UCLouvain

École polytechnique de Louvain

Reverse-Engineering the Physical Configuration of Smart Homes

Author: **Martin Vivian**
Supervisor: **Ramin Sadre**
Readers: **Benoît Macq, Igor Zavalyshyn**
Academic year 2020–2021
Master [120] in Computer Science

Acknowledgement

First of all, I thank my supervisor, Ramin Sadre, for his help, availability, and enthusiasm. He regularly challenged my ideas, permitting me to reach the goal of this thesis.

I also thank Igor Zavalyshyn for having lent me the devices, essential to the achievement of the experiments.

Also, special thanks to Benoît Macq and Igor Zavalyshyn for their agreement to read this thesis and to take part in the jury.

Finally, I also thank my family for their support and encouragement to realize this thesis.

Abstract

A major concern with cyber-security for Smart Homes are attacks against the privacy of their residents. Unfortunately, as has been shown in recent years, traditional cyber-security techniques, such an authentication, firewalls, and encrypted communication, are not sufficient to protect against privacy-invasion attacks. Researchers have shown that an attacker can detect the activities of the occupants by passively monitoring the encrypted wireless network traffic of the smart devices.

In this thesis, we focus on the question how the smart devices in a Smart Home are physically arranged. More specifically, we want to identify devices that are located in the same room or at least in close proximity. Similar to other works, we will only rely on information obtained through passive network measurements. We present techniques to determine whether a smart lamp and an IP camera are located in the same room. We then extend those techniques to cameras that have an automatic infrared mode switch. Finally, we show how the relative position and orientation of multiple cameras can be estimated.

Contents

Chapter 1

Introduction

The Internet of Things (IoT) has turned into an immense success in the past years. Nowadays, IoT devices and applications can be found everywhere, in the streets (Smart Cities), in factories (Industrial IoT and Industry 4.0), in homes (Smart Homes), and soon also in interconnected self-driving cars. Not entirely unexpectedly, this success has also attracted the interest of malicious actors. We have witnessed more and more attacks against IoT systems in the past years. Among those, one of the most spectacular ones was the Mirai malware that infected probably millions of networked devices and turned them into members of a botnet, in this way misusing them to attack other Internet hosts and services [1].

With the raising popularity of IoT in Smart Homes, it can be expected that attacks against IoT systems will not only increase in frequency, but also become more harmful with an impact beyond the virtual world of the Internet. Similar to attacks against industrial control systems, an attack against a Smart Home can cause physical damage, for example by disturbing the control of a heating system. Another major concern of Smart Home security are attacks against the privacy of the residents. An attacker who gains control over devices in a Smart Home gets access to sensitive data that concerns the private life of the residents. An IP camera is the most drastic example of a privacy-sensitive device [2], but more humble devices can be transformed into spying instruments, too: a smart lamp with a motion sensor could tell when the house owners are at home, a smart thermometer when they go to bed; a fitness tracker reveals information about the health of the person wearing it, etc.

Unfortunately, as has been shown in recent years, traditional cyber-security techniques, such as authentication, firewalls, and encrypted communication, are *not* sufficient to secure Smart Homes against privacy-invasion attacks. Researchers have repeatedly demonstrated that attackers can obtain information about the activities of the members of a Smart Home household just by passively monitoring ("sniffing") the encrypted wireless network activities of the smart devices [3, 4, 5, 6, 7, 8]. To

achieve this, researchers use machine learning to train activity detection systems on labeled traffic datasets. In that way, a detection system can for example learn that a specific traffic pattern stands for a triggered door sensor. This can be extended to more complex human-device interactions: entering a room results in network activities first of the door sensor and then of a motion sensor in the same room; a system trained on a suitable dataset can conclude that a person has entered the room if it observes the network activities of those two devices in the above order.

In this thesis, we look at a related problem. We want to know how the smart devices are physically arranged in a Smart Home. More specifically, we want to identify devices that are located in the same room or at least in close proximity. Similar to the above works, our goal is to obtain this information from passive network measurements. The knowledge gained in this way can be used for benign purposes, such as troubleshooting or localizing forgotten or unregistered devices. But it could be also used my malicious parties. For example, they could learn whether there are rooms that are monitored by several cameras (e.g. a visible one and a hidden one). Combined with external observations of the Smart Home (e.g. illuminated windows), they could determine which rooms to avoid for a burglary or which rooms might contain valuable objects.

Our approach could be applied to various kind of devices. However, in this thesis, we focus on smart lamps and IP cameras. Concretely, we first present an algorithm to detect whether an IP camera and a smart light are in the same room. We then introduce an alternative detection approach for cameras with an automatic infrared mode. Finally, we show how the relative position and orientation of multiple cameras can be estimated.

The thesis is organized as follows. Related work is discussed in Chapter 2. Our approach is presented in Chapter 3. Our implementation and tools used are presented in Chapter 4. We present the results of our experiments and a discussion of the limitations of our approach in Chapter 5. The limitations are discussed in Chapter 6. The Thesis is concluded in Chapter 7.

Chapter 2

Related work

This chapter describes the related works. First of all we describe all the work to recognize a fingerprint of the connected devices. Then the activity detection of which some works have been reused in our approach and finally the defenses.

2.1 Device fingerprinting

In our adversary model (see Section 3.1), we assume that the attacker is able to identify the type of device from passive network measurements. Many methods to do this can be found in the literature. Simple methods based on techniques such as MAC address analysis or analysis of DNS queries are described in [9, 10]. Methods applying machine learning algorithms on observations of the IP address, the protocols, the ports, or the packet lengths can be found in [11, 12, 3, 13]. More specific to IP cameras, Li *et al.* describe techniques to recognize typical fluctuations in the network traffic caused by video compression [7] . This is possible for some codecs, like for example H.264 that generates distinct patterns of I frames and P frames in the packet stream. Finally, Amar *et al.* [10] analyze data collected at a testbed consisting of various types of devices, including Apple TV, Hue Bridge, and Amazon Echo. They establish statistics on the typical communication of these devices, such as their consumption of bandwidth, the frequency of communication with other services like Dropbox services, etc. All this information can be used to derive a unique fingerprint of a device that helps to identify it later.

2.2 Activity detection

Here, activity detection refers to the problem of inferring device (or human) activities in a Smart Home from observations of the device's network traffic.

Acar *et al.* [3] determine the state of IoT devices (for example, if they are on or off) by using different machine learning classifiers like Random forest. With this information, they are able to recognize some activities that involve a state change. They also discuss time-depend activities. For example, someone who walks from room A to room B will trigger device activities that appear in a specific order. Zhan and Haddadi [14] identify the trajectory of multiple people in a house from measurements, provided the position of the sensors in the house is known. Their approach uses machine learning algorithms and graph theory.

Many IoT devices in Smart Homes are controlled by smartphone applications. Junges *et al.* [6] show that activities can be recognized by looking at the packets sent by the smartphone to an IoT Gateway. They do that by analyzing the number of bytes of the commands sent by the user. This approach has been tested on smart lamp holders and smart plugs.

Cheng *et al.* [4] describe an algorithm to detect human movement by monitoring the traffic rate of the IP camera. They implement their algorithm in an Android application running on a smartphone with a WiFi-card in monitor mode. The latter allows them to eavesdrop wireless network traffic without joining the network. Moreover, they are able to identify the camera device. Their *CUSUM* approach used by their algorithm to detect movements is also used by us (see Chapter 3).

Li *et al.* [7] present a technique to identify the type of movement a person does in front of a camera. They recognize activities such as eating, dressing, moving, styling hair, etc. To this end, they observe the traffic size evolution over time. They divide the traffic timeseries into multiple segments by employing Binary Segmentation. For example, to recognize eating, they find the segment that corresponds to getting food and the segment corresponding to chewing the food. In that way, the pattern of an eating activity is the concatenation of those two segments.

The influence of the luminosity on the traffic sent by an IP camera is studied by Wampler *et al.* [15]. They show for different camera brands that the traffic decreases when the light in the room is turned off. The reason for this is that a camera is not able to record scene details if the light is too low. This increases the compressibility of the image and, therefore, reduces the amount of traffic sent by the camera. They also observe that luminosity impacts the quality of the detection of a movement.

2.3 Defenses

It is difficult to defend against the attacks described above since they do not require access to the payload of the network traffic. Countermeasures proposed so far by the research community are not very practical since they either require modifications in the network and application protocols used by the devices or the injection of

significant amounts of fake traffic into the Smart Home network to hide the real traffic from a malicious observer. Moreover, many of those countermeasures are simply not effective against sophisticated analysis techniques [3, 16, 17].

Chapter 3

Approach

In this chapter, first, we describe our approach to determine whether an IP camera and smart lamp are in the same room. Next, we extend this to the automatic infrared mode of the IP camera. Finally, we will explain how to establish a mapping thanks to human activity and cameras. Moreover, we will show how this approach with human movement, allows us to have a relative estimation of the orientation and position of several cameras.

3.1 Adversary Model

We assume that the attacker is able to perform traffic rate measurements on the network used by the IoT devices. If devices use different networks or network technologies (e.g. WiFi and Zigbee), the attacker has to monitor all those networks. The monitoring happens in a completely passive way, i.e., the attacker can neither inject/modify traffic nor physically interact with the devices.

We also assume that the attacker is able to identify the type of the devices, i.e., which devices are IP cameras and which are lights. This is not difficult to do since IoT device types have distinct network fingerprints. For further details, we refer to the existing literature on device fingerprinting (see Chapter 2 for some examples).

3.2 Determining the colocation of devices

We start with a description of a new algorithm to determine whether an IP camera and a smart light are in the same room (or at least close enough that the light has an impact on the camera). Our approach is based on the observation made by Wampler *et al.* [15] (see Chapter 2) that the brightness of a scene influences the amount of traffic sent by an IP camera per time unit. In fact, the camera's traffic rate (measured in the following in packets/s) follows a bimodal distribution with

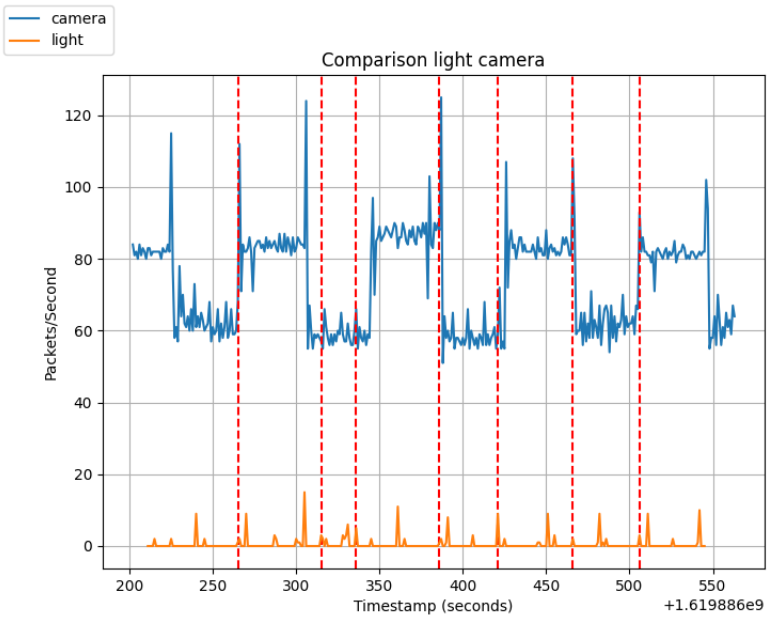


Figure 3.1: Network traffic of camera and light device when light is repeatedly switched on or off

the larger (smaller) mode corresponding to the camera filming a lit (dark) room. A bimodal distribution is a distribution that has two different modes. In our case, it is the combination of two other normal distributions. This bimodal distribution is determined by the traffic rate's mean that corresponds to a dark room and the traffic rate's mean corresponding to a lit room. Moreover, this bimodal distribution has two variances. These two variances correspond to the distributions of each room state (dark or lit).

Consequently, when the light in the room is turned on or off, this will cause a sudden change in the traffic rate of the camera in the same room, as shown in Figure 3.1. In addition, since we are using a smart light, whenever we turn it on or off, the light device itself will send or receive a certain number of network packets. Between those actions, no or nearly no traffic is transmitted to or from the light. However, as can be seen in Figure 3.1, the changes in the network traffic of the camera and the smart light are not perfectly synchronized due to delays in the network and the involved IoT services.

Our algorithm to determine whether the smart light and the camera are linked (in terms of cause and effect) is depicted in Figure 3.2. The algorithm takes as input the traffic rate timeseries of the camera and the light. The algorithm first finds all timestamps $t_1, t_2, \ldots$ with a network activity of the smart light (step 1). To reduce the number of occurrences, only one per 4-second time window will be

counted. For each of those timestamps t_i, we compare the traffic of the camera in the window $[t_i - T, t_i]$ to the traffic in the window $[t_i, t_i + T]$ (step 2). The general idea is the following: If the average traffic rate in the two windows is different, it means that the camera is influenced by the light and we conclude that both devices are in the same room. However, as visible in Figure 3.1, the observed traffic rates are disturbed by noise caused by, among others, fluctuations in the network link quality. Therefore, instead of directly comparing the traffic rates, the algorithm performs a hypothesis test where the null hypothesis is that the traffic rates in the two windows are identical. To calculate the P-value, the algorithm uses Welch's t-test [18] ($\alpha = 0.05$). This test is an adaptation of the student test for the comparison of means of distributions with different variances. If the null hypothesis is rejected the algorithm assumes that the camera's behavior has changed. To perform this statistical test the algorithm use the SciPy library [1] .

In practice, basing our decision on the windows $[t_i - T, t_i]$ and $[t_i, t_i + T]$ of a single timestamp t_i would not be very reliable. The risk of false detection of a change in the camera's data traffic is too high. To make the detection more robust, our algorithm keeps track of the observed bimodal distributions of the camera's traffic at the timestamps $t_1, t_2, \ldots$ and only decides that the light and the camera are linked if those distributions are consistent over several timestamps. This happens in steps 3 to 9 in Figure 3.1. For the first timestamp, the observed bimodal distribution is just stored and a "consistency coefficient" variable is set to 0 (steps 3 and 4). For a subsequent timestamp t_j, we test whether the bimodal distribution observed in the windows $[t_j - T, t_j]$ and $[t_j, t_j + T]$ is identical to the one observed so far for the previous timestamps (step 5). Again, we use Welch's t-test. If they are identical, the distributions are merged and the coefficient variable is incremented by one (step 6).

The merge between the distribution is done with a merge between the standard deviation and the mean.

We get the merged mean x with :

$$x = \frac{n_1 * x_1 + n_2 * x_2}{n_1 + n_2}$$

We get the merged standard deviation with :

$$S = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1) * s_2^2 + n_1(x_1 - x)^2 + n_2(x_2 - x)^2}{n_1 + n_2 - 1}}$$

Where n_1 and n_2 are respectively the number of data of the distribution 1 and distribution 2, x_1 and x_2 the distribution of the mean of the distribution 1 and 2. s_1 and s_2 the distribution standard deviation of the distribution 1 and 2. And then

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

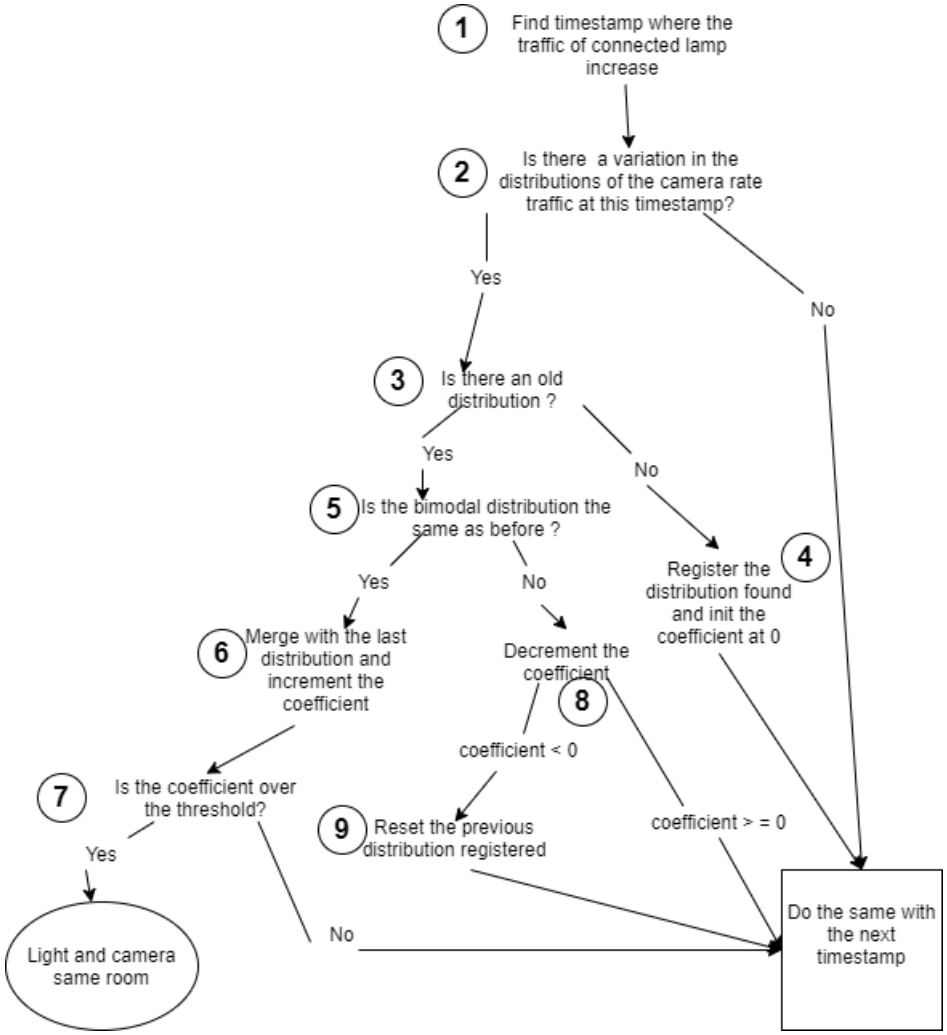


Figure 3.2: Steps of algorithm to determine colocation of camera and smart light

we get the new normal distribution. With the merge of the mean and the standard evaluation we determine the new normal distribution.

To continue with the algorithm, if it reaches a certain threshold θ (step 7), the algorithm concludes that the light and the camera are in the same room. However, if the bimodal distributions are not identical, the coefficient is decremented by one (step 8), and if the new observations are too different from the old ones, the history is completely deleted (step 9).

The value of the threshold θ controls how often the algorithm must see a change in the traffic rate of the camera before it decides that the camera's behavior is linked to the light. In Figure 3.1, the dashed vertical lines represent the timestamps when the algorithm reaches step 6, i.e. when the coefficient is incremented. Note

that the first observed traffic rate change at timestamp 220 does not increment the coefficient (step 4).

3.3 Multiple cameras or lights

The algorithm described in the previous section can be extended to Smart Homes with multiple cameras or lights in the same room. To decide whether two cameras are in the same room, we check whether the detected traffic rate changes happen at (more or less) the same time.

On the other hand, if the algorithm does not find any light that could explain a camera's behavior, this either means that (a) there is no artificial light near the camera (e.g. because the camera is outside), (b) there is an artificial light source but it is not a smart light, or (c) there is a smart light but its network traffic cannot be observed. Case (a) and (b) can be distinguished by looking for the presence of the typical bimodal distribution in the camera's traffic. To confirm/exclude case (c), the attacker would have to extend the network monitoring to other network types (e.g. the camera uses WiFi, while the light connects to a wired Zigbee hub) to make sure that no traffic is missed.

3.4 Cameras with auto infrared light

3.4.1 Justification to have an approach that works with infrared

Many IP cameras have built-in infrared illumination. Typically, the infrared light is automatically switched on when the room becomes too dark. We tested our algorithm from Section 3.2 with two cameras and discovered that the network traffic of one of them, a D-Link camera, does not show anymore a bimodal distribution when its automatic infrared mode is activated. Instead, it shows two distinct traffic patterns of around five seconds duration when the smart light in the room is switched on or off (see Figure 3.3).

3.4.2 Description of our approach

To detect these patterns in the traffic we use a sequence search. A similar approach has been followed by Fu *et al.* to recognize activities of smartphone applications. First, we take the traffic rate timeseries of the camera containing the patterns of interest. Then, we compute a new timeseries with the rate change, i.e. the value at time t is the difference between the traffic rate at time t and at time $t-1$. We

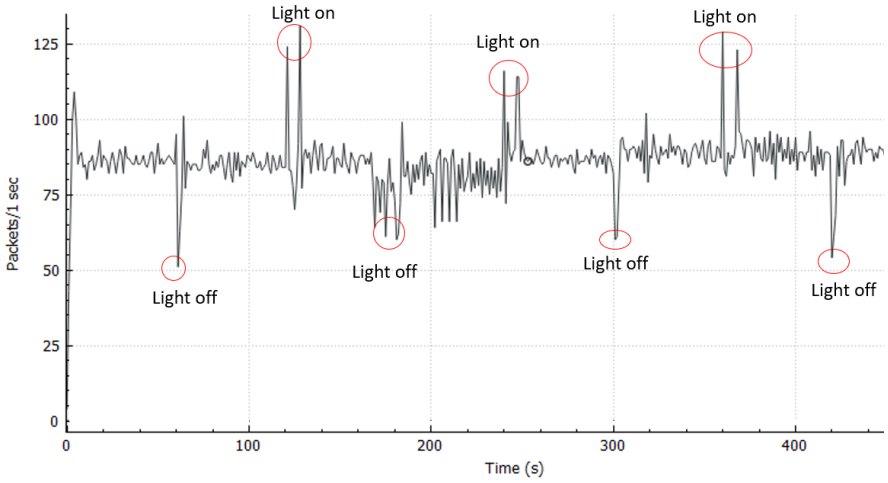


Figure 3.3: D-Link camera traffic with automatic infrared night vision

normalize and quantize the values in the new timeseries to a smaller set of labels, e.g. rate changes between 0 and 10 packets/s will get the label '1', rate changes between 11 and 20 packets/s get the label '2' and so on. Finally, we apply the Spade algorithms [19] for frequent sequence mining to the series of labels. Once the sequences representing the light-on/off patterns have been identified, we can replace step 2 in our algorithm by a comparison of the observed traffic with the identified sequences. As before, we deduce that the camera and the smart light are in the same room if the behavior of the camera matches the network activity of the light.

3.4.3 Detailed description of the spade algorithm used in the approach

The Spade algorithm is an algorithm for data mining that searches for label sequences by keeping the order and finding their frequency between the different transactions. Here is an example in Table 3.1. Before running the Spade algorithm, our program divides the dataset into multiple transactions. As for example what is indicated in Table 3.1.

Alsoo, before running the Spade algorithm, we decide a minimal "support". The support is the number of times we find a sequence in the transaction. For the example illustrate in Table 3.1 , we chose a support of two. This choice depends on the size of the dataset and the number of times that we repeat the pattern. In this example, the sequence [3,-2,1] is present in "Transaction 1" and "Transaction 2". "Transaction 2" is accepted because this is the order. The "Transaction 3" doesn't have the value "3" and "Transaction 4" doesn't have the correct order. Then the

Table 3.1: Example of transaction

Transaction number	Items
1	[3, -2, 1]
2	[3, -2, 0, 0, 0, 1]
3	[-2, 1, 0, 1,5]
4	[-2,3, 1, 0, 1]

algorithms return "[3,-2,1] 2" (the "2" at the end is the support of this sequence). To search the frequency of the sequence efficiently, the algorithm uses a depth-first search and creates the node of the tree during the runtime. That also allows us to prune and only look at the potential candidate. In our example, the label "5" has a support of 1 (below our threshold 2). Then it doesn't have the sense to search for the sequence "51" since "5" has only support of 1 then the support of "51" will never be over 1.

In addition to the rules on the minimal support that the sequences must have, we have added additional specific constraints to the problem we are trying to solve. We do this to have the least amount of irrelevant sequences.

- **Beginning with label "0":** First sequences never start with the label "0". The label "0" is assigned when there is no significant variation and knowing that there was no variation before is not interesting.
- **Length:** The pattern has a duration of around five seconds, so we reject all sequences longer than eight and sequences with a length of less than three.
- **Negative label:** The pattern always has ups and downs, so the sequences must have at least one negative label.

Finally, with these constraints, on our training dataset of 9 minutes with a pattern repeated every 30 seconds, the Spade algorithm return around fifty potential sequences. It would have been possible to create an algorithm to determine which of these sequences is the most relevant to recognize the pattern. In our case, we select this most relevant sequence without algorithm but with a method described in Section 3.4.4.

3.4.4 Method to select the most relevant sequences

In Figure 3.3 we observe that the when we turn on the light the traffic rate increase, decrease, and increase again. Moreover, we observe symmetry between the two increases. Then, we should have a sequence that more or less symmetric and that represents these increases. We can have similar reasoning to find a pattern when we

turn off the light. Moreover, to recognize the pattern ideally is to have a "strong" sequence. For example a small sequence like [3,-2,1] should also accept the sequence [3, -2, 0, 0, 0, 1]. And then if the sequence [3, -2, 0, 0, 0, 1] have also high support, it should be great to use this less general sequence. For the experiment part, in Chapter 5, we don't choose too strong a pattern to be able to recognize them also when the connection is less stable. Indeed more the network is unstable, the more the pattern is more modified. There is a trade-off between the possibility to choose a strong pattern and reduce the false positive or have more false-positive with more recognition.

3.4.5 Disadvantage of this approach

While this approach allows us to analyze the traffic of cameras that do not show the bimodal traffic distribution, it has the disadvantage that it requires a training dataset from which the sequences can be extracted. The sequences are camera specific, which means that not only do we have to prepare a separate dataset for each camera model, we also have to correctly identify the camera model from the Smart Home traffic in order to know which sequence set to use. Fortunately, the latter is not a big hurdle given the existing literature on how to do accurate device fingerprinting (see Chapter 2).

3.5 Multiple cameras and human movement

3.5.1 The CUSUM approach

The goal of the following technique is to determine whether and how two cameras film the same human movement. To this end, we use the CUSUM formula by Cheng et al [4]. It is part of a sequential analysis technique used for monitoring change detection. Since we are looking for an increase in traffic (movements increase the amount of traffic sent by a camera), we need to calculate the upper bound S_n of the CUSUM at time n:

$$S_0 = 0, \qquad S_n = max(0, S_{n-1} + x_n - w_n)$$

where w_n is the mean of the traffic rate, S_{n-1} is the CUSUM value at time $n-1$, and x_n is the current traffic rate at time n. In our code, we implement a modified version $S_n = max(0, S_{n-1} + x_n - w_n - \frac{k}{2})$ where k is the standard deviation of the traffic rate. This additional term allows the CUSUM value to decrease faster and stay at 0 when there is no significant increase in the traffic rate. With this technique we can rapidly detect if someone moves in front of a camera.

3.5.2 Application of the CUSUM

In Figure 3.6, we show the positive bound of the CUSUM calculated for two cameras that film the same person who walks back and forth in an apartment. The filled rectangles at the top of the figure represent the time periods when the CUSUM values of a camera exceeds a certain threshold (that we have set close to zero since we want to detect all movements). The fact that those time periods of movement of camera 1 do not significantly overlap with those of camera 2 indicates that the two cameras are not filming the same scene. In fact, both cameras were placed by us in different rooms for this experiment. Conversely, a high degree of overlap would indicate that the fields of view of the two cameras overlap, and the higher the overlap the more likely the two cameras are near each other or even located in the same room. For the experiment in Figure 3.6, the overlap is only 7 percent.

3.5.3 Estimation of the distance between two cameras

We can estimate the "distance" between the two cameras by calculating the time lag between the two CUSUM timeseries. Note that this distance does not directly represent the physical distance between the positions of the cameras. Instead, it represents the amount of overlap of their fields of view. If it is zero, it indicates that the cameras are monitoring more or less the same part of the room, although not necessarily from the same direction. A lag close to the width of a peak in the CUSUM curve means that the cameras monitor adjacent rooms or adjacent parts of a room.

3.5.4 About the lag and the cross-correlation

We estimate the lag by calculating the cross-correlation of the two CUSUM timeseries. The lag corresponds to the time by which we have to shift one of the timeseries to maximize the cross-correlation. The Figure 3.4 (source [2])) illustrate how works the cross-correlation. Intuitively the cross-correlation is the total area in common between the two CUSUM timeseries. And we calculate this cross-correlation for each shift.

Mathematically it corresponds to :

$$(g * f)[n] = \sum_{m=-\infty}^{+\infty} g[m] f[m+n]$$

Where $g * f$ are discrete functions, n the shift value, and m the coefficient to calculate the area between the two function for a shift n.

[2]https://en.wikipedia.org/wiki/Cross-correlation

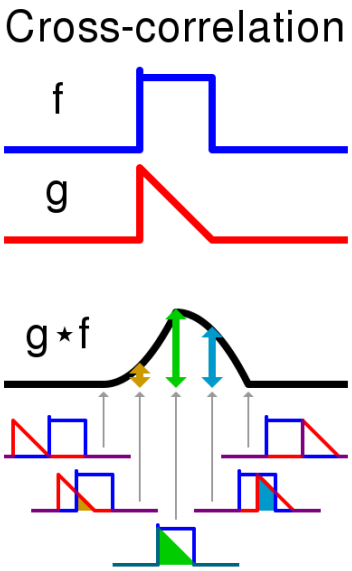


Figure 3.4: Cross correlation between function f and g

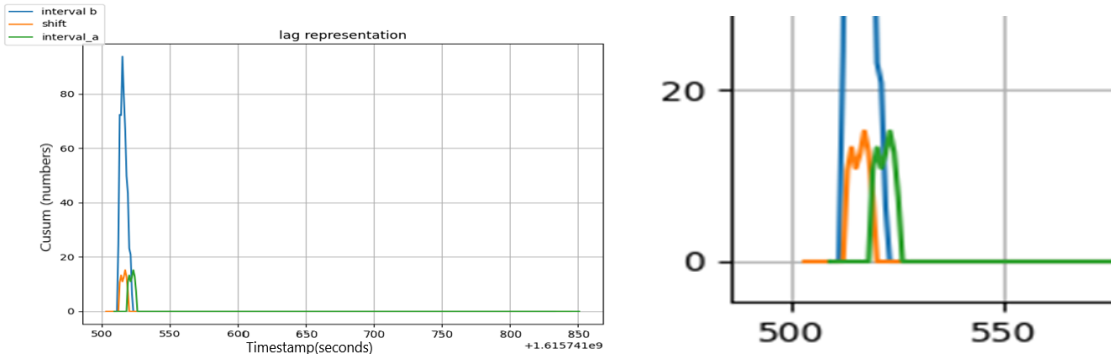
In our case, the lag corresponds to the n that maximizes $(g * f)[n]$. In practice we don't use the $[m = -\infty; m = +\infty]$. Since our experience is limited in time. And also because we can recognize the increase in the CUSUM curve and isolate these increases inside an interval. The algorithm records a list of intervals corresponding to one increase to isolate the increases in the CUSUM curve. As for the movement, we consider that there is a rise when we are above the threshold. When we have these isolated increases, we can apply cross-correlation. Here is an example to illustrate the use of the cross-correlation between this list of curves. In other words, the time intervals correspond to the domains of the CUSUM function.

Example :

- **List time intervals "camera 1" :** $[(t1; t2), (t3; t4), (t5; t6), (t7; t8)]$
- **List time intervals "camera 2" :** $[(s1; s2), (s3; s4), (s5; s6), (s7; s8)]$

For example, we chose "camera 1" to determine how the CUSUM curve is shifted compared to "camera 2". We take the interval $(t3; t4)$. We calculate the cross-correlation between all the curves of CUSUM isolated by the time intervals domain defined for "camera 2". And we save the negative lag and the positive lag closest to 0. We save a positive and negative lag because we don't know in advance which camera has detected the movement first, and sometimes the person turns around, which changes the sign of the lag. Section 4.3.3 gives more details on the structure used to return these results. Figure 3.5 illustrates one increase isolated from "camera 1", one increase isolated from the camera, and the optimal shift for

"camera 2". The "interval_b" correspond $(t3; t4)$. The "interval_a" corresponds to one of the interval lists of the "camera 2". And we see how the isolated increase of the CUSUM has shifted to maximize the cross-correlation value.



(a) The two CUSUM curves isolated and the shift (b) Zoom

Figure 3.5: Optimal shift thanks to the cross-correlation

3.6 Camera and traditional lamp

It is impossible with the approaches described above to determine if a non-intelligent lamp is in the same room as a camera. But we have found a relatively simple way to do this by combining the CUSUM approach described in Section 3.5.1 and the algorithm presented in Section 3.3. Indeed, whether the lamp is not connected does not change the sensitivity of the camera to light. But we need to find a way to replace the information given to us by the intelligent lamp.

Since the CUSUM allow us to detect when the traffic increase. Then, we replace the smart lamp's traffic rate increase (that the connected lamp would have given us) with the increase of the CUSUM caused by a change of distribution due to a rise of luminosity.

Moreover, with light, the traffic rate of the camera can decrease (when we turned off the lamp), and then, to detect this, it is necessary to calculate the lower bound of the CUSUM.

The lower bound S_n of the CUSUM at time n:

$$S_0 = 0, \qquad S_n = max(0, S_{n-1} - x_n + w_n)$$

And In our code, the modified version $S_n = max(0, S_{n-1} - x_n + w_n - \frac{k}{2})$

Every time we detect a rise of the CUSUM, we will look (as for the connected lamp in Section 3.2 for step 1) to which Timestamps happened. This method

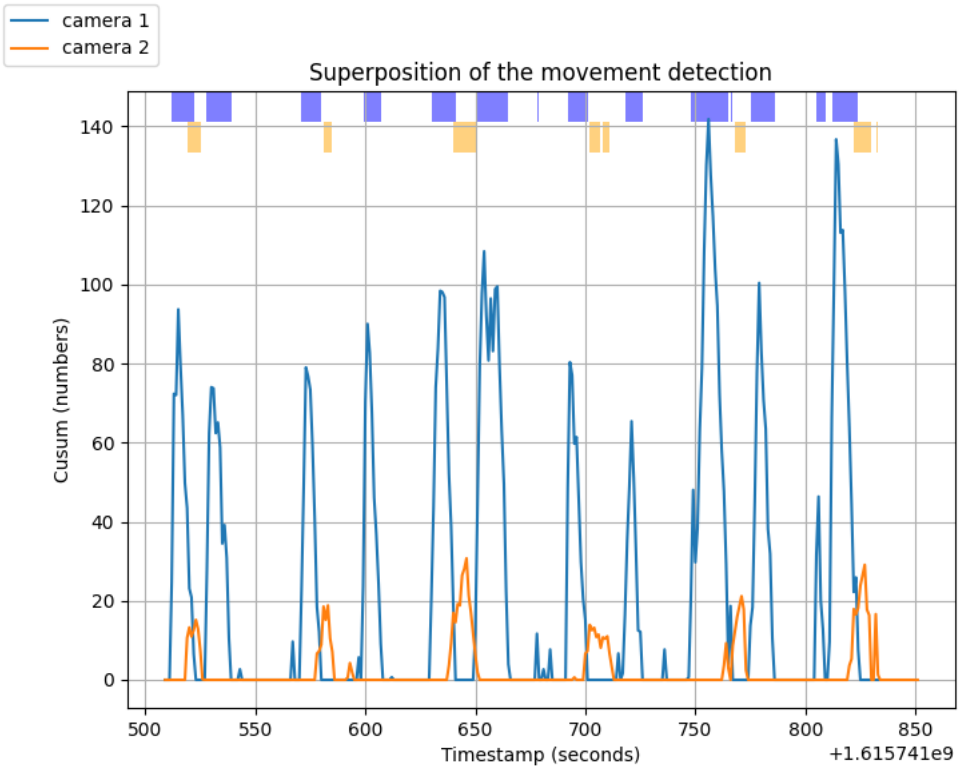


Figure 3.6: Crossing room A, reaching room B, and coming back in the opposite direction. Repeated six times.

replaces step 1 described in Figure 3.2 allows us to reuse all the other steps of the algorithm.

This approach shows that we are not limited to a few devices and that it is not difficult to combine these approaches for other situations.

Chapter 4

Implementation and tools used

This chapter present the basic principles of the code by describing the design choice of the program. It also briefly describe the tools used. Some tools help us to generate and collect the data that have been analyzed by our implementation. Finally, it end by presenting the weaknesses of our implementation that could be improved. For more details, the code is available in the GitHub repository. [20].

4.1 Tools used

- **Python 3.9:** The version of python used for our implementation. Our code was running on a Windows 10 computer.
- **Wireshark[1]:** Tool to analyze the network. We use this software to collect the traffic data and generate the pcap file that our program analyses.
- **BlueStacks 4[2]:** It's an android emulator. We had to use an emulator because when the android camera application was not running, the camera was not transmitting any traffic. Moreover, as we had only one smartphone, we could only run one IoT device at once. We chose Bluestack, first of all, because other emulators like "GenyMotion"; "Android Emulator" we could not install the applications IoT from the play store. Moreover, with BlueStack, it was easier to create several instances.
- **IoT android applications:**
  - **Mi Home[3]:** for the Xiaomi camera;

[1]https://www.wireshark.org/
[2]https://www.bluestacks.com/fr/index.html
[3]https://play.google.com/store/apps/details?id=com.xiaomi.smarthome

– **mydlink**[4]**:** for the D-Link camera;

– **Philips Hue**[5]**:** for the smart lamps;

4.2 Important libraries used in our implementation

- **Matplotlib:** [6] A python library for graphical representation. We use it in our algorithm to realize a graphic representation of the traffic rate, the pattern, CUSUM curves...
- **Scipy:**[7] A python library with method for statistics, numerical integration... We used it to realize the statistical analysis like Welch's t-test, the cross-correlation, the lag calculation.
- **Numpy:**[8] A python library to manipulate array or matrix. We often used arrays then this library was useful and simplified the code.
- **Scapy:**[9] Scapy is mostly known for its software to manipulate packets. Also, this project has a complete library that allowed us to read, filtering removing the payloads from the TCP/UDP packets recorded in the pcap files the Scapy.

4.3 Design choice in the implementation

Figure 4.1 indicates different parts to use our implementation. Part 1 indicates what we must do before running the program (see Section 4.3.1). Part 2 when the algorithm filtering the data and give them a good format (see Section 4.3.2). Part 3 indicates the different algorithms used for the analysis (see Section 4.3.3). Finally, part 4 is how the results have returned by our implementation (also described in Section 4.3.3).

4.3.1 Data to run the program

First, we need a pcap file with the data registered in the smart house to run the implementation. Since in our approach, the payloads of the packets are useless,

[4]https://play.google.com/store/apps/details?id=com.dlink.mydlinkunified
[5]https://play.google.com/store/apps/details?id=com.philips.lighting.hue2
[6]https://matplotlib.org/
[7]https://www.scipy.org/scipylib/index.html
[8]https://numpy.org/
[9]https://pypi.org/project/scapy/

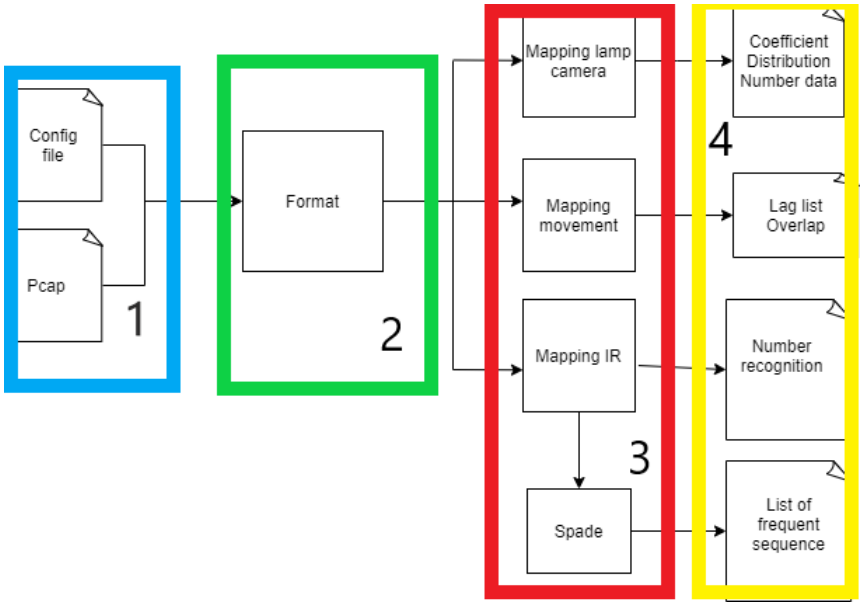


Figure 4.1: Design choice

we can remove them to make the file read faster and take less storage. Removing payloads is an option implemented in our program (with the Scapy library).

Moreover, we also need to set up a configuration files. The goal of this file is to give parameters to the program. These parameters are used to filter the traffic that corresponds to each device.In this file, we need to specify the source and destination IP address and the protocol. There is a configuration file for each IoT where specifies these parameters.

4.3.2 Filtering and format of the data

Before recognizing the pattern, the data in the pcap file is filtered (by source, destination, protocol...). Our program, after filtering creates a tuple list for each device entered in the configuration file with the number of packets for each timestamp. For example the list of tuples for the smart lamp should be :

$$[(10, 2), (17, 3), (21, 2)]$$

Where 10 is the timestamp and 2 is the number of packets received. Since the "packets per second", in our approach, is the metric used to recognize the patterns then we create a tuple for each timestamp. This list of tuples will then be used and analyzed to recognize patterns and find a mapping.

4.3.3 The pattern recognition

The choice of type of pattern is done manually in the code (by calling the corresponding method in the implementation) but it could have been integrated as an option in the configuration file. This part describes how we implemented the approaches described in Chapter 3. For the pattern recognition (Part 3 in Figure 4.1), three large parts correspond to three different types of mapping: a part that concerns the one with the lamp connected and the camera with the analysis of distribution, the second with a mapping related to the movement between the different cameras, and the last to recognize the mapping related to the infrared.

- **The mapping with the connected light:** The program receives the lists of tuples corresponding to the cameras (our program works with only two list because we did not have more cameras) and a list of tuples that corresponding to the traffic of the connected lamp. After the calculations, the program returns to the terminal :
  - **The distributions:** The mean, the variance, the number of data of the two distributions.
  - **Graphical representation** : It indicates the recognition of the patterns as you can see in the figures in the Chapters 3, Chapter 5.
  - **The coefficient:** The value of our approach explained in Section 3.2
  - **Debug mode:** It is also possible to activate a debug mode to see the calculated values, when the distributions are not accepted, etc.
- **The mapping movement with camera:** The algorithm receives two lists of tuples (explanation about tuples in Section 4.3.2) that correspond to the camera traffic. The algorithms will create two new lists of tuple with the timestamps and the CUSUM value. Next, we compare these two lists and calculate the lag. The program return in the terminal the percentage of overlap between the two CUSUM curves. For the lag, we use a hashmap where the key corresponds to the interval for which we will evaluate the shift. The value corresponding to the key are two tuples that contain the interval. For example, the hashmap :

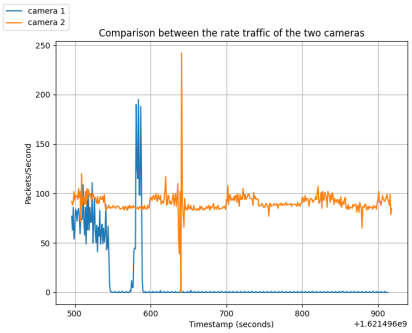
  $$\{(519, 525) : (((528, 539), -10), ((512, 522), 6))\}$$

  There is the key $(519, 525)$ that represents the CUSUM interval time of camera one. The tuple $((528, 539), -10)$ indicate the interval time CUSUM of the second camera with $(528, 539)$ and -10 is the lag calculate with the cross-correlation between the interval $(519, 525)$ and the interval (528, 539) of the second camera. For the time interval, $((512, 522), 6))$ this the same as before the time interval is before the time interval of the key.

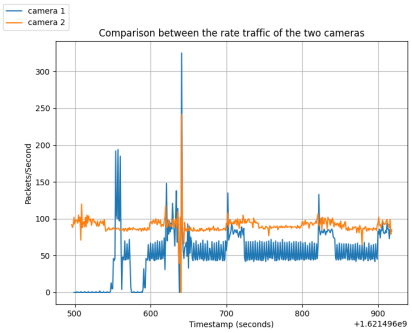
- **The auto infrared recognition:** this algorithm is divided into two parts.
  - **The training dataset:** We receive a dataset, and we mining the frequent interval thanks to the implementation of the Spade algorithm [19] and save the frequent sequences in a file.
  - **Searching the sequence:** This part consists of searching for an interval candidate that would have been found in the first part. For this second part, the algorithm returns a list of timestamps where it has recognized the sequence.

4.4 Limitation in the implementation

- **Time to read pcap file:** Each time an analysis is executed, the program rereads the entire file to be analyzed. Parts 1 and 2 are repeated each time. This step is the part that takes the longest time. For example, an experiment of 7 minutes with two IP cameras takes around 3 minutes for part 2 in Figure. If we were to use this program for experiments lasting several hours with many more algorithms, it would be very long. It would have been necessary for the conception to separate parts 1 and 2 from the rest of the program because it takes a lot of time to read pcap files. Or in place work with CSV files that can be generated with Wireshark instead of directly using a raw pcap.
- **Unique IP destination server:** Sometimes the Xiaomi camera redirects its packet flow to another server than the original one. This case is not frequent. It happened about three times on all the measurements taken with the Xiaomi camera. More precisely, the Xiaomi camera sends control packets to several servers and sends the video stream to one unique server. Sometimes the camera can change the server while we are watching in real-time. Since this change is rare, it has not been taken into account in our algorithm. Indeed our algorithm considers that an IoT object has only one source IP address and one destination IP address. It should therefore be possible to assign several IP addresses for an IoT. Figure 4.2a illustrate the first server that sends the stream at the beginning and finally redirects the traffic and sends some packets controls. Figure 4.2b represents the traffic rate of the server that receives the redirected traffic.



(a) Xiaomi with begining server



(b) Xiaomi with new server

Figure 4.2: Limitation

Chapter 5

Experimental results

In this chapter, we validate the techniques presented in Chapter 3 and present the experimental results.

5.1 Setup

The setup for the experiments is shown in Figure 5.2. some pictures of the setup are shown in Figure 5.1 . Two cameras are connected to a router/WiFi-AP (router 2). Two smart lamps communicate over Zigbee with a Hue bridge wire-connected to the same router. Router 1 provides Internet connectivity, so that the IoT devices can reach their services deployed in the cloud. The cameras stream their videos to apps running on a smartphone connected to router 1. Between router 1 and router 2, we have placed a notebook that records the incoming and outgoing WiFi traffic. The following list contains the used hardware:

- **D-Link Mini HD Wi-Fi dcs-8000lhe:** WiFi camera with sound, infrared night vision (see Figure 5.1);
- **Xiaomi Mi Home Security Camera 360° 1080p:** WiFi camera with sound sensors, infrared night vision, loudspeaker, motor for rotation (see Figure 5.1);
- **Hue bridge:** Zigbee bridge for the lamps (see Figure 5.1);
- **Philips Hue bulb E27:** connected lamp whose brightness can be controlled by a smartphone application (see Figure 5.1);
- **Lenovo Ideapad 320-15IKB** with Windows 10;
- **Technicolor TG799TSvn V2** router.

(a) Xiaomi Camera (b) D-Link camera

(c) Philips Hue bulb (d) Hue Bridge

Figure 5.1: Setup

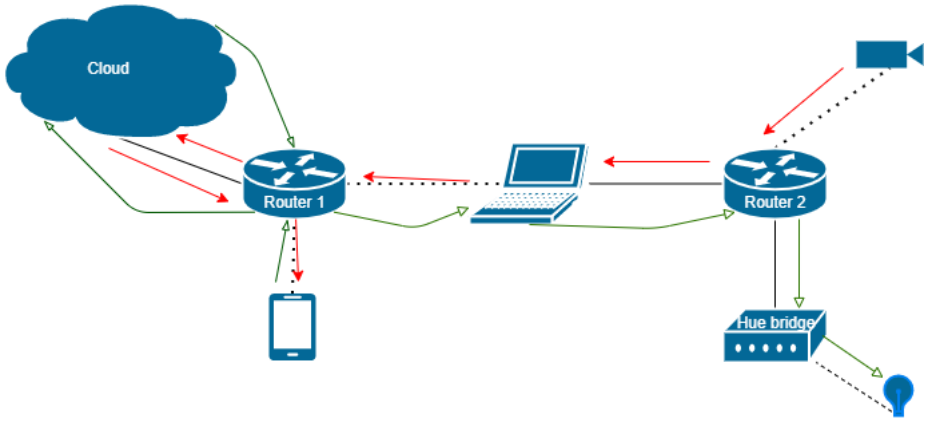
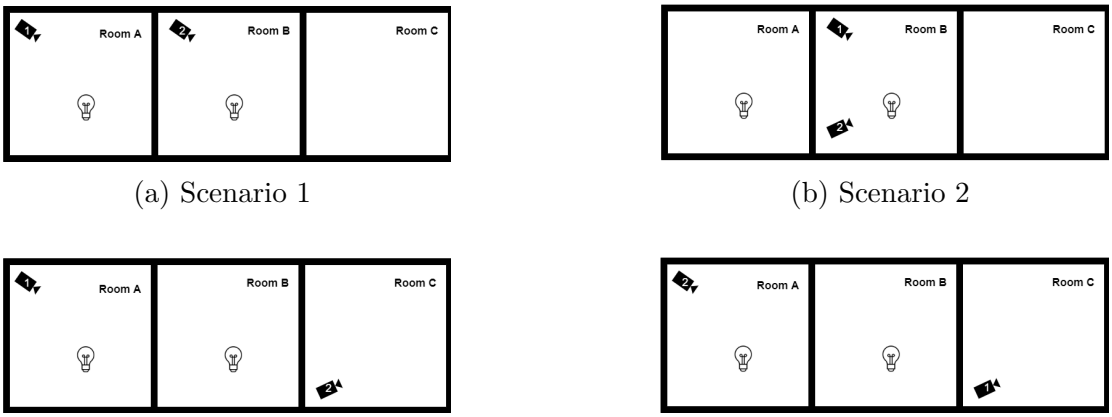


Figure 5.2: Configuration to collect the packets data of the IoT devices

The traffic generated by the two cameras takes different paths depending on the camera model. The D-Link camera sends its video stream directly to the smartphone. In contrast, the video stream of the Xiaomi camera is first sent to the cloud and then returns to router 1 to finally reach the smartphone. When we control the lamp with the phone the traffic passes through router 1 and router 2. Note that all traffic is visible to the notebook placed between the two routers.

5.2 Lamps and cameras without infrared mode



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Figure 5.3: Scenarios

To evaluate the performance of the algorithm from Section 3.2, we test four scenarios without natural light where we place the devices in three different rooms

Table 5.1: Results after step 1 and 2

	Location 1		Location 2	
	Xiaomi	D-Link	Xiaomi	D-Link
Accuracy	0.56	0.71	0.5	0.73
Precision	0.78	0.77	0.88	0.86
Recall	0.67	0.91	0.54	0.83
F1-Score	0.72	0.83	0.67	0.84

of a house (see Figure 5.3). The infrared night vision is not activated. Scenario 3 and 4 allow us to test the robustness of the detection in terms of false positives since room B only contains a lamp and room C only contains a camera. Figure 5.4 shows how we control the lamps during the experiments. For each scenario, we repeat the depicted control pattern for 6 minutes. We test all four scenarios in two different houses called 'Location 1' and 'Location 2' in the following.

We choose a window size of $T = 15s$. This is a compromise between detection latency and detection quality. A larger T means to wait longer before the algorithm can be executed. On the other hand, more data is available for the distribution estimation. Another limiting factor is the expected time between two light changes. In our experiment, we switch a light on or off after 40 seconds. Obviously, T should be shorter than that.

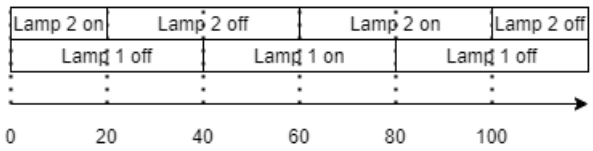


Figure 5.4: Time schedule (in seconds) for lamps turned on/off

When we use a relatively conservative threshold of $\theta = 4$, the algorithm correctly identifies without any errors, for all scenarios and locations, whether a specific lamp and camera are located in the same room.

To get a better insight into the performance of the algorithm, we have also analyzed the results of just step 1 and step 2 of the algorithm, i.e., what would happen if we decided whether a lamp and a camera are in the same room just based on the tests in step 1 and 2, without performing the additional tests in steps 3–9. Table 5.1 shows for each location and camera the results obtained from the four scenarios. We observe that especially the accuracy for the Xiaomi camera is low. In general, the traffic of this camera has a higher variability than the other camera, which makes the detection more difficult.

5.3 Auto infrared with sequence search method

We have repeated the same four scenarios with the automatic switch to infrared night vision activated. As already explain in Section 3.4, the Xiaomi camera keeps its binomial distribution and the results reported in the previous section do not change. For the D-Link camera, we first create a pcap trace of 9 minutes where we have a lamp and the camera in the same room in Location 1 and we turn on/off the light every 30 seconds. We run the sequence mining algorithm on the trace to identify the patterns that characterize the changes of the state of the camera. Table 5.2 shows the results. We observe that they are similar to the results obtained without infrared night vision. Like in the previous experiments, the results demonstrate that a single test is not sufficient; multiple tests and a consistency threshold are needed to obtain a reliable detection.

Figure 5.5 shows an example of the pattern recognition. The dashed vertical lines indicate the timestamps when the switch of the infrared mode has been detected.

Table 5.2: Results automatic switch to infrared night vision on the D-Link camera

	Location 1	Location 2
Accuracy	0.74	0.69
Precision	0.95	0.86
Recall	0.77	0.78
F1-Score	0.85	0.82

5.4 Human movements

To illustrate the general behavior of the cameras in the presence of movements, we show in Figure 5.6 the resulting traffic rate in a situation where someone walks in front of the two cameras for 20 seconds and repeats that every minute. As can be seen, both cameras react to the movement, but with different traffic rates. Figure 5.7 shows the corresponding CUSUM curves. The filled rectangles at the top of the figure visualize the movement detection by our algorithm. The more the rectangles of the two cameras overlap, the more likely it is that the two cameras are seeing the same scene.

Figure 5.8 and Figure 5.9 show the traffic rate and CUSUM curves obtained in a scenario where someone repeatedly crosses the field of view of the cameras at walking speed.

Table 5.3 shows the amount of overlap of the movement detection between the two cameras for different scenarios. And Figure 5.10 indicate the different position

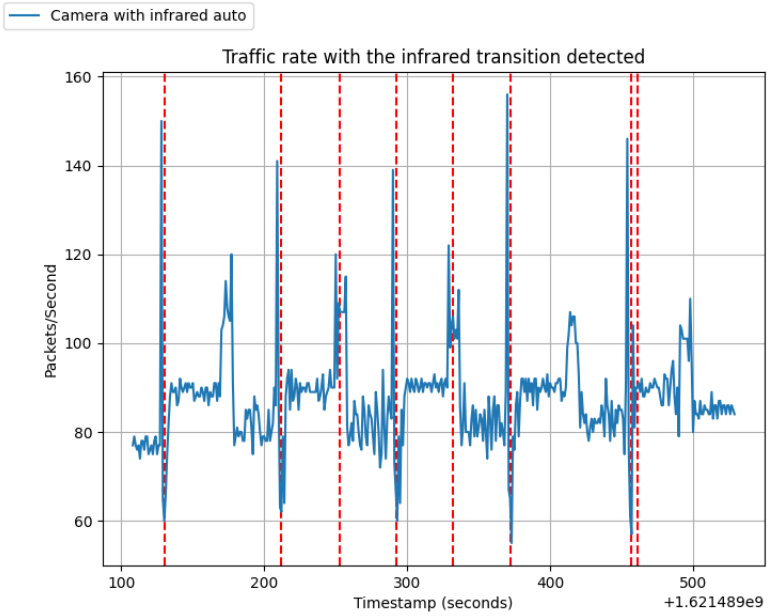


Figure 5.5: Infrared switch recognition

Table 5.3: Percentage of overlap between the two cameras

Scenario	Location 1	Location 2
Walking for 20s in front of the cameras	85%	94%
Crossing a room in front of the cameras	57%	82%
Crossing two rooms in front of the cameras	25%	35%

and orientation used in the house. Figure 5.10a indicate the configuration of the first and second scenario described in the Table 5.3 and the Figure 5.10b indicate the last scenario.

In the first scenario, both cameras are in the same room with the same field of view, and a person walks in front of them during 20 seconds. In the second scenario, both cameras are again in the same room and a person crosses their field of view during two to three seconds. In the last scenario, the cameras are in two adjacent rooms that are both crossed by the person. Each scenario was tested for 7 minutes with natural light. As expected, the shorter the movement, the smaller the overlap. If the cameras are in different rooms, the overlap is even smaller.

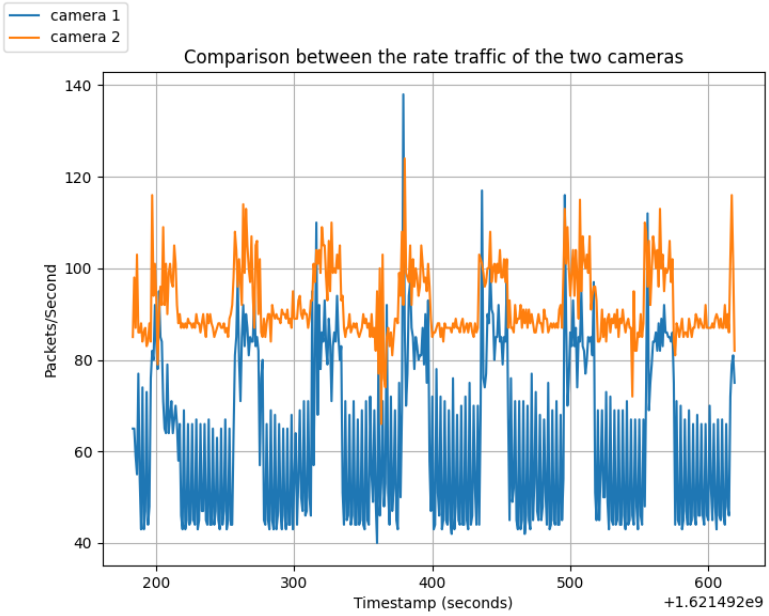


Figure 5.6: Traffic rate: Walking for 20 seconds in front of the camera

5.5 Orientation of the cameras

In this experiment, we let a person cross a room monitored by two cameras. Forty seconds later the person crosses the room in the opposite direction. The cameras have been oriented so that each one films a part of the room and that the two cameras together cover the entire room. The movement detection results are shown in Figure 5.12. We observe that one time out of two, one of the two camera detects the movement first. But we still get a large overlap of the movement detection regions of the two cameras. That was not the case when we had two rooms with one camera in each one. We can also numerically estimate the lag between the two cameras as described in Section 3.5. We calculate a mean positive lag of 2.8 seconds for the time periods when the person crosses the room in one direction and a mean negative lag of 3.2 when the person crosses the room in the other direction. This lag corresponds to the time it approximately takes to cross the part of the room only seen by one camera before being sighted by the other camera. Note that this method could also be applied to estimate the distance between two rooms if the cameras were located in different rooms. The Figure 5.11 indicate two example of configuration that we can use to film all the room. For the results we use configuration illustrate in the Figure 5.11b.

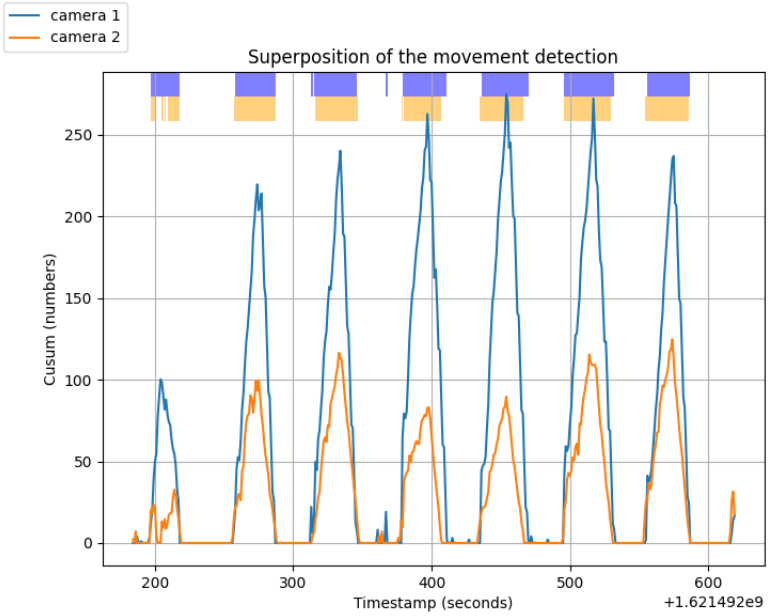


Figure 5.7: CUSUM: Walking for 20 seconds in front of the camera

5.6 Camera and traditional lamp

In the approach part, we discuss that we could also determine the bimodal distribution of an IP camera without a smart lamp. Since this is not directly related to the mapping between IoT devices, we just took an example. For 8 minutes, we turned on or off a smart lamp every 30 seconds. Then to compare with the algorithm that merges the two approaches, we reused the same data without taking into account the network information of the connected lamp.

The Figure 5.13 indicates with a dashed line when the algorithm recognizes a change in the distribution. For the algorithm that works on the connected light, we see in Figure 5.13b that we have more recognition than with the merged algorithm in the Figure 5.13a. These differences can be explained by the CUSUM that grows recursively and therefore does not increase instantaneously. This property of the CUSUM creates a slight delay in detection. This delay has an important impact given that we use a window size of $T = 15s$. For the smart lamp (illustrate in Figure 5.13) we had a threshold of 6 and the same data without the connected light information (illustrate in Figure 5.13a we had a threshold of 4.

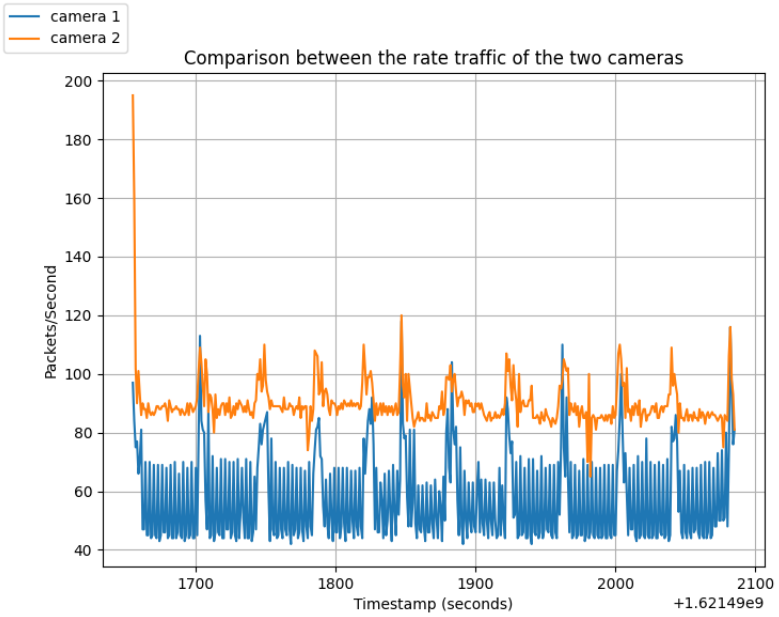


Figure 5.8: Traffic rate: Crossing a room

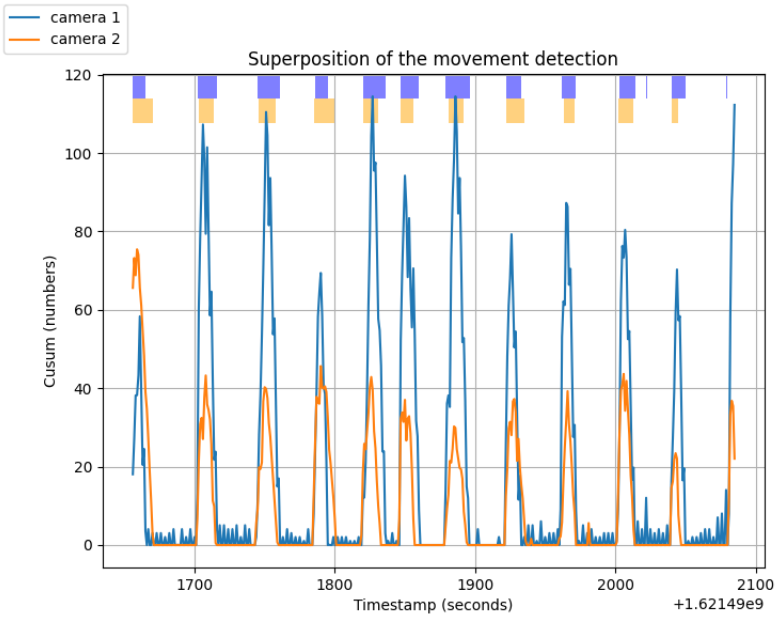
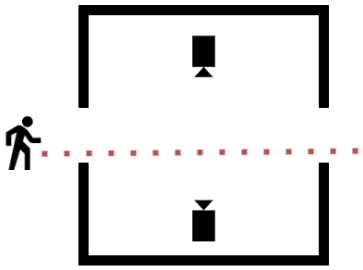
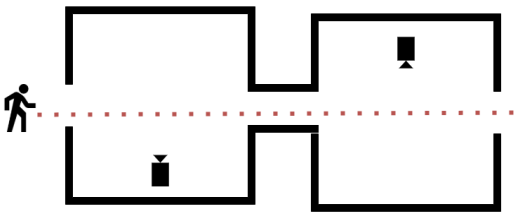


Figure 5.9: CUSUM: Crossing a room

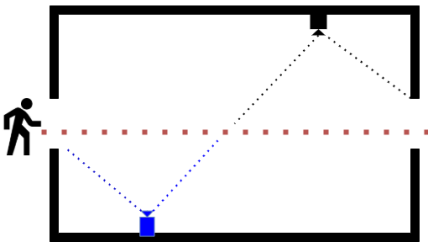


(a) Configuration for the two first scenarios

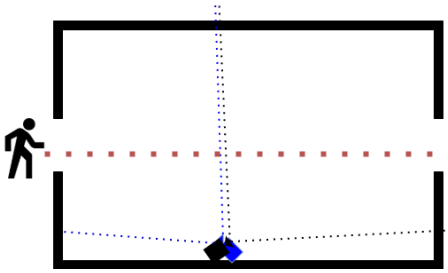


(b) Configuration for the last scenario

Figure 5.10: Positions and orientations of the cameras for human movement scenarios



(a) Camera with opposite orientation



(b) Camera with same position and different orientation camera

Figure 5.11: Positions and orientations of the cameras

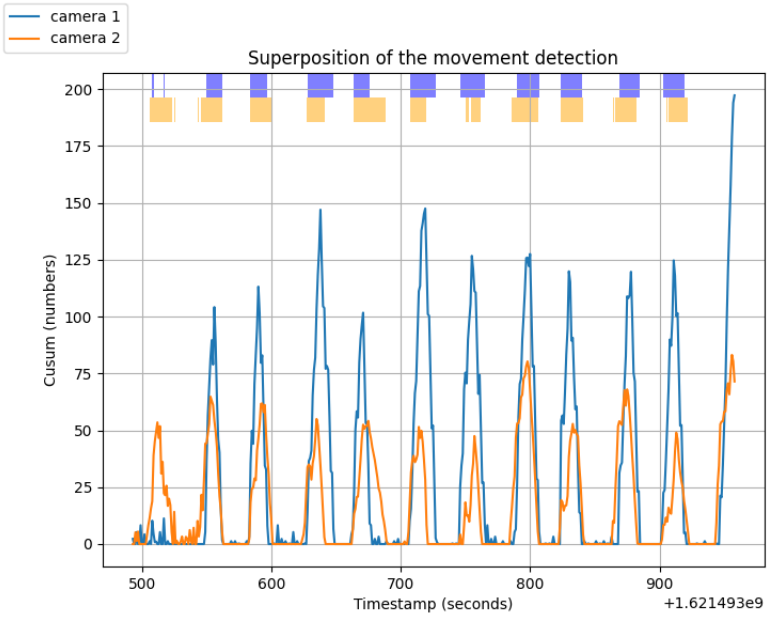
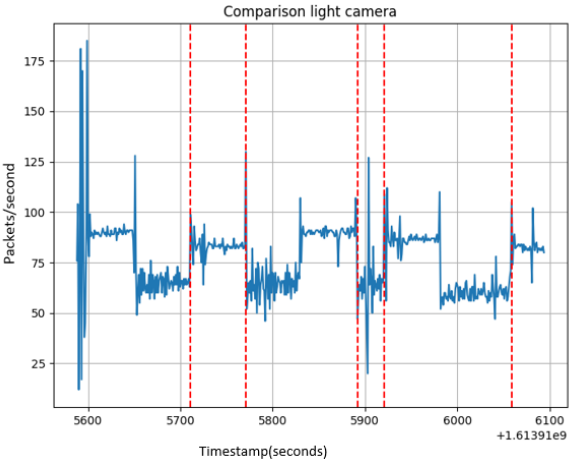
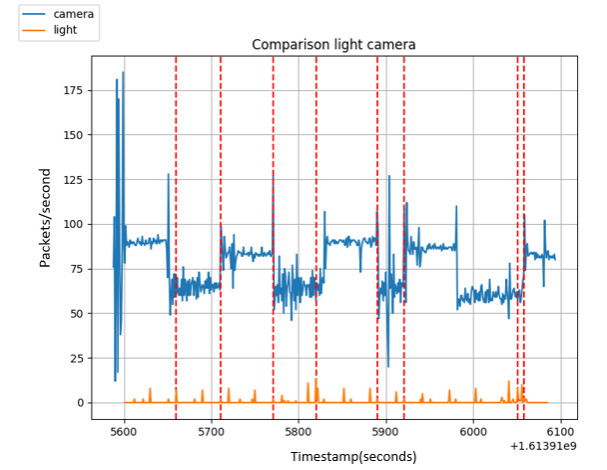


Figure 5.12: Crossing a room monitored by two cameras oriented in different directions



(a) Classical lamp



(b) Smart lamp

Figure 5.13: Comparison between classical and connected lamp

Chapter 6

Discussion and limitations

6.1 The stability of the network

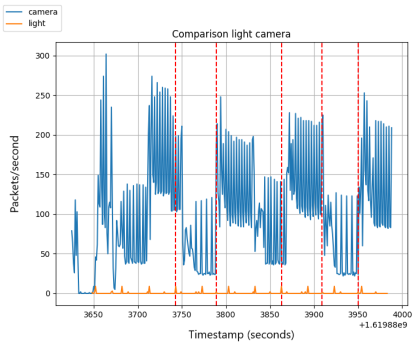
As already mentioned above, the stability of the network has a significant impact on the quality of the results. A bad connection will make it more difficult to recognize some patterns and the speed at which we could recognize them. For example, a sudden drop in the network bandwidth could cause a change in the distribution of the camera traffic that could be misinterpreted. Figure 6.1c illustrates the traffic rate of a camera during instability in the network. For this figure, the camera was not in the room with the smart lamp so there is no bimodal distribution. But the instability of the network creates distributions that could at first be registered by our algorithm. This is one of the reasons that we have created more than step 1 and step 2 in the Section 3.2.

The algorithm the most sensitive to the network quality is the infrared sequence recognition. In a stable network, we are even able to reliably distinguish the switching off of a lamp from when it is switched on. But since we cannot assume that we always have good network conditions, we have limited the presentation and discussion of the results in Section 5.3 to the question whether a state change (on to off or vice versa) can be reliably detected. The Figure 6.1d illustrates a period where the connection was unstable which created a lot of oscillation. Since our approach searches for a sequence, the probability of finding the sequence that matches the pattern increases strongly. The dashed line in the Figure 6.1d indicates all false positives.

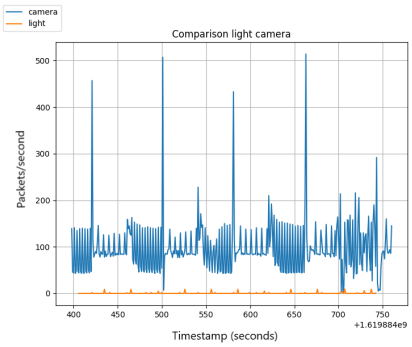
6.2 The sensitivity of the camera to the light

In our four scenarios with lights and lamps, we eliminated all other light sources, i.e. the rooms were almost completely black when the smart lamp was switched off.

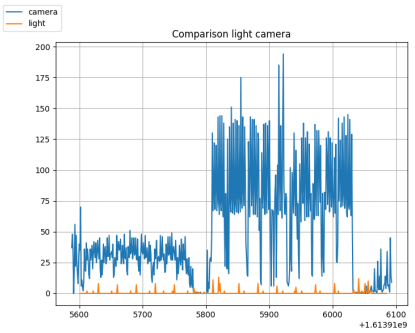
We have also tested less dark rooms and discovered that the different sensitivity of the cameras to low light conditions can become a challenge that leads to wrong results. Concretely, if one of the cameras has a higher sensitivity than the other, it will show greater traffic variations when light conditions change than the less sensitive camera. Our algorithm might not understand this situation correctly and conclude that the cameras are in different rooms. We have observed such problems with the Xiaomi camera and, indeed, it has a higher light sensitivity than the D-Link camera. Figure 6.1 illustrates the comparison between a dark room and a less dark room. For this scenario, a smart light was turned on or off every 40 seconds. As we can see, our algorithm detects no change in the distribution for the less dark room. That is because the statistical tests in our algorithm compare only the mean and we see graphically (on the Figure 6.1) that they are relatively close. We also see that the variances are not the same, so we could have improved our algorithm to also test the variances.



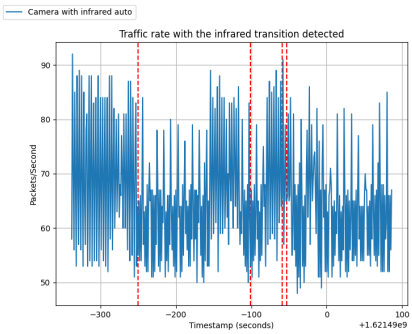
(a) Dark room (Xiaomi)



(b) Less dark room (Xiaomi)



(c) Bad connection for lamp mapping (Xiaomi)



(d) Bad connection for infrared detection (D-Link)

Figure 6.1: Limitations on the rate traffic

6.3 Number of people

Our analysis of traffic variations caused by human movements assumes that only one person is present. If the cameras film multiple people, wrong conclusions on the locations of the cameras are likely. Similarly, this will affect the estimation of the proximity of cameras using the cross-correlation method.

6.4 Limited set of hardware

Both our cameras use the UDP protocol. In addition, we have only tested one smart light brand. As future work, we have to validate our results with other protocols and devices. In reality, these devices could also use the TCP protocol but in some situations.

- **Xiaomi:** For this camera during the months of February-March-April, it camera was connecting most of the time in TCP and rare times in UDP. By the way, all our algorithms had been tested on the TCP traffic of the Xiaomi camera. But around May, when we were taking the experimental results, this camera was connecting only in UDP. Fortunately, since our approach uses properties that do not depend on the model of the device, we were able to obtain good results. And that is why we can think that this approach also has a lot of chances to work with TCP.
- **D-Link:** For this camera, we get TCP traffic only when the camera sent packets through the cloud. We found two ways for the camera to use the cloud. The first is to use the camera's website instead of the application [1]. And the second way is to use the android application in another network to make local connections impossible. We did not use these possibilities for two reasons. First, we could not have the real-time of the camera for more than 5 minutes without interruptions. Moreover, in practice, there were too many errors. With these errors, we rarely had more than two minutes without interruption of the connections.

[1] https://eu.mydlink.com/entrance

Chapter 7

Conclusion and future work

In this chapter we will conclude this work and then give ideas for future improvements.

7.1 Conclusion of thesis

Privacy is a major concern of Smart Home security. In this thesis, we have presented techniques that allow to infer the physical arrangement of smart devices in a Smart Home from passive network measurements. More specifically, we have presented algorithms to detect whether smart lamps and smart cameras are located in the same room. We have also shown how the relative position and orientation of multiple cameras can be estimated. Moreover we have obtained these results with an approach that works without depending too much on the different brands of IoT.

We have evaluated our techniques in different scenarios and shown that they provide results that give an insight into the physical organization of a Smart Home. We argue that such techniques can be useful for troubleshooting and management purposes, but could be also misused by malicious parties.

These results are interesting because they show that with simple actions like crossing a room, turning off a lamp, we can deduce the physical arrangement of different devices. Moreover, given the diversity of connected devices, the number of actions that can be done at home, there are still many things to discover on the subject. Finally as the connected objects will be more and more present in our daily life these questions of confidentiality will have to be taken more and more seriously.

7.2 Future work

- **Approach more accurate:** In this thesis, we use the number of packets per second as the metric. In the literature, some analyzed payload sizes, SYN/ACK packet proportions... These others metrics would also have given us other information.

  Moreover, We are not primarily interested in the traffic sent by the android applications but analyzing this traffic could also make the approach more accurate.

- **Sensitivity of the IP camera with light:** This propriety could be exploited to determine the intensity of luminosity in the room.

  Moreover, we could exploit this sensitivity with luminosity to determine which side of the house the cameras are present. Indeed the east side of a house would not have the same lighting as the west side. This difference could be seen in the traffic of IP cameras. E.g. if in the evening one group of cameras has its traffic dropping faster than another group. Then we could assume that the first group should be on the east side and the other group on the west side.

- **Stability of the network:** The stability of the network is a limitation of our approach. It would be interesting to estimate the stability of the networking on the measurements. E.g. with a good network, we distinguish for the automatic infrared of the D-Link when the lamp was switched off or on. Since a very stable network did not happen often, we detect when the smart lamp changed its state without distinction between on/off. But if we could estimate this stability, our algorithm could adapt itself according to this and find more patterns when the network allows it.

- **Device diversity:** To test the approach on other brands, other protocols. We could also think about more complex configurations than just two cameras and two smart lights. With more cameras or by adding new devices such as a computer that uses its camera in Skype.

- **Scenario:** Other scenarios could also have used. For the movement detection, we had limited crossing rooms. We could think of other activities such as working while sitting in front of the camera. Or scenarios with multiple people, pets that would not move in the same way as a human.

- **Stream:** Our approach with the cameras focused on the video stream. We could extend it to the audio stream of the cameras, the movement sensors.

- **Security:** Another big part is what concerns security, how to protect its privacy.

Bibliography

[1] A. Shoemaker, "How to identify a mirai-style ddos attack," https://www.imperva.com/blog/how-to-identify-a-mirai-style-ddos-attack/, accessed: 2021-05-27.

[2] Y. Seralathan, T. T. Oh, S. Jadhav, J. Myers, J. P. Jeong, Y. H. Kim, and J. N. Kim, "Iot security vulnerability: A case study of a web camera," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2018, pp. 172–177.

[3] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[4] Y. Cheng, X. Ji, X. Zhou, and W. Xu, "Homespy: Inferring user presence via encrypted traffic of home surveillance camera." in *ICPADS*, 2017, pp. 779–782.

[5] B. Copos, K. Levitt, M. Bishop, and J. Rowe, "Is anybody home? inferring activity from smart home network traffic," in *2016 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2016, pp. 245–251.

[6] P.-M. Junges, J. François, and O. Festor, "Passive inference of user actions through iot gateway encrypted traffic analysis," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 7–12.

[7] H. Li, Y. He, L. Sun, X. Cheng, and J. Yu, "Side-channel information leakage of encrypted video stream in video surveillance systems," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[8] 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*, 2018, pp. 1074–1088.

[9] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *arXiv preprint arXiv:1708.05044*, 2017.

[10] Y. Amar, H. Haddadi, R. Mortier, A. Brown, J. Colley, and A. Crabtree, "An analysis of home iot network traffic and behaviour," *arXiv preprint arXiv:1803.05368*, 2018.

[11] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.

[12] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," *arXiv preprint arXiv:1804.03852*, 2018.

[13] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: a machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, 2017, pp. 506–509.

[14] Y. Zhan and H. Haddadi, "Mosen: Activity modelling in multiple-occupancy smart homes," *arXiv preprint arXiv:2101.00235*, 2021.

[15] C. Wampler, S. Uluagac, and R. Beyah, "Information leakage in encrypted ip video traffic," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–7.

[16] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart (er) iot traffic shaping," *arXiv preprint arXiv:1812.00955*, 2018.

[17] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE symposium on security and privacy*. IEEE, 2012, pp. 332–346.

[18] B. L. Welch, "The generalization ofstudent's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[19] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42, no. 1, pp. 31–60, 2001.

[20] M. Vivian, “pcap_analyzer (code that we implemented for the thesis),” https://github.com/Martinviv/pcap_analyzer, 2021.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**