POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Channel State Information (CSI) features Collection in Wi-Fi Access Points for IoT forensics

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Alessia Luoni

Advisor: Prof. Alessandro Enrico Cesare Redondi

Co-advisor: Dr. Fabio Palmese

Academic year: 2021-2022

Abstract

Human activities sensing can contribute to solve a wide range of problems, including those in the IoT Forensics field. Techniques used to perform human detection often require IoT devices that need to be carried, which may be unconventional in places such as at home. Several alternative solutions exist, but their success is affected by the surrounding environment. Therefore, experts can rely on wireless sensing, and in particular on the *Channel State Information (CSI)*, which can leverage the existing radio signals generated inside a Wi-Fi infrastructure for human activities sensing. This work proposes a tool that allows to capture CSI features directly in a Wi-Fi access point and which can be easily controlled via its user web interface. After the project overview and its implementation details, we show how the data extracted from common IoT devices can be used for IoT Forensics analysis tasks, such as detecting the presence of humans inside an indoor environment as well as its passage through the room door.

1. Introduction

In the last years, IoT companies are continuously launching new products into the market, as well as making smart the existing ones used in daily activities. Therefore, we have a high number of connected devices producing traffic simultaneously that could be used as potential source of evidence in case of crimes. This is why IoT forensics is becoming popular and applied. This work focuses on a subcategory of IoT Forensics called IoT Network Forensics, which aims at capturing and analysing IoT device network traffic for forensics purposes. In particular, since collecting CSI data directly from an Access Point (AP) is not often feasible, our first goal is the creation of a tool, named *CSI Feature Sniffer*, that allows the collection of CSI features directly in the access point, to be easily controlled via the user web interface and that returns an already processed file. After presenting the tool implementation details as well as its configuration parameters, we show potential uses of the tool output for IoT forensics analysis tasks. In particular, we show that it is feasible to recognize when a person is inside a room and the moments in which he/she passes through the door using a smart camera as source of traffic. The

rest of the paper is structured as follows: Section 2 presents the CSI related works, Section 3 describes the implementation of CSI Feature Sniffer, Section 4 shows real case scenarios that exploit the collected CSI features, Section 5 contains the results obtained and in Section 6 there are the conclusions.

2. Related works

Device-free methodologies used for human activities sensing in indoor environments includes low-cost approaches mainly based on two indicators: the RSSI (Received Signal Strength Indicator) and the CSI (Channel State Information). However, most of the related works uses the CSI because it is considered more stable and more fine-grained than the RSSI.
CSI is a measurement from the Physical layer which illustrates how the propagation of a signal from the sender to the receiver occurs.
The CSI is represented by a vector where each entry is a complex number, which phase and amplitude of every subcarrier can be derived [4]:

$$\boldsymbol{H_i} = |\boldsymbol{H_i}|e^{j\sin(\angle \boldsymbol{H_i})} \quad (1)$$

Where $|\boldsymbol{H_i}|$ is the CSI amplitude of the i_{th} subcarrier and $\angle \boldsymbol{H_i}$ is its phase.
The paper which inspired the thesis the most demonstrates, by only analysing the CSI feauters collected with two ESP32 microcontrollers, how it can be done a through-wall occupancy monitor activity in a hallway [3].
Another work that applies the same data for similar scopes is presented in [2], where the authors propose a semi-supervised learning approach for counting people in a room. Here Guo et al. leveraged the spatial diversity on the MIMO-enabled WiFi devices to extract effective features from the CSI difference between two antennas.
Moreover, because of their great social impact, many studies have been done for recognizing atypical activities, such as detect falls. The first work proposed that aims at sensing these actions by analysing the CSI is WiFall [5]. In this case, the authors trained the model both with a one-class Support Vector Machine classifier and then with a Random Forest algorithm.
As we have seen, CSI features allow to conduct a wide number of researches. However, this measurement requires one between the specific tools available for capturing data, so among them we chose the Nexmon CSI Extractor [1].

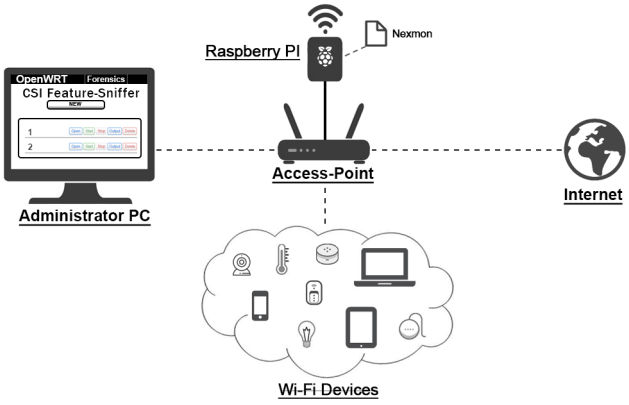

Figure 1: CSI Feature-Sniffer architecture

3. CSI Feature Sniffer

CSI Feature Sniffer is a forensic tool which can be installed in any AP with OpenWrt firmware and that allows users to capture CSI data directly from it through the LuCI web interface. This enables the gathering of data and the downloading of the parsed features in a very user-friendly environment, rather than using several bash commands which require a deep knowledge.

3.1. Architecture

Since the tool is intended to be usable also by non-expert users, its setup must be easily replicable; indeed, its components are:

- **An Access Point (AP)** with OpenWrt and LuCI web interface installed.
- **A Raspberry Pi** equipped with the Nexmon CSI Extractor. It is used to collect the CSI features and to that end it needs to be connected with an ethernet cable to the AP. The user must take care that the Wi-Fi card and the installed kernel are compatible with those required by the nexmon project
- **Wi-Fi devices** that are connected wireless to the AP and from which we want to sniff traffic. In this work, we tested an indoor security camera (Teckin Camera TC100) as the IoT device that sent CSI traffic.
- **A personal computer** from which the user can access CSI Feature Sniffer via the AP web interface.

A sketch of the architecture is reported in Figure 1

3.2. User interface

The CSI Feature Sniffer user interface is an extension of the LuCI web interface. Its homepage allows to create new configurations as well as to control the existing ones with proper buttons to: *open/edit* its settings, *start* and *stop* the CSI capture, *download* the resulting output as CSV, *delete* the configuration. The user can also create new configurations through the proper button that opens a page where to enter the configuration parameters (i.e. Wi-Fi channel, devices to filter).

3.3. Implementation details

3.3.1 AP - Raspberry Pi communication

MQTT was used to connect the AP with the Raspberry Pi, so that Nexmon could be controlled through the web interface.
For establishing the communication between the two devices we installed Mosquitto on both of them and then, on the Raspberry, we created a Python script to connect to the broker and to subscribe to the topics: *start, stop, prepare* and *download*. In addition, based on the received message topic, it is responsible for calling other scripts that activate the CSI Feature Sniffer functionalities. At this point, since we needed to subscribe to the topics as soon as possible, the script was embedded in a service which starts when the operating system boots.

Create: Creating a configuration is an easy functionality that operates only at the AP side, which is triggered after the user enters its parameters and saves it. At this point a function in the web server immediately checks the input and then appends the parameters in a configuration file.

Start: When the user clicks the start button, the AP sends an MQTT publish message to the broker with *prepare* as topic, containing the entire configuration file in the message payload. As the Raspberry receives the publish message, it retrieves the file content and prepares the settings for the Nexmon process. Then the AP sends another publish to the *start* topic that will trigger the Raspberry to start the Python script which executes all the commands required by Nexmon to capture the CSI data. We remark that these phases require the Raspberry to be previously subscribed to the proper topics, as previously stated.

Stop: The capture interruption is simply implemented by the AP sending a message to the topic *stop*, causing the subscribed process in the Raspberry side to send a SIGINT signal to kill the Nexmon capture execution.

Download: In order to avoid the downloading of old .csv files, the first thing that happens when hitting the download button, is the deletion of possible retained messages.
Next, the AP publishes a message with topic *download*, the receipt of which triggers the start of a Python script on the Raspberry that initially parses the file resulting from the capture, then creates a CSV file containing all the CSI features and finally sends it as a retained MQTT publish message. The file is then forwarded to AP as soon as it subscribes to the *output* topic.
However, the file is not immediately ready to be downloaded by the end user, but he/she has to wait for all the data to be sent, so the whole procedure requires several request/response messages until the final output is eventually sent back to the user in the HTTP response.

4. Human detection with CSI

All the experiments are conducted in the same room and with no one else besides the moving person in this environment.Several scenarios are tested in order to understand how the Channel State Information react to human activity. In particular, we examined the following scenarios:

- **Scenario 1:** The Raspberry Pi and the Teckin camera are facing each other at the room entrance and spaced $\sim$ 1.5 m apart. In this case, at the beginning of the capture the person is just outside the room. Afterwards, she enters the room, passes between the devices, walks until she reaches the opposite wall, then turns around and reaches the starting position. All these actions are repeated 20 times, which results in 40 door crossings. Since the devices were both near the entrance, in this context, the main focus was to understand whether it was possible to detect the moment in which a person

passed through the door from the CSI.

- **Scenario 2:** The Raspberry is placed at the room entrance, while the camera is placed near the right wall, around ~ 5.5 m away from the other device.
  In this case, at the beginning of the capture the person is outside the room. Afterwards, she enters the room, walks a few seconds inside it, then turns around and reaches the starting position. All these actions are repeated for 15 times, which result in 30 crossing from the door. This time, the two instruments are far apart, so our goal is to detect the presence of the person in the room, instead of detecting the entrance.
- **Scenario 3-4**: these two experiments are analogous to Scenario 1 and 2 respectively, but with the smart camera facing the wall. This means that for Scenario 3 the camera has not the room entrance in its Field of View (FoV) and the same holds for Scenario 4, in which the moving person is outside the FoV of the camera. The goal of the experiments is the same as before, detecting the passage (Scenario 3) and the presence (Scenario 4) of a moving person.

4.1. Preprocessing and data analysis

We firstly collected CSI data by only filtering the traffic from the camera on the 20 MHz band, after we moved on to the preprocessing and analysis stages. Before starting the preprocessing we opened the PCAP file produced by the Nexmon tool containing the CSI capture using the *csiread* tool, a library available on GitHub which parses data. After this parsing, we obtain the CSI features (for each one of the data subcarrier available in the band, we got for example, the subcarrier number, the frame to which it belongs and the amplitude derived from the CSI value) and then we create our dataframe.
Although the CSI measurements are composed of both phase and amplitude, only the amplitudes have been considered for this experiment because the phases are influenced by device clock and carrier frequency, so they need a calibration which introduces several drawbacks. The next step involves the discovery of outliers, which consists in applying a windowed filter for each subcarrier $i \in I$, where $I = [-28, -27, \ldots, -1, 1, \ldots, 27, 28]$. Specifically, the dataframe is divided into w seconds wide non-overlapping windows. Therefore, each packet is assigned to a proper window by checking its timestamp.
Subsequently, the filtering was applied as follows:

$$\overline{\boldsymbol{A}}_{\boldsymbol{t}}^{(\boldsymbol{i})} = \begin{cases} \boldsymbol{A}_{\boldsymbol{t}}^{(\boldsymbol{i})} & \frac{|\boldsymbol{A}_{\boldsymbol{t}}^{(\boldsymbol{i})} - \mu(\boldsymbol{A}^{(\boldsymbol{i})}\{\overline{\boldsymbol{w}}\})|}{\sigma(\boldsymbol{A}^{(\boldsymbol{i})}\{\overline{\boldsymbol{w}}\})} < \lambda \\ \boldsymbol{A}_{\boldsymbol{t-1}}^{(\boldsymbol{i})} & \boldsymbol{otherwise} \end{cases}$$

Where:

- $\mu(\boldsymbol{A}^{(\boldsymbol{i})}\{\overline{\boldsymbol{w}}\})$ is the mean of the amplitude values for the *i-th* subcarrier in the window $\overline{w}$.
- $\sigma(\boldsymbol{A}^{(\boldsymbol{i})}\{\overline{\boldsymbol{w}}\})$ is the standard deviation of the amplitude values for the *i-th* subcarrier in the window $\overline{w}$.

After the cleaning of the amplitude measurements, some statics were applied. The first computation included the calculation of the standard deviation as before, for each subcarrier, to update the value of $\overline{\boldsymbol{A}}_{\boldsymbol{t}}^{(\boldsymbol{i})}$:

$$\overline{\boldsymbol{A}}_{\boldsymbol{t}}^{(\boldsymbol{i})} = \sigma(\boldsymbol{A}^{(\boldsymbol{i})}\{\overline{\boldsymbol{w}}\}) \tag{2}$$

Then, the dataframe was modified again in order to compute $\boldsymbol{A_{CSI,t}}$ as the average standard deviation on all the subcarriers for a certain time instant t:

$$\boldsymbol{A_{CSI,t}} = \mu(\overline{\boldsymbol{A}}_{\boldsymbol{t}}^{(\forall \boldsymbol{i} \in \boldsymbol{I})}) \tag{3}$$

Subcarriers close to each other should have similar values, so, when there is noise across several subcarriers at time t, $A_{CSI,t}$ assumes a high value, which means that a target is present in the environment; conversely, if the noise is not frequent across subcarriers, $A_{CSI,t}$ is low and it means that no target is present [3]. At this point, we need to understand if there is effectively a correlation between the $A_{CSI,t}$ and the passage/presence of a person. To do it, we used a binary threshold classifier.
We firstly added a ground truth column to the dataframe where the rows assumed a specific value in accordance with the activity the person was carrying out in that moment: in the case of the passage detection task, the correspondent value assigned was 0 if in that moment the person was not passing through the entrance/exit, while in the presence detection alternative, it was 0 when the person was not inside the room; on the contrary, the value was 1 whenever the

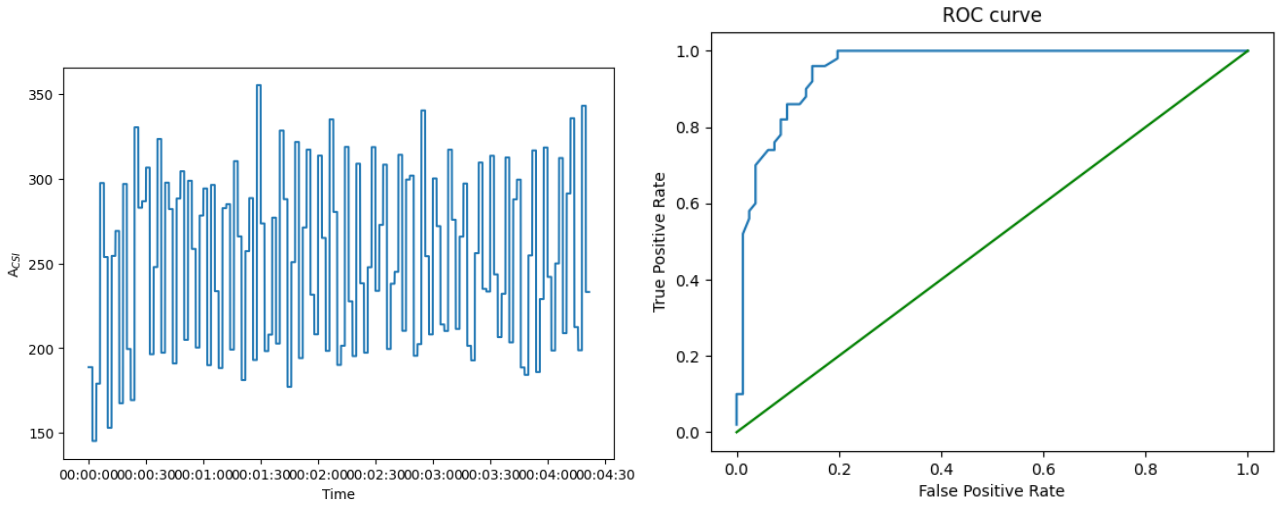

Figure 2: Scenario 1 $\boldsymbol{A_{CSI}}$ and ROC curve

person was passing or was inside the room.
The following step included the definition of a threshold parameter $\boldsymbol{\tau}$, which was initialized with the value of the minimum $\boldsymbol{A_{CSI,t}}$ and that produced a prediction variable $\boldsymbol{Y}$, which could assume the values 0 or 1. Its assignment follows this rule:

$$\boldsymbol{Y_t} = \begin{cases} \mathbf{1} & \boldsymbol{A_{CSI,t} \geq \tau} \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

After the comparison of each $\boldsymbol{A_{CSI,t}}$ with the threshold parameter, $\boldsymbol{\tau}$ was incremented by 1 until it reached the maximum $\boldsymbol{A_{CSI,t}}$ value, and at every update the comparisons were repeated.

4.2. Classification metrics

At each iteration, before updating the threshold, the predicted Y(t) values is compaired with the ground truth column in order to evaluate the results.
Since binary variables are used, we computed: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (TN). From them we then obtained the following classification metrics that are required to achieve ROC curves :

- **True Positive Rate (TPR) or sensitivity**: the probability that an actual positive is classified as positive

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

- **False Positive Rate (FPR)**: the probability that an actual negative is classified as positive

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

5. Experimental results

The scenarios mentioned in section 4 are tested with several values for the parameters w (window size) and λ (used for filtering outliers) and the best ones produced the following results.
The first scenario is the one that yielded the best results (represented in fig. 2),which means that the classifier, when the camera was close and facing the Raspberry, was almost always able to recognize the passage of a human. Indeed, the resulting AUC value was 0.9551.
The second scenario, which had the camera distant from the entrance and whose objective was to recognize the presence of a person in a room, also performed well, indeed it reached an AUC value of 0.9376.
Considering scenario 3, the AUC value of 0.8380 shows that turning the camera toward the wall decreased the performance, but nevertheless the presence detection still occurred for the vast majority of times.
As for the last scenario, it was the one that gave the worst outcomes, although theyactually cannot be considered bad this time either, since we got a value for the AUC of 0.8135.

5.1. What happens if the device sends less packets?

Considering the results obtained, we then tried to stress the tests further to see the changes obtained if the number of packets sent by the IoT device was lower. This is done also because usually smart cameras produce much more traffic than other types of IoT devices as for example smart plugs, smart bulbs or other sensor devices. In order to simulate the behaviour of different rates IoT devices, we reduced the number of packets produced by the camera simply discarding some of the packets captured. In particular, we compared the results where we had all packages, half of the packets, 1/5 of the packets, 1/10 of the packets, 1/50 of the packets.
We discovered that a large number of frames had to be removed before performance deteriorated.

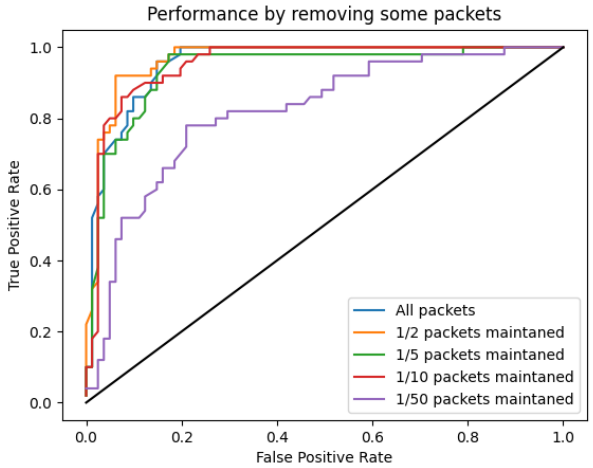

Figure 3: Scenario 1 with less packets

In fact, in most cases, the results remained almost unchanged as long as 1/5 packets were kept in the dataframe. Actually, in some situations, such as in scenario 1 (represented in fig. 3) and 4, the outcomes even improved when half of the packets were eliminated. Except that for the first scenario, where performance was still good, the threshold that started the deterioration of the results was when only 1/10 of the frames were kept in the dataframe.
Finally, in all cases, when only 1/50 of the packets were considered, the shapes of the ROC curves deteriorated dramatically, which was predictable since, in doing so, only a little more than one packet per second was considered.

6. Conclusions

This work presented *CSI Feature Sniffer*, a tool that allows Channel State Information features collection in OpenWrt based Wi-FI Access Points. In particular the aim of the work has been to ease the collection of such features and to show its possible application cases for IoT forensic analysis tasks by extracting such features from common IoT devices.
As future research directions we plan to test our system with different IoT devices to understand whether they can all recognize the activities performed in this thesis and to find which one performs better.
In addition, it would be interesting to see how the CSI features react if multiple subjects take turns in performing the described activities. From this, using machine learning algorithms, work could be done aimed at trying to recognize the person who is performing the action.

References

[1] F. Gringoli, M. Schulz, J. Link, and M. Hollick. Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets. In *ACM WiNTECH 2019*.

[2] X. Guo, B. Liu, C. Shi, H. Liu, Y. Chen, and M. C. Chuah. Wifi-enabled smart human dynamics monitoring. In *SenSys '17*. ACM.

[3] S. M. Hernandez and E. Bulut. Adversarial occupancy monitoring using one-sided through-wall wifi sensing.

[4] J. Liu, H. Liu, Y. Chen, Y. Wang, and C. Wang. Wireless sensing for human activity: A survey. *IEEE Communications Surveys Tutorials*, 2020.

[5] Y. Wang, K. Wu, and L. M. Ni. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 2017.