special case.

2. Vertically getting out of the detection range with the start location at the midpoint M of the line segment $\overline{AB}$, as shown in Figure 8(b): when it is determined that the user leaves the detection range, the location is marked as D; similarly, the start time and the time the user leaves the detection range are recorded as t_3 and t_4, and the new walking distance s' within the detection range can be computed as $s' = |MD| = v \cdot (t_4 - t_3)$.

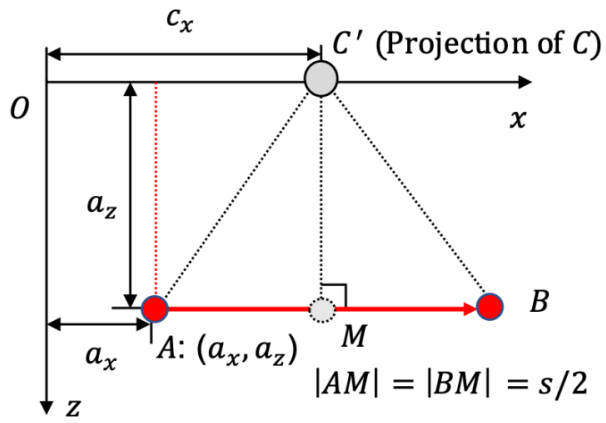


Figure 9: Calculation of c_x for the special case.

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 Figure 9, 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 Figure 10.

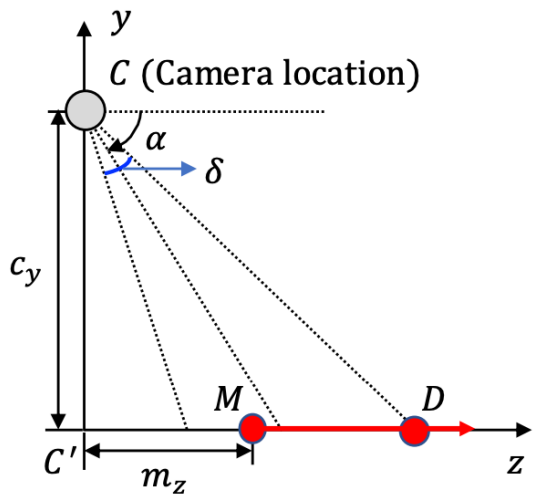


Figure 10: Calculation of c_y for the special case.

A general rule of thumb is to install the security 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 angle of the camera. The camera optical axis divides δ into two equal angles. With step 1, we can calculate the z-coordinate m_z of point m, which is equal to a_z. Meanwhile, we utilize γ to denote angle $\angle DCC'$. We thus have

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

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

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

4.4.4 Coordinates Calculation: General Case.

As aforementioned, the camera is not always mounted perpendicular to the wall. The

camera body may 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.

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

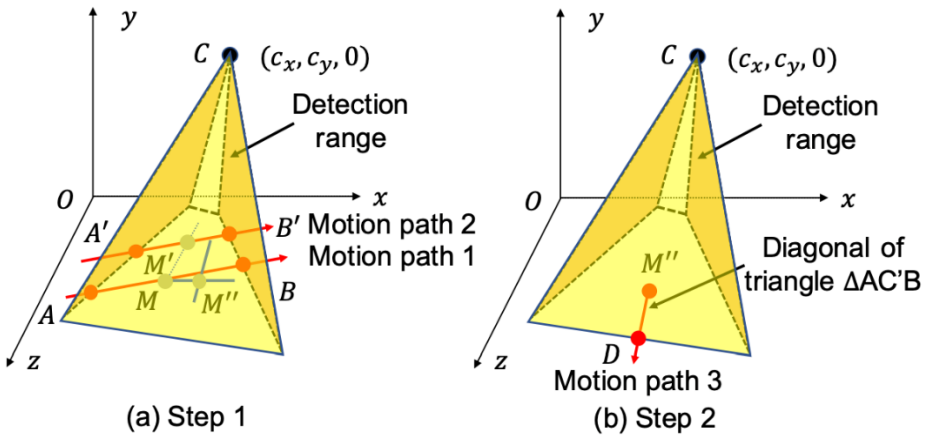


Figure 11: Improved procedure for the general case.

1. As shown in Figure 11(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. Four points A, B, C' and B' are obtained, and 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). With the points A, B and C', the user can measure angle $\angle AC'B$ and find its angle bisector.

The user can draw another line parallel to the x-axis and through point M, and it will intersect the above angle bisector at a point, denoted with M''.

2. As shown in Figure 11(b), the user then gets out of the detection range 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, the location is marked as D. The start time and the time the user leaves the detection range are recorded as t_5 and t_6. 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 Figure 12, 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

$$c_x = a_x + \frac{a_z}{tan(\epsilon+\theta)} \tag{5}$$

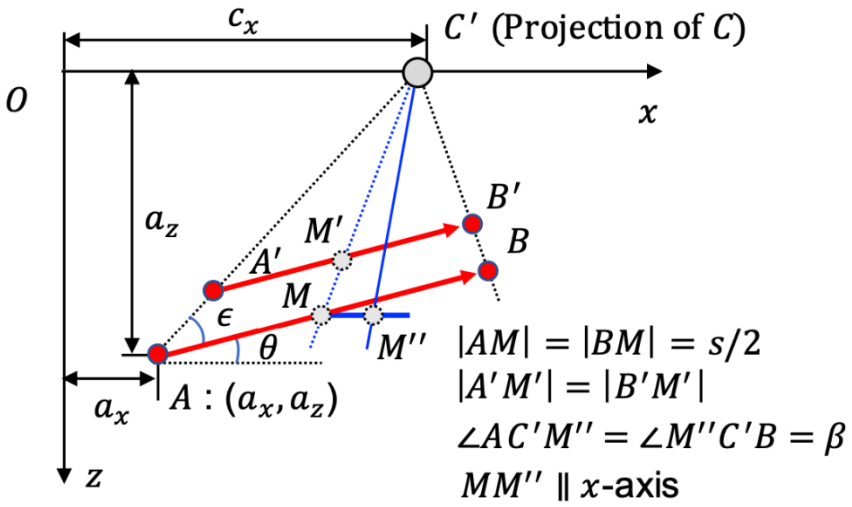


Figure 12: Calculation of c_x for the general case.

Meanwhile, we can calculate the z-coordinate m_z of point M as $m_z = a_z - \frac{s}{2} \cdot sin\,\theta$. As the line MM'' is parallel to the x-axis, the z-coordinate m_z'' of point m'' is equal to 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 like 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., $\gamma = \left(\frac{\pi}{2} - \alpha\right) + \frac{\delta}{2}$. We then have $\tan\gamma = \frac{|C'M''|+|M''D|}{|C'C'|}$, and thus obtain

$$c_y = \frac{(2a_z - s \cdot sin\theta)/(2 \cdot sin(\epsilon+\theta+\beta)) + s'}{tan[(\pi - 2\alpha + \delta)/2]} \qquad (6)$$

CHAPTER V
EXPERIMENTAL EVALUATION

We implement *MotionCompass* on the Android platform. Figure 13 shows the designed user interface (UI). The default mode for a smartphone's network interface card (NIC) is managed mode, in which it only listens to the traffic that comes for it and discards any packets not destined for it. While *MotionCompass* needs to listen to all the wireless traffic nearby, the NIC needs to be in monitor mode. We 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 [32].

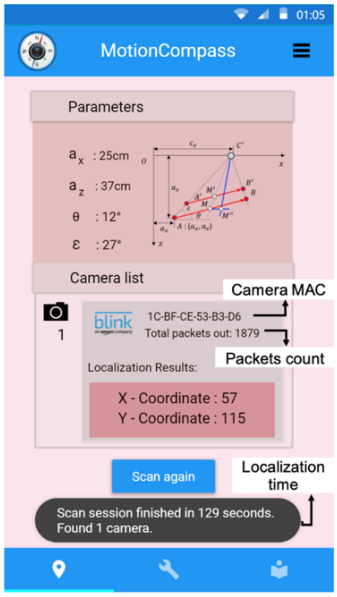


Figure 13: The UI snapshot of *MotionCompass* when one camera on a wall is localized.

5.1 EVALUATION SETUP

The adversary first scans the possible MACs for wireless cameras and then performs manual 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 between the first motion path and the wall, the adversary can then calculate the exact location of the camera.

Testing Cameras: We test 18 most popular wireless cameras, as shown in Table 2. Those cameras can be divided into two groups:

G1 consisting of cameras (ID 1-15) with one motion sensor.

G2 including cameras (ID 16-18) with two motion sensors.

Table 2: The list of wireless security cameras we test.

Camera ID	Model	Amount of PIR Sensors
1	AIVIO Cam	1
2	Arlo Essential	1
3	Arlo Pro 2	1
4	Arlo Pro 3	1
5	Blink Indoor	1
6	Blink XT2	1
7	Blue by ADT	1
8	Canary Flex	1
9	Conico Cam	1
10	EufyCam 2C	1
11	Reolink Argus 2	1
12	Reolink Argus Pro	1
13	Ring Door View	1
14	Simplisafe Cam	1
15	Swann Wireless	1
16	Arlo Ultra	2
17	Ring Spotlight	2
18	Ring Stickup Cam	2

Testing Scenarios: two scenarios are tested.

- **Outdoor**: we conduct the experiment outside a typical American single-family house. The camera is installed at five different locations with different fields of view on the front

outside wall (with a width of 10 m and a height of 5.5 m).

- **Indoor**: we select a bedroom to perform the experiment. We also 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.

Figure 14 shows the selected positions for the camera in 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 recording capability of the camera is highly restricted under these circumstances.

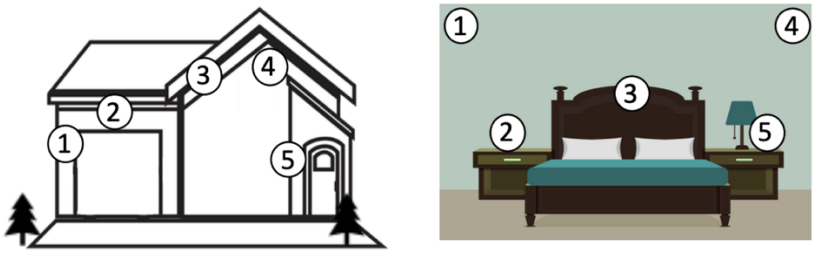


Figure 14: Layout of the experimental environments.

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.

5.2 CASE STUDY

In this example, a Blink XT2 camera is installed at Location 2 of the house, as shown in Figure 14(a). *MotionCompass* is then launched for 10 times. We manually rotate the camera horizontally or vertically to make the camera aim at different areas for each time.

During an example test, the user determines there exists a wireless camera monitoring the target area (i.e., the driveway of the house). Figure 15 shows the traffic flow generated by different devices. The user initiates the continuous movement in the target area. We observe a strong correlation between the camera traffic throughput and the motion. The count of newly generated packets matches with the newly performed motion. However, non-camera traffic flows do not have an obvious relationship with the motion. By comparing MAC addresses, we further find the non-camera traffic flows 1 and 2 belong to an iPhone in use and an Android device in standby mode, respectively. For all 10 tests, the camera traffic is identified successful.

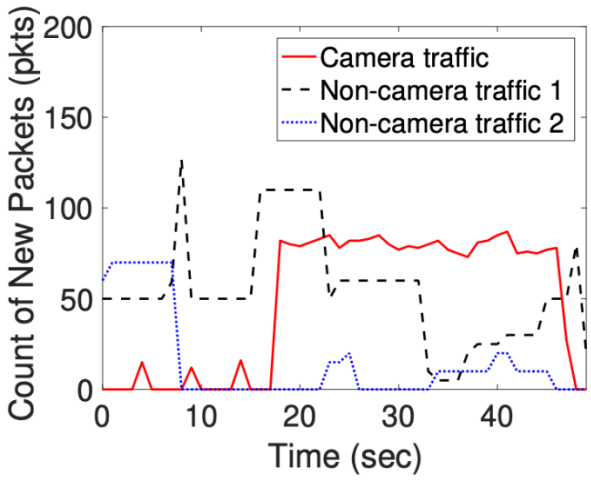


Figure 15: Identifying camera traffic.

We compare estimated *x*-and *y*-coordinate of the camera with their respective true values, and also calculate the localization error. Figure 16 shows the localization error along with errors in *x*- and *y*-coordinate.

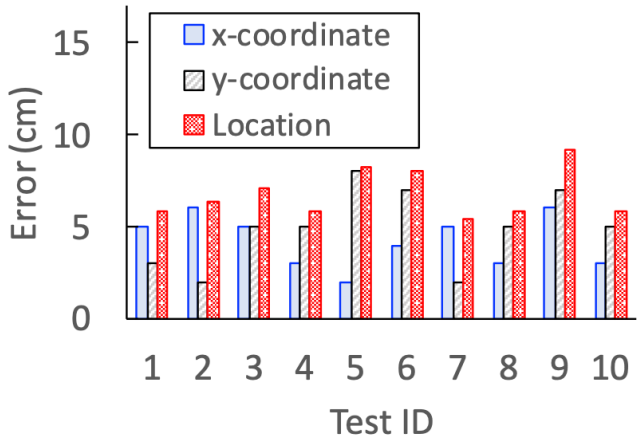


Figure 16: Localization accuracy.

We observe that the localization error is consistently below 9.2 cm. In most tests, the error in *x*-coordinate is slightly smaller than that in *y*-coordinate. Also, the average errors in the two coordinates are 4.2 cm and 4.9 cm, respectively. These results show that *MotionCompass* can achieve a high accuracy.

Figure 17 shows the localization time for each test. We can see the camera can be localized within a range of 125 to 142 seconds, demonstrating the efficiency of *MotionCompass*.

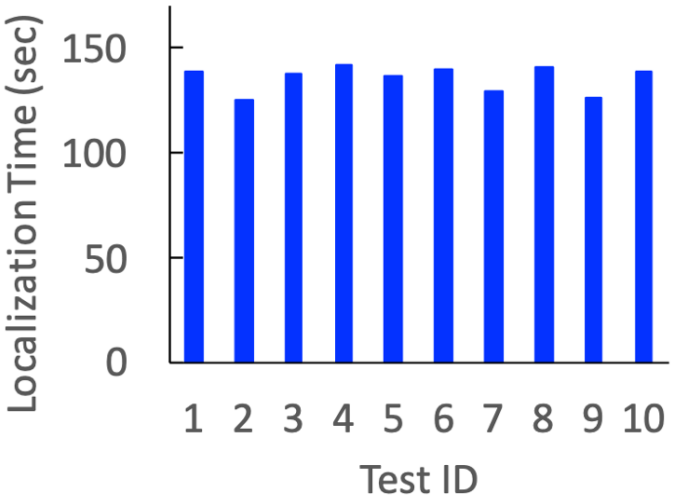


Figure 17: Localization time.

Impact of Camera's Initial Angle: The camera may have a non-right angle along the wall in the *xz*-plane (i.e., ground). We refer to such an angle between the wall and the camera as the *initial angle*, denoted with ψ. The motion sensor of the Blink XT2 camera has an about 105° horizontal angle of view. The initial angle can be thus adjusted from 52.5° to 90◦; otherwise, the field of view would be obstructed by the wall. We vary ψ from 90° to 55°, with decrements of 5◦. For each ψ , we perform 10 attempts to localize the camera.

Figure 18 shows the localization errors for different initial angles. We can see the localization error remains consistently small for different ψ. Specifically, the median localization error ranges from 5.7 to 8.0 cm, and the average localization error is just 5.2 cm.

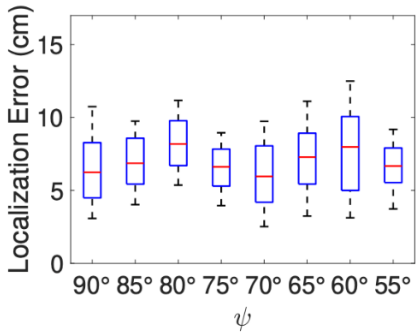


Figure 18: Localization error vs. initial angle (ψ).

Figure 19 presents the corresponding localization time. We can observe the median localization time ranges from 130 to 136 seconds. These results demonstrate that *MotionCompass* is robust to the change of the initial angle.

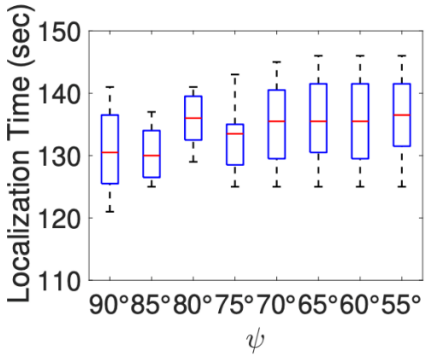


Figure 19: Localization time vs. initial angle (ψ).

Impact of Movement Speed: We change the user's movement speed υ from 0.2 m/s to 1.0 m/s, with increments of 0.2. For each υ, we perform 10 attempts of *MotionCompasss* to localize the camera.

Figure 20 illustrates the localization errors when the movement speed varies. We observe the localization error slightly increases with the value of υ, demonstrating the robustness of *MotionCompass* to the speed variation. Specifically, when υ=0.2 m/s, the mean localization error is 3.6 cm, while it increases to 10.7 cm for υ=1.0 m/s. This is because a higher speed would naturally result in larger error in distance measurement. On the other hand, with a higher speed, the localization can be finished in a shorter time.

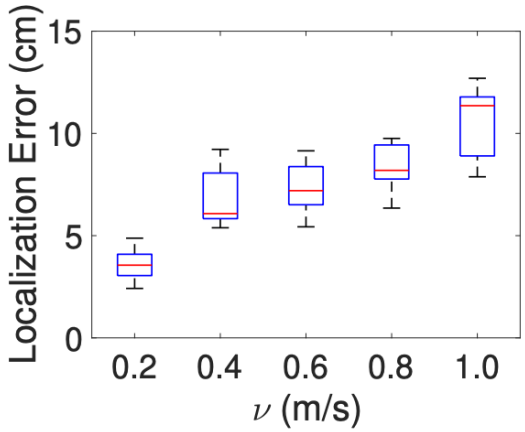


Figure 20: Localization error vs. speed (*v*).

Figure 21 shows the relationship between the localization time and the movement speed. We observe that the median localization time equals 150 seconds when υ is 0.2 m/s, and it drops to 117 seconds when υ is increased to 1.0 m/s.

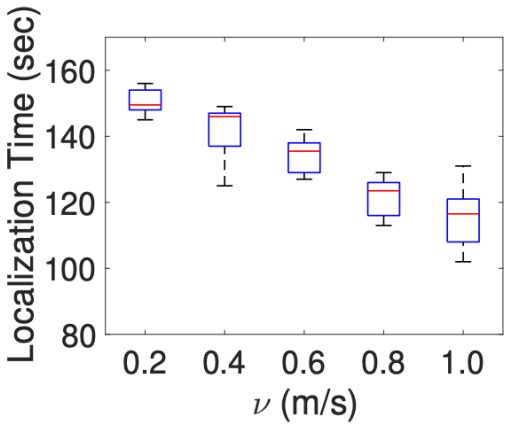


Figure 21: Localization time vs. speed (v).

5.3 OVERALL LOCALIZATION PERFORMANCE

We test all cameras in both indoor and outdoor environments. For localizing each camera at every selected location, we perform 25 trials. Thus, we have 18×2×5×25=4,500 attempts in total. For each attempt, we compute the localization error and record the time spent on finishing the task (i.e., localization time).

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 Figures 22 and 23. 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 environments, on average, a camera in G2 causes a larger localization error and requires 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 outdoor environment compared with indoor. This appears due to the fact that the outdoor environment provides a wider space and the user may spend longer time generating the simulating motion.

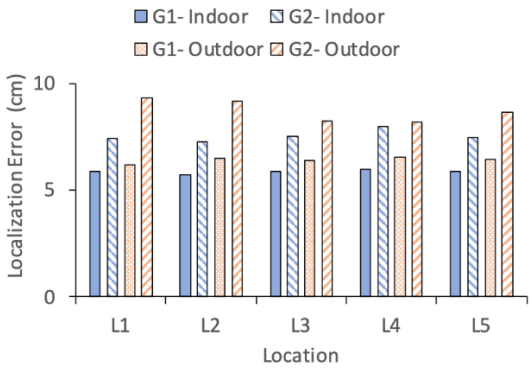


Figure 22: Mean localization error.

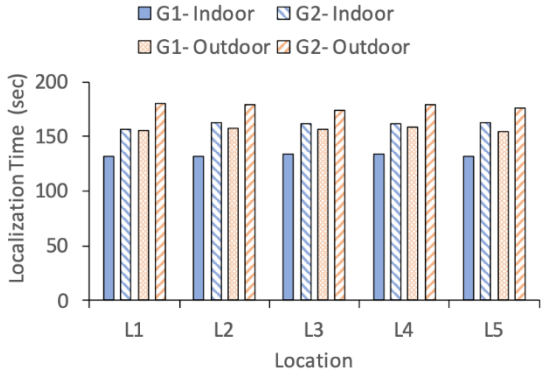


Figure 23: Mean localization time.

Figures 24 and 25 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. Specifically, in the outdoor environment, the median localization error has a range of 3.7 to 6.5 cm for cameras 1-15, and 7.8 to 9.2 cm for cameras 16-18. Meanwhile, *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. These observations convincingly show *MotionCompass* works for different cameras in both environments, and cameras with two motion sensors cause slightly higher localization errors than cameras with one motion sensor.

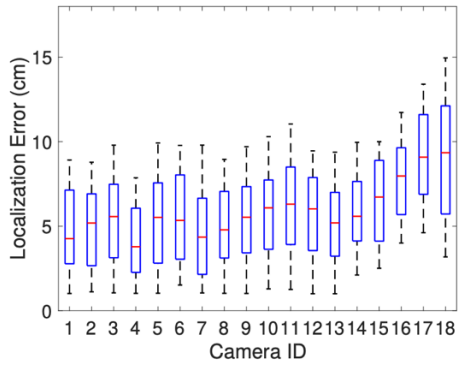


Figure 24: Outdoor localization error.

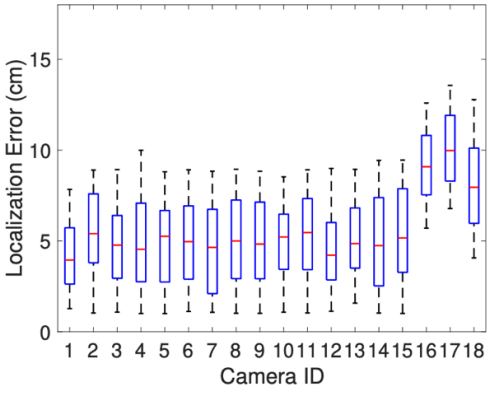


Figure 25: Indoor localization error.

Figure 26 plots the empirical cumulative distribution functions (CDFs) of the localization errors D_{in} and D_{out} for different groups of cameras under indoor and outdoor environments. We can see for cameras in G1, D_{in} and D_{out} are less than 9.0 cm with probabilities 98.1% and 92.0%; for cameras in G2, D_{in} and D_{out} are less than 12.0 cm with probabilities 94.5% and 88.8%. These

results again demonstrate that outdoor environment or more motion sensors may lead to a higher localization error, but also confirm conclusively that *MotionCompass* is robust against different environments and cameras.

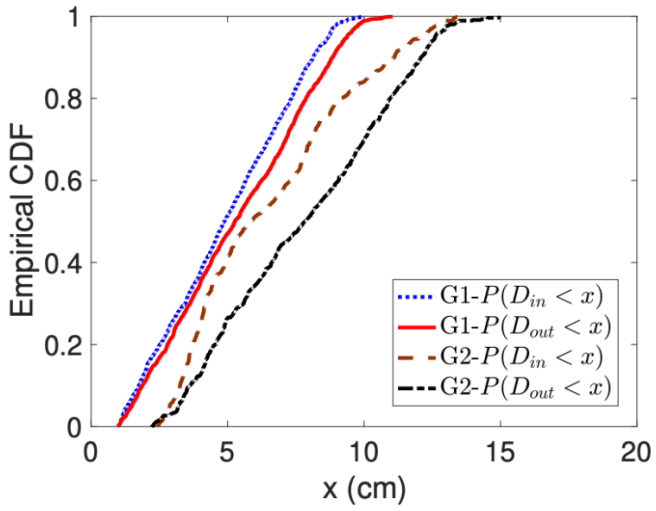


Figure 26: CDFs of D_{in} and D_{out}.

Figure 27 plots CDFs of the localization time T_{in} and T_{out} for G1 and G2 under both environments. Overall, the outdoor environment or more motion sensors cause longer localization time. Specifically, T_{in} and T_{out} are less than 137 and 161 seconds with probability 90.0% for G1, and they are less than 166 and 185 seconds for G2 with the same probability.

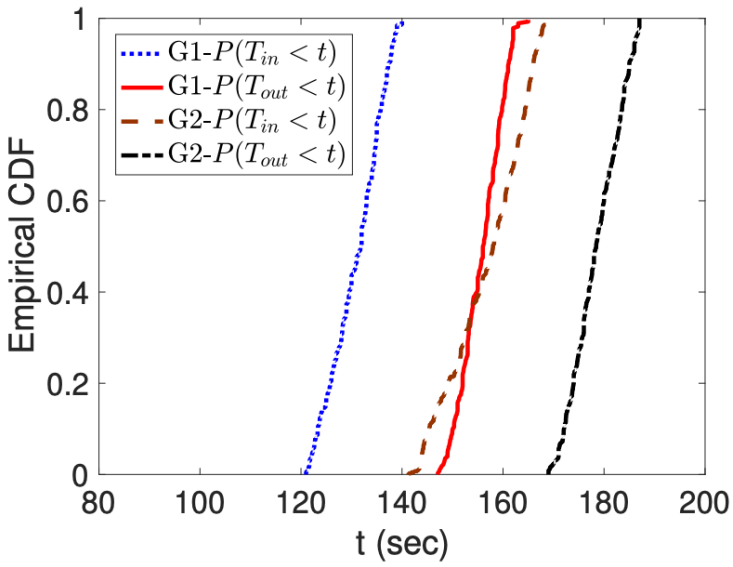


Figure 27: CDFs of T_{in} and T_{out}.

5.4 LOCALIZATION OF MULTIPLE CAMERAS

In some situations, there are multiple wireless cameras with overlapped fields of view in an area. The proposed method monitors and analyzes the wireless traffic based on MAC. Therefore, we can simultaneously monitor multiple traffic flows, each of which belongs to a corresponding wireless camera, and different cameras will not interfere with each other's localization.

To verify the effectiveness of pinpointing multiple cameras, we deploy different numbers (1 to 5) of cameras in the tested room. For multiple cameras, we manually adjust their fields of view and make them partially overlapped. For each camera count, *MotionCompass* is launched for 25 attempts. We randomly change the location of each camera every attempt. We find *MotionCompass* can successfully find each camera with a small localization error, similar with which we obtain for localizing a single camera.

Table 3: Localization time vs. camera count.

Camera count	Localization time (seconds)		
	Average	Minimum	Maximum
1	132	126	149
2	276	252	295
3	387	368	412
4	510	501	559
5	622	609	677

Table 3 shows the mean, minimum, and maximum localization time for different numbers of cameras. We observe the localization time is almost proportional to the camera count, demonstrating that performing localization of multiple cameras equals performing localization of a single camera for multiple times.

5.5 USER STUDY

We recruited 5 volunteers and asked each of them to perform *MotionCompass* to figure out the location of a hidden wireless camera randomly selected and deployed in the outdoor or indoor environment. Every participant performed 25 attempts for each environment. For each deployment, 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. We compare the localization result and the true location of the camera to quantify the localization error.

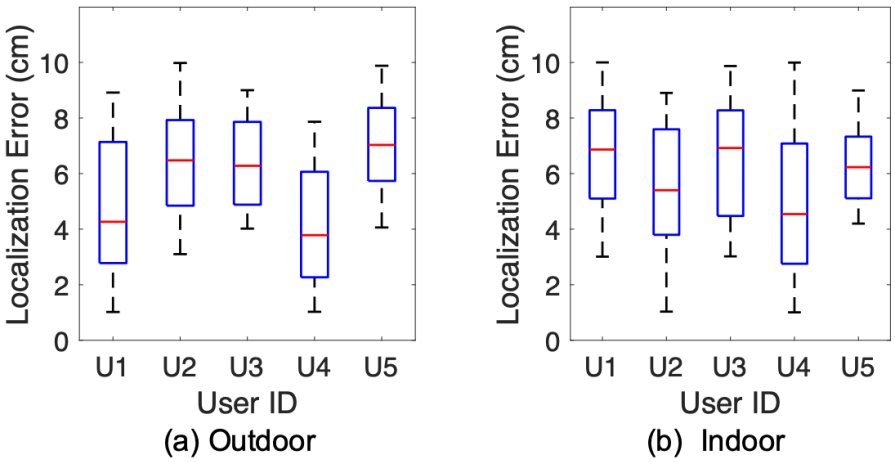


Figure 28: Localization errors for different users.

Figure 28 shows the obtained localization errors for different users under outdoor and indoor environments. 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), 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 of *MotionCompass* is quite consistent among different users.

Table 4 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, indicating a user can normally localize the camera with less than 150 seconds. This verifies the practicality of the proposed camera localization strategy.

Table 4: 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

CHAPTER VI
DISCUSSION

6.1 LIMITATIONS

The Requirements of WiFi and Motion Stimuli. *MotionCompass* sniffs in 802.11 (WiFi) networks and it does not work for cameras using cellular connections. Also, it requires generating motion in the target area in order 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 cost and require the attacker to operate the drone/robot flexibly. We leave exploring the impact of different motion stimuli to our future work.

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 [5]. 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 customization scenarios. First, the owner only creates one activity zone: if the customized activity zone has a slightly smaller size compared with the default one (or if there

is a small obstacle), we expect that *MotionCompass* will still work, but 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 the motion within a small area. Second, the owner creates multiple activity zones: the attacker can first 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 Videos. Cameras with local SD cards may not upload any videos. Thus, no real-time wireless camera traffic would be generated, leading to the failure of *MotionCompass*. Meanwhile, such cameras also prevent their owners from receiving the recording notifications in time.

Wide-angle Cameras. If a wide-angle camera covers the whole room, the relationship between the intercepted wireless traffic and the performed motion path cannot be obtained. Thus, *MotionCompass* fails. However, using such a mode (with full field of vision), the camera would be triggered frequently. As a result, the battery may be depleted quite quickly.

6.2 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., [33]) so that the camera goes into standby mode and will not respond to any motion. However, this solution is 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.

A practical defense is to randomize the recording length. Since we cannot predict when the motion occurs and need to make sure that the owner can receive 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. As a result, the attacker obtains inaccurate localization results. This technique, however, will speed up the camera's power consumption.

Alternatively, we can postpone uploading motion-induced video. Once the motion is detected, current wireless cameras not only instantly send an alert, but also start recording and immediately upload recorded 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. In this way, the motion detection capability is 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, the attacker is unable to pinpoint the camera. However, this defense also has disadvantages, such as requiring the camera to be equipped with a large local memory.

CHAPTER VI
RELATED WORK

Traffic Analysis: Traffic analysis can achieve various applications, including detecting drones [31, 35], inferring apps [36, 38], monitoring misbehaving apps [42], enforcing network access restrictions [14], identifying actions on apps [11], and detecting hidden wireless cameras [10]. Unlike existing traffic analysis-based approaches (e.g., [10, 36, 38, 42]), which utilize the inherent traffic patterns (side-channel information leaks) to detect devices or apps which generate them, our work correlates the traffic pattern with human activities.

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, 22, 24, 29, 39]. For example, [10] proposes a hidden wireless camera detection approach by utilizing the intrinsic traffic patterns of flows from wireless cameras. [24] investigates the responsive traffic variation corresponding to the light condition change to determine whether the traffic is produced by a wireless camera. [29] 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.

Besides, [22] utilizes exterior received signal strength (RSS) measurements to localize wireless cameras behind the wall with room-level accuracy (i.e., a median error of around 4-5

meters). With this method, the adversary does not need to communicate with the target. As the RSS-based approach and our method deal with wireless camera localization in different scenarios leveraging different tools, they can be complementary.

CHAPTER VII
CONCLUSION

We leverage wireless traffic stimulated by specifically designed motion and propose *MotionCompass*, a lightweight technique for pinpointing the location of a wireless camera. Its novelty stems from identifying and proving the motion activation property of wireless cameras that 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 develop an Android app to implement *MotionCompass* and verify its effectiveness and efficiency via extensive real-world experiments.

This work has a paper version [45] which was accepted on ACM MobiSys '21: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services with DOI 10.1145/3458864.3467683.

APPENDIX I
USING SNIFFING TOOLS

We motioned the method of sniffing wireless network traffic in Section 4; this section aims to provide more details on how we get traffic sniffing works. Our work requires packet capture to extract traffic and correlate traffic and user movement. But the most challenging part is choosing the proper capture method for the WiFi packet. Our work proposes a technique that works primarily in public environments like Hotels or Homes, so the network package capture must always be available. There are two types of traffic capture methods:

Active: Active sniffing is using a method which is actively scanning the nearby traffic using APIs like PCAP. But it requires the user has access to the network.

Passive: Passively sniffing means it can scan the network using tools like Airmon-ng[7] and monitor the network traffic over the air. Even though parts of the traffic were encrypted, the sniffing can still exact helpful information for traffic analysis. This is what we chose in our system and motioned before.

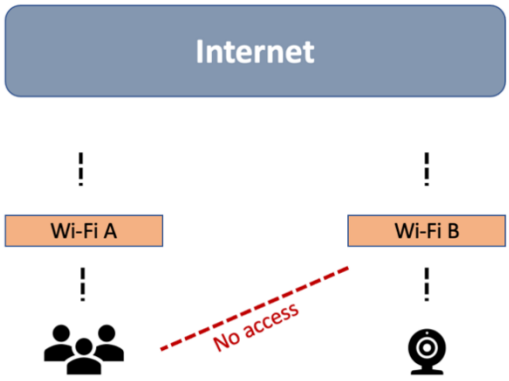


Figure 29: Problems if there is no access to the Camera's network

We know it's not guaranteed that the camera is connected to a network that gives the user access. Just like the scenario shown in Figure 29, I chose a passive method in my work to get the camera's traffic.

Due to the WiFi standard, which states the package-level transportation methods [12], we can see from the protocol that the packet header is constantly exposed to the public. We can sniff the network and get the total packet counts and the original hardware MAC address from the broadcasted WiFi traffic; then, we will know the length of the traffic and which device it is based on OUI as we did. Airmon-ng is a tool to help us monitor network devices across all WiFi channels. It is a command line tool that utilizes the monitor mode of a WiFi interface and is used to intercept WiFi communication over the air. A general approach for using it is running it on a laptop with the monitor mode of the network interface enabled.

The primary user interface of the Airmon-ng shows in Figure 30, which illustrates that the tool scans all the AP (Access point) devices (Based on BSSID) around the user. The AP devices are mostly wireless routers independent of each other's, so the first step is clear now. The first step is using the tool to get a list of all APs nearby; the camera devices must connect to one of the AP if it exists. We can also see much information exposed here, like the data volumes transferred by the AP or the encryption type and even the amount of beacons on the AP.

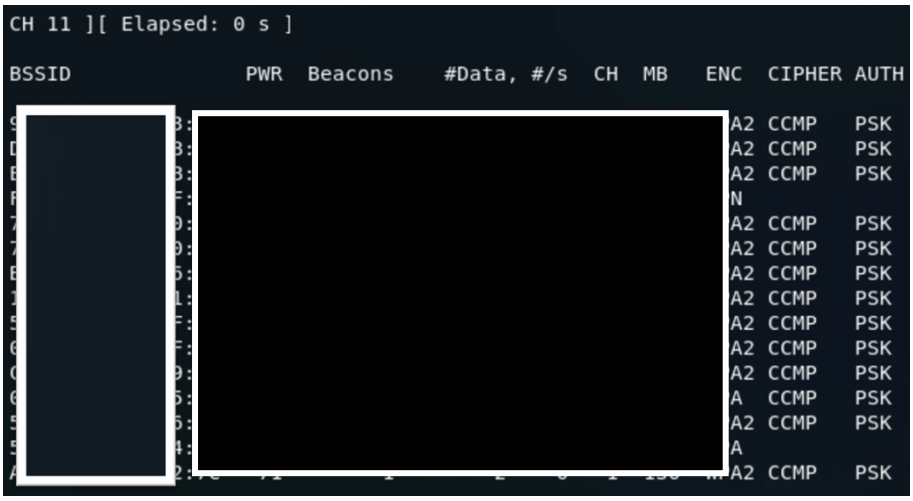


Figure 30: Using Airmon-ng under the command line

The next step here is to determine the camera MAC address; we already obtain all the MAC addresses of the APs. In the world of WiFi networks, all devices connected can be called 'Stations.' We can find all stations under one AP using the same tool. The example of the result shown in Figure 31: all devices appear as 'Stations' with their RSSI (Signal Strength) and data frames from the devices. The tool can easily monitor all devices connected to the BSSID of the AP by repeating the processes; then, we are possible to loop through the list of BSSIDs to find the possible camera devices under that AP. We can check all stations in the next step by using OUI to match them.

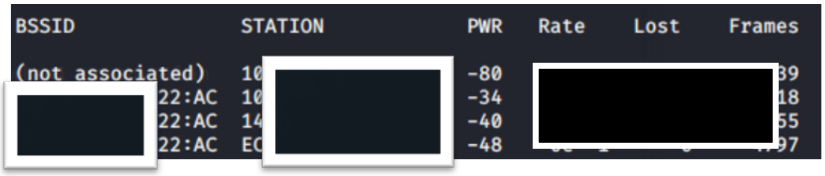
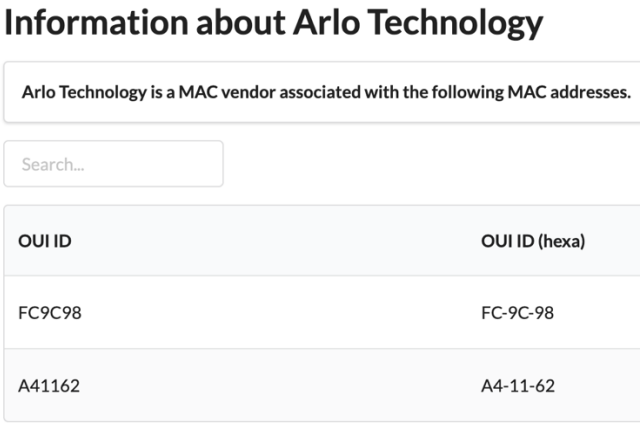


Figure 31: Using Airmon-ng to find stations

The next step is using OUI to find if there is any match in the OUI database that appears like a wireless camera. A quick example is using Arlo wireless camera as an example; we can see the device marked with two OUI prefixed are claimed to be Arlo's camera devices in Figure 31. So, it means if we find a certain OUI in this list from the station scan in the previous step, we can assure that an Arlo camera is deployed nearby.



Information about Arlo Technology

Arlo Technology is a MAC vendor associated with the following MAC addresses.

Search...

OUI ID	OUI ID (hexa)
FC9C98	FC-9C-98
A41162	A4-11-62

Figure 32: Using OUI to match Cameras

(https://ouilookup.com/vendor/arlo-technology)

The final part is simple, as we already found the camera target. We need to lock in one camera device as a target "station" to monitor its traffic volumes. We can record the increment of the packet count by seconds when we are walking. The increment can show us if one camera is actively transmitting packets, then we know if we are still in the camera's field of view.

APPENDIX II
INTEGRATION OF PASSIVE TRAFFIC CAPTURE

We can use Airmon-ng on a laptop as a general approach to passively sniffing the nearby network traffic, which is mentioned in APPENDIX I. However, the general approach provided by Airmon-ng [7] requires a laptop/PC that can connect to a wireless interface with monitor mode enabled. This is inconvenient for us, because it's not easy to carry, especially for a user who needs to monitor the wireless traffic and perform walking simultaneously. Therefore, we seek to integrate the traffic scanning capability into one smartphone. We use the following approaches to successfully build an application *MotionCompass* with Airmon-ng embedded in it.

Not all smartphones can access the monitor mode of the wireless interface. Some do not provide the necessary permissions to access the API for system-level controls. For Android-based systems, we may root them and get system-level access permissions. However, only some wireless interfaces support the monitor mode; we choose a particular phone (Nexus 5) that natively supports monitor mode and flash it with the Kali Nethunter system [32]. The Kali system turns on the monitor mode of the wireless interface and provides root access to the system's terminal for command lines. This is the start for the integration of Airmon-ng. We also need a customized application to monitor user movement and automatically sniff traffic.

For the next step, we develop an android application for *MotionCompass*. This application embeds a complete toolkit of Airmon-ng using Android NDK (Android native development kit), which allows cross-compile Airmon-ng on the native android platform. We need this to run the

tool on android, and we can then call the device within the application. The most important issue is solved as soon as we can access the Airmon-ng within the application. The rest of the work monitors the user's movement using an android accelerometer and gyroscope API (application programming interface) to calculate the movement duration and distance.

REFERENCES

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

[2] Arlo Ultra: 4K security camera: 4K wireless camera system, 2020. https: //www.arlo.com/en-us/products/arlo-ultra/default.aspx.

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

[4] Smart WiFi motion sensor, 2020. https://www.bazzsmarthome.com/products/ smart-wifi-motion-sensor.

[5] What are activity zones and how do I create them?, 2020. https://kb.arlo.com/ 1001908/What-are-activity-zones-and-how-do-I-create-them.

[6] K. Abas, C. Porto, and K. Obraczka. Wireless smart camera networks for the surveillance of public spaces. Computer, 47(5):37–44, 2014.

[7] Aircrack-ng. Main documentation – aircrack-ng suite, 2020. https:// www.aircrack-ng.org/documentation.html.

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

[9] Y. Cheng, X. Ji, T. Lu, and W. Xu. DeWiCam: Detecting hidden wireless cameras via smartphones. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, ASIACCS '18, pages 1–13, New York, NY, USA, 2018. ACM.

[10] Y. Cheng, X. Ji, T. Lu, and W. Xu. On detecting hidden wireless cameras: A traffic

pattern-based approach. IEEE Transactions on Mobile Computing, 19(4):907–921, 2020.

[11] 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 Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pages 297–304, 2015.

[12] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai. IEEE 802.11 wireless local area networks. IEEE Communications Magazine, 35(9):116–126, Sep. 1997.

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

[14] 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 Transactions on Network and Service Management, 17(1):45–59, 2020.

[15] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke. The IEEE 802.11 universe. IEEE Communications Magazine, 48(1):62–70, January 2010.

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

[17] IEEE. OUI public listing, 2020. http://standards-oui.ieee.org/oui/oui.txt.

[18] IPX1031. Survey: Do Airbnb guests trust their hosts?, April 2019. https: //www.ipx1031.com/airbnb-guests-trust-hosts/.

[19] X. Ji, Y. Cheng, W. Xu, and X. Zhou. User presence inference via encrypted traffic of

wireless camera in smart homes. Security and Communication Networks, 2018:1–10, 09 2018.

[20] 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 Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, WiSec ’19, page 239–244, New York, NY, USA, 2019. Association for Computing Machinery.

[21] A. Li. Security camera blind spots: How to find and avoid them, November 2018. https://reolink.com/find-and-avoid-security-camera-blind-spots/.

[22] Z. Li, Z. Xiao, Y. Zhu, I. Pattarachanyakul, B. Y. Zhao, and H. Zheng. Adversarial localization against wireless cameras. In Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications, HotMobile ’18, pages 87–92, New York, NY, USA, 2018. ACM.

[23] 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. Applied Physics Letters, 111(24):243901, 2017.

[24] T. Liu, Z. Liu, J. Huang, R. Tan, and Z. Tan. Detecting wireless spy cameras via stimulating and probing. In Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’18, pages 243–255, New York, NY, USA, 2018. ACM.

[25] S. Mare, F. Roesner, and T. Kohno. Smart devices in Airbnbs: Considering privacy and

security for both guests and hosts. Proceedings on Privacy Enhancing Technologies, 2020(2):436–458, 2020.

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

[27] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown. A study of mac address randomization in mobile devices and when it fails. Proceedings on Privacy Enhancing Technologies, 2017(4):365–383, 2017.

[28] J. Martin, E. Rye, and R. Beverly. Decomposition of MAC address structure for granular device inference. In Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC '16, page 78–88, New York, NY, USA, 2016. Association for Computing Machinery.

[29] R. Mitev, A. Pazii, M. Miettinen, W. Enck, and A.-R. Sadeghi. Leakypick: IoT audio spy detector. arXiv preprint arXiv:2007.00500, 2020.

[30] 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 Proceedings of the 14th International Conference on Information Processing in Sensor Networks, IPSN '15, page 142–153, New York, NY, USA, 2015. Association for Computing Machinery.

[31] 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 Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys ’17, page 211–224, New York, NY, USA, 2017. Association for Computing Machinery.

[32] Offensive Security. Kali Linux NetHunter, 2020. https://www.kali.org/kali-linux-nethunter/.

[33] Reolink. How to turn on/off the PIR sensor, 2020. https://support.reolink.com/hc/en-us/articles/360004379493-Turn-on-off-the-PIR-sensor.

[34] K. Scarfone, D. Dicoi, M. Sexton, and C. Tibbs. Guide to securing legacy IEEE 802.11 wireless networks. National Institute of Standards and Technology (NIST) Special Publication, 800:48, 2008.

[35] 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. Computer Networks, 168:107044, 2020.

[36] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Robust smartphone app identification via encrypted network traffic analysis. IEEE Transactions on Information Forensics and Security, 13(1):63–78, 2018.

[37] 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 Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pages 413–424, 2016.

[38] 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 2015 IEEE Conference on Communications and Network Security (CNS), pages 433–441, 2015.

[39] K. Wu and B. Lagesse. Do you see what I see? Detecting hidden streaming cameras through similarity of simultaneous observation. In 2019 IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 1–10, 2019.

[40] 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 Processing Magazine, 35(5):41–49, 2018.

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

[42] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu. Homonit: Monitoring smart home apps from encrypted traffic. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS ’18, page 1074–1088, New York, 2018.

[43] E. Yang, S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu. Wireless Training-Free Keystroke Inference Attack and Defense. In IEEE/ACM Transactions on Networking, vol. 30, no. 4, pp. 1733-1748, 2022.

[44] S. Fang, Y. Liu, W. Shen, and H. Zhu. 2014. Where are you from? confusing location distinction using virtual multipath camouflage. In Proceedings of the 20th annual international conference on Mobile computing and networking (MobiCom '14). Association

for Computing Machinery, New York, NY, USA, 2014.

[45] Y. He, Q. He, S. Fang, and Y. Liu. MotionCompass: pinpointing wireless camera via motion-activated traffic. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21). Association for Computing Machinery, New York, NY, USA, 215–227, 2021.

[46] S. Fang, T. Wang, Y. Liu, S. Zhao, and Z. Lu. Entrapment for Wireless Eavesdroppers, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 2530-2538, 2019.

[47] Q. He, S. Fang, T. Wang, Y. Liu, S. Zhao, and Z. Lu. Proactive Anti-Eavesdropping with Trap Deployment in Wireless Networks. In IEEE Transactions on Dependable and Secure Computing, 2021.

[48] S. Fang, Y. Liu, and P. Ning. Wireless Communications under Broadband Reactive Jamming Attacks. In IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 3, pp. 394-408, 2016.

[49] S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu. No Training Hurdles: Fast Training-Agnostic Attacks to Infer Your Typing. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18). Association for Computing Machinery, New York, NY, USA, 1747–1760, 2018.

[50] S. Fang, Y. Liu, W. Shen, H. Zhu, and T. Wang. Virtual Multipath Attack and Defense for Location Distinction in Wireless Networks. In IEEE Transactions on Mobile Computing, vol. 16, no. 2, pp. 566-580, 2017.

[51] E. Yang, Q. He, and S. Fang. WINK: Wireless Inference of Numerical Keystrokes via Zero-Training Spatiotemporal Analysis. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22). Association for Computing Machinery, New York, NY, USA, 3033–3047, 2022.

ACM Author Rights

Distribute

Authors can post an Author-Izer link enabling free downloads of the Definitive Version of the work permanently maintained in the ACM Digital Library.

- On the Author's own Home Page or
- In the Author's Institutional Repository.

Reuse

Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

- Contributing complete papers to any edited collection of reprints for which the author is notthe editor, requires permission and usually a republication fee.
- Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).
- Commercially produced course-packs that are sold to students require permission and possibly a fee.

This work has a paper version [45] which been accepted on ACM MobiSys '21: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services with DOI 10.1145/3458864.3467683. As shown on author rights above, I can reuse the content from this paper as the author.

https://authors.acm.org/author-resources/author-rights