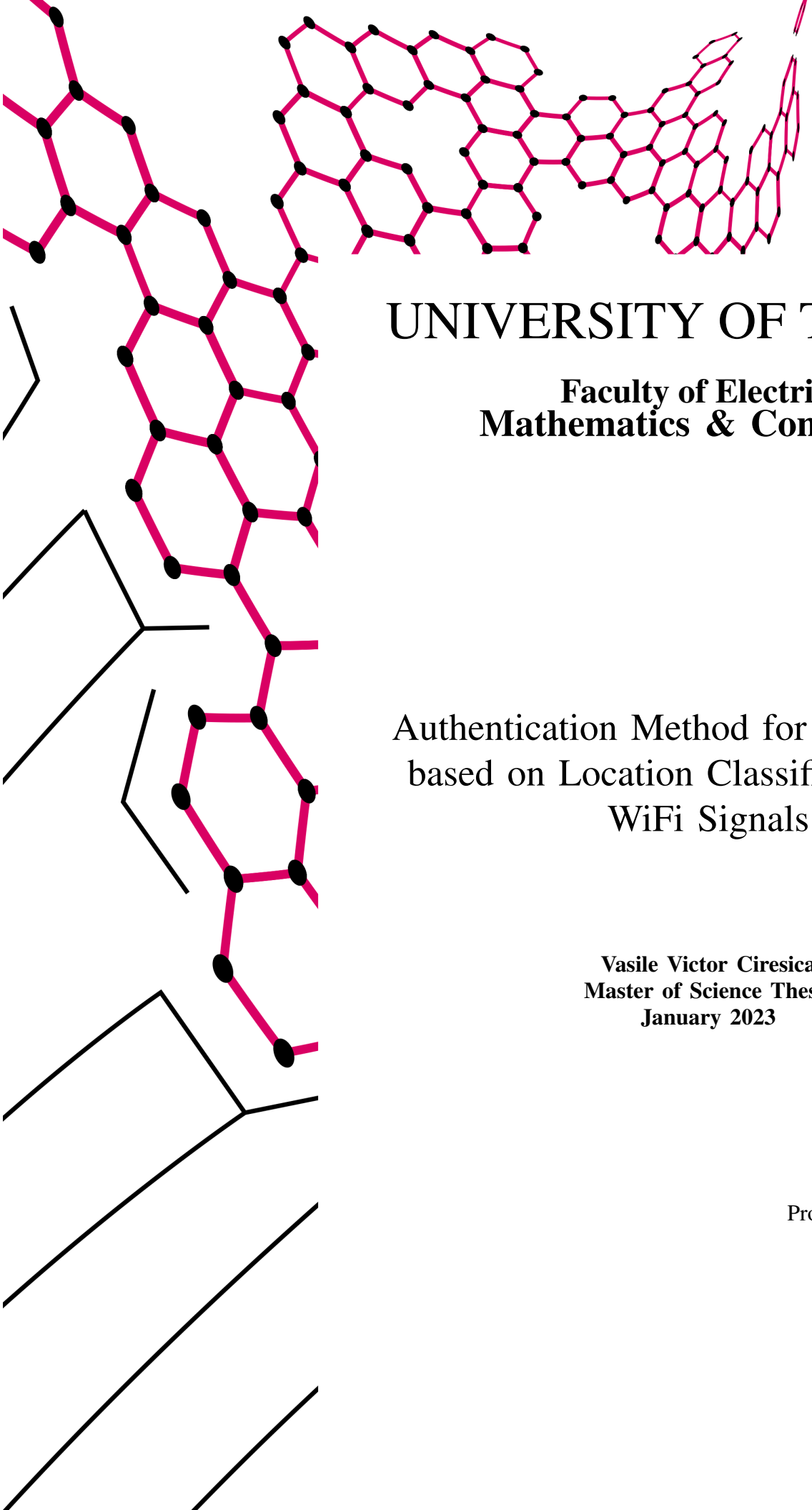

UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Authentication Method for Windows OS based on Location Classification using WiFi Signals

Vasile Victor Ciresica
Master of Science Thesis
January 2023

Supervisors:
Prof. Dr. Andreas Peter
Prof. Dr. Maarten van Steen
Philipp Jakubeit

University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

***Abstract*—In an ever evolving digital world authentication is an essential procedure in the day-to-day activities which involve interacting with digital systems. It is a basic and crucial part of the security of a system, as it is the requirement for two parties to assure the identity of the other. Its apparatus is continuously increasing in complexity to keep the systems secure against evolving adversaries. Multi-factor authentication systems have been widely adopted in recent years to increase the security of the systems in terms of assurance of authenticity. This led to the research and development of numerous authentication methods. In this wealth of authentication methods, conditions of the user, like location, stand out as a candidate for further research and development. They have the advantage of being intrinsically nonintrusive (i.e. do not require user input). In this paper we extend an existing scheme which describes the use of WiFi for location based authentication. This classifies WiFi signal measurements to be on location or off location, in a privacy preserving manner. Our work focuses on adapting this authentication method to the predominantly used operating system Windows. We constructed a dataset of 33 non-overlapping locations on which we build the adapted method for Windows. We then evaluate its performance and benchmark it against state-of-the-art authentication methods designed with the same objectives and with similar traits. We also present limitations of the method and particularities that Windows imposes for WiFi data analysis. Finally, we describe further directions that can be explored regarding location authentication based on WiFi data.**

1. Introduction

Migration toward digital systems is growing rapidly in most domains nowadays to meet the consumers' needs. A direct implication of this massive increase is the high demand for cyber security to protect the digital services from being misused. The US Bureau of Labour Statistics forecasts an increase of 33% in the number of security analyst jobs between 2020 and 2030, much more than the average growth for all jobs (8%) or computer science jobs (13%) [1].

One of the most basic and common procedures when interacting with a digital system in a secure manner is authentication. Authentication is defined in the ISO 27000 standard [2] as the "provision of assurance that a claimed characteristic of an entity is correct". The National Institute of Standards and Technology (NIST) defines authentication as "Security measures designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information" [3]. Authentication can be achieved through multiple methods, distinguished by their authentication factors. The most common authentication schemes use as claimed characteristic something you know (e.g. a password), something you have (e.g. a token), something you are (e.g. fingerprint) or somewhere you are (e.g. location) to identify a user. Currently, the use of combined factors into a Multi-Factor Authentication (MFA) scheme, to improve assurance of authenticity, has became prevalent [4]. MFA is a must for entities operating with high value assets or privacy sensitive data, like banks or hospitals. According to the 2017 Annual Report to US Congress on the Federal Information Security Management Act [5] up to 65% of cyber security incidents could have been prevented with strong MFA.

Location as authentication claim can be used in MFA schemes. Location can be determined using data from different sensors, e.g. GPS, proximity sensor. Unfortunately, using the geographical location of a user has the drawback of being privacy invasive. To overcome this, the location can be described in terms of other sensor measurements. In this paper we study how ambient WiFi signals can be used for location classification, which has authentication as a direct application. Those signals can be collected from surrounding Access Points (AP) and processed in a privacy preserving manner, as was shown by [6].

A. Motivation

The main advantage of location based authentication is being intrinsically nonintrusive. Nonintrusiveness describes that user input is not required [7]. This property will increase the adoption of MFA schemes, as it makes the process of authentication easier for users. While there already exist many methods that are nonintrusive, none have been widely adopted because they require dedicated

sensors, like wristbands [8], or, in the case of gait based authentication, it requires a physically moving user [9]. WiFi, on the other hand, is ubiquitous and identifying the location does not require user movement.

A preliminary work in the direction of WiFi-based authentication was done by the authors of [6]. They developed a method that uses WiFi signals to uniquely identify non-overlapping locations. Further, they build a dataset and analyse the performance of the method in the real-world environment. Their results show that WiFi-based location classification is a feasible method that can be applied for authentication. The authors of [6] developed and tested their methods for the Linux Operating System (OS), which represents a small percentage of overall laptop users. A more widespread OS is Windows, which, according to Statscounter, a web traffic analysis service, is used by the vast majority (76.33%) of laptop users. Thus, to facilitate mass adoption of WiFi-based authentication we use the work in [6] as a foundation to build a location classifier based on WiFi for the Windows OS.

B. Research Questions

In this paper we describe the steps we made to implement and evaluate the WiFi-based location classification method for the Windows OS. Throughout the research the aim was to answer the following research questions:

- How can the WiFi data be accessed on Windows?
- What data from the WiFi signals, available on Windows, is most suitable for WiFi-based authentication?
- How can a WiFi-based location classification method for Windows OS be implemented?
- How does the WiFi signal data influence the security and privacy of the method?
- How does the performance of the WiFi-based location classification method compares to other methods used in authentication?

To answer the research questions we look into how to access WiFi data and how the data can be used. We also investigate the entropy of WiFi data accessible on the Windows OS and possible privacy issues regarding this data. Further, we look into WiFi data processing and location classification for authentication.

To develop the classification method, evaluate the performance and showcase our results, we use a real-world dataset of WiFi signals that we gathered through distributed scanning, using a dedicated software we developed during the research.

C. Contributions

This work delivers four main contributions. Our first contribution is to highlight a method to access WiFi data on the Windows OS. This method can be used in further research regarding WiFi. Our second contribution is to build a software tool capable of collecting WiFi data. This software includes a user interface which makes it easy to use and it can be distributed to build datasets of WiFi signals. Another contribution we bring is a real-world WiFi dataset of 33 non-overlapping locations. This dataset can be used to develop, test and benchmark new methods that employ WiFi signals. Having common datasets tremendously improves accuracy when comparing the performance of different methods. Our final contribution is developing a location classification method based on WiFi, for Windows. We use the work of [6] as foundation. By making the method available on Windows we unlock the vast majority of laptop device users, making the method more feasible for mass adoption, which will further lead to an increase in the users' data security.

D. Outline

In this section we provide an overview of the structure of the thesis.

Section 2 introduces the background knowledge and the relevant literature that forms a basis for this thesis. We detail how the WiFi protocol enables devices to distribute information through radio signals, and how the data packets are structured, what information they carry and how can it be extracted. Also, we present the authentication method designed in [6].

Section 3 provides a thorough description of how the dataset for the research was built, how the

software tool was developed to collect the WiFi data and how the preliminary processing was performed. We also present the analysis of the WiFi data, how the classification method is designed and how it is applied on the dataset. Further more, we describe the new parameters introduced by the Windows OS.

Section 4 describes attack scenarios and if the WiFi based authentication method is vulnerable in those scenarios. We also provide additional actions the user can take, if necessary, to ensure secure authentication.

Section 5 showcases the results of the analysis and we interpret them in perspective to the research questions. Also, we compare our solution with other results from the literature.

Section 6 concludes this work. We provide the answers to the research questions, describe the limitations that were encountered and propose directions for future research.

2. Background and related work

In this section, we introduce preliminary knowledge which forms the basis for understanding the research further presented in this thesis. We also discuss the related research that forms the foundation for this paper.

A. *The IEEE 802.11 WiFi standard*

Many advances in technology came as solutions to the connectivity needs of people. Devices are becoming smaller and smaller so they are easier to carry and communication between them becomes simpler to achieve. One technology that facilitates connectivity is WiFi. WiFi is a set of protocols that enables devices to communicate over a small area using radio signals, and connect to the Internet. The Institute of Electrical and Electronics Engineers (IEEE) defined the 802.11 Standard for WiFi to assure compatibility between devices. Over the years WiFi gained popularity at a very fast pace and became an ubiquitous technology.

The IEEE 802.11 standard defines the protocols used to manage access to the WiFi media channel and transmit data. Well defined rules for channel access are required because it must be shared between different devices and they can easily interfere with each other.

The WiFi protocol also defines the format and content of the frames that the devices send. In order to manage the proliferation of WiFi enabled devices, they are grouped into small local networks. Access to a WiFi network and communication between devices are mediated by an AP. The AP periodically broadcasts special packets called discovery beacons to make the network visible to devices in its proximity. Those beacons contain the information necessary for devices to connect to the network and start communication. The information in a beacon frame is organised in fields. There are 247 element fields defined in the 802.11 IEEE 2020 standard [10]. The fields present in a discovery beacon frame vary with the network's capabilities and the devices that are part of the network.

B. *WiFi beacon frame format*

The beacon frame does not contain application data, it contains only meta data about the network, the radio channel and the communication parameters used to facilitate the actual data exchange. The format of the beacon frame is show in **Figure 1**. The WiFi beacon contains two types of fields, fixed fields and element fields.

The fixed fields are mandatory fields with a set length. Those fields include the time period at which beacon frames are sent, the physical capabilities for data transmission supported by the device and information about the network's structure and properties.

The element fields occupy a larger part of the beacon frame. They are variable length fields, they are not all mandatory and are designed to be flexible, to support protocol updates. As shown in **Figure 2** each field has an ID (1 byte), a length in bytes (1 byte), an ID extension (0 or 1 byte) and a body which can vary between 1 byte and 253 bytes. The ID and length subfields have a fixed format while the ID extension and body vary depending on the element field.

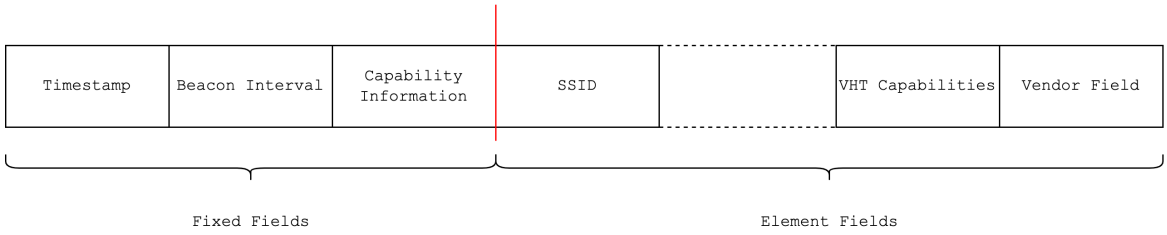

Fig. 1: Beacon frame format

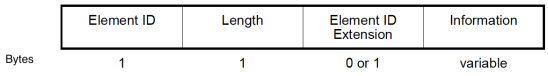

Fig. 2: Element field format

We can use the mentioned knowledge of the beacon frame format to process WiFi data from surrounding APs. We can split a beacon frame into fields and process each field accordingly.

The first step in our analysis is to access the WiFi data. Most personal-use devices come equipped with a WiFi antenna. Those include laptops, mobile phones, smart-watches, smart-bracelets. For those devices, WiFi communication is handled by the OS and user applications have limited access. This is done to protect the hardware and to simplify programming applications that communicate over the Internet. Sometimes access to the WiFi internals of a device is necessary for a user application. For this reason some OSs offer more elaborate interfaces. We can leverage the interfaces the OS supplies in order to access the necessary data for our WiFi-based Authentication method.

C. WiFi Authentication Method for the Linux OS

The authors of [6] propose a WiFi-based classification method and show how it can be used in a multi-factor authentication scheme. The method uses six WiFi beacon fields that are accessible on Linux and the received signal strength indicator (RSSI) to construct fingerprints of different locations. The RSSI represents the power of the radio signal received by the WiFi antenna of the device. The six fields used in [6] are: the Capability flags, the WiFi Protected Access (WPA) flags, the Robust Secure Network (RSN) flags, the Frequency, the Mode and the Maximum Bit Rate. The fingerprints build from the six fields and the RSSI are then compared to determine if they were constructed from measurements collected at the same physical location or not.

There are two key factors in deciding which fields to use for classification, specifically, availability and privacy. Preserving the privacy of a user implies not disclosing sensitive personally identifiable information (PII). PII is data that can be traced back to an individual and that, if disclosed, could result in harm to that person. Such information includes biometric data, medical information, financial information and unique identifiers such as passport or social security numbers. Threats to privacy include not only crimes, such as identity theft, but also disclosure of personal information that the individual would prefer remained private.

The authors of [6] also have access to the Media Access Control (MAC) address of the AP and the Service Set Identifier (SSID), but both represent PII. The MAC address is a 6-byte number that encodes the device vendor and the device model. An adversary can use this information to increase the probability of a successful attack. The SSID is a 32-byte text field representing a human readable identifier of the WiFi network. The SSID is set by the administrator of the AP and might contain sensitive information, for example his address. Also, the default value of the SSID of a device contains device specific information (e.g. model, vendor, internet service provider). This information can be used by an adversary to identify vulnerabilities

based on the AP's hardware or software. These vulnerabilities will most likely result in a higher probability of a successful attack on the AP. The authors of [6] choose not to use these fields in order to protect the user's privacy. Although they aim to construct a fingerprint of the WiFi signals at a location, it is not the geographical location that we fingerprint, but the measurable environment. We also choose to avoid fields that can be linked to a geographical location or contain PII.

The method for constructing the WiFi fingerprint from [6] has two main components, an access point identifier (APID) and the RSSI. The APID is constructed using the six mentioned fields, by concatenating them into a 63 bits value [11]. The RSSI is measured in decibels relative to a milliwatt (dBm) or as a percentage, denoting the measured dBm of the signal relative to the maximum measurable dBm value of the antenna (i.e. $\frac{measured\ dBm}{maxim\ dBm}$). In [6] further normalization is applied such that the RSSI $\in [0, 1]$.

The WiFi-based authentication method defined in [6] uses as main component a WiFi-based location classifier *C* to establish if two sets of measurements were conducted at the same location. To define how the classifier works, they first distinguish a measurement (*m*), a scan (**S**) and a profile ($\mathbb{S}$). A measurement *m* is defined as the set of fields gathered from a beacon frame (i.e. the WiFi data), transmitted by an AP at a certain moment in time. A scan **S** is defined as a series of measurements *m* collected over a fixed period of time, with the same sensor and at a fixed location. A profile $\mathbb{S}$ is defined as a set of scans collected over a fixed period of time.

In order for a user to authenticate using the method from [6] he would follow the next steps. First a profile $\mathbb{S}$ of a location is registered by a user by collecting measurements at the location. Then, when a user tries to authenticate, measurements are collected from the AP signals in the surrounding environment to construct a scan **S**. Next, a similarity metric is computed using **S** and $\mathbb{S}$ and compare to a threshold to determine if **S** is sufficiently similar to $\mathbb{S}$ (i.e. it is smaller or bigger than the threshold). If it is sufficiently similar, the authors of [6] infer that the location where **S** was collected is the same as the location where $\mathbb{S}$ was collected. The similarity between **S** and $\mathbb{S}$ is computed using a modified version of Jaccard Similarity that takes into account the APID and the RSSI. The formula for the Jaccard Similarity with RSSI (**JSR**), defined in [6], is:

$$JSR(\mathbf{S}; \mathbb{S}) = \frac{|\mathbf{APID}(\{\mathbb{S}, \mathbf{S}\})|}{|\mathbf{APID}(\mathbf{S}) \cup \mathbf{APID}(\mathbb{S})|} \tag{1}$$

The denominator $|\mathbf{APID}(\mathbf{S}) \cup \mathbf{APID}(\mathbb{S})|$ is the total number of unique APIDs in either the scan **S** or the profile $\mathbb{S}$. To define the numerator $|\mathbf{APID}(\{\mathbb{S}, \mathbf{S}\})|$, they first define $RSSI(\mathbf{S}; AP)$ as the average of all RSSI values from the measurements from the same AP (i.e. with the same APID), belonging to the scan **S**. Next, they define

$$\begin{aligned} ||\mathbf{S}; \mathbf{S'}; APID|| = ||RSSI(\mathbf{S'}, APID) - \\ RSSI(\mathbf{S}, APID)|| \end{aligned} \tag{2}$$

as the absolute difference between the average RSSI values for the same AP, from two scans **S** and **S'**. Using the previous defined absolute RSSI difference, they use $\{\mathbf{S}, \mathbb{S}\}$ to denote the collection of scans $\mathbf{S}'$ from the profile $\mathbb{S}$, that contain only APs that are in **S** and the absolute RSSI difference between **S** and $\mathbf{S}'$, for all common APs, is smaller than a profile specific threshold $d(\mathbb{S})$. From the definition of $\{\mathbf{S}, \mathbb{S}\}$ they obtain the following equation:

$$\begin{aligned} \{\mathbf{S}, \mathbb{S}\} = \{APID \mid \left\|\mathbf{S}, \mathbf{S}', APID\right\| < d(\mathbb{S});\ \mathbf{S}' \in \mathbb{S}; \\ APID \in \mathbf{S}' \cap\ \mathbf{S}\} \end{aligned} \tag{3}$$

The set $\{\mathbf{S}, \mathbb{S}\}$ is strictly dependent on $d(\mathbb{S})$, which is defined in [6] as the maximum absolute difference between the RSSI values for the same APID, from any two scans in $\mathbb{S}$, **S** and **S'**. The formula for the threshold $d(\mathbb{S})$ is given by:

$$\begin{aligned} d(\mathbb{S}) = max\{\left\|\mathbf{S}; \mathbf{S}'; APID\right\| \mid AP \in APID(\mathbf{S}) \\ \cap APID(\mathbf{S}'), \mathbf{S}, \mathbf{S}' \in \mathbb{S}, \mathbf{S} \neq \mathbf{S}'\} \end{aligned} \tag{4}$$

Thereby, the numerator of $JSR(\mathbb{S}; \mathbf{S})$, $|\mathbf{APID}(\{\mathbb{S}, \mathbf{S}\})|$, is the total number of APs in $\{\mathbb{S}, \mathbf{S}\}$.

After the JSR is computed, the WiFi-based classifier compares the value to a precalculated threshold $T(JSR; \mathbb{S}; \mathbb{U})$ to determine if the scan $\mathbf{S}$ was conducted at the location of the profile $\mathbb{S}$. The threshold is computed using existing scans and their location labels. To compute the threshold the authors of [6] first calculate the local scan limit (LSL) and the remote scan limit (RSL). The LSL is computed for a given profile using all the existing scans in the profile. The LSL is the minimum JSR between each scan in the profile and the profile itself.

$$LSL(JSR; \mathbb{S}) = min(\{JSR(\mathbf{S}, \mathbb{S}) \mid \mathbf{S} \in \mathbb{S}\}) \quad (5)$$

To calculate the RSL for a given profile $\mathbb{S}$, they first define the set of L-remote scans $\mathbb{U}$ as scans that were collected at different locations than the profile $\mathbb{S}$. For each remote scan they compute the JSR against the profile and take as the RSL the maximum value, as defined in [6].

$$RSL(JSR; \mathbb{S}; \mathbb{U}) = max(\{JSR(\mathbf{U}, \mathbb{S}) \mid \mathbf{U} \in \mathbb{U}\}) \quad (6)$$

Given the LSL and the RSL, the threshold for determining if a scan was conducted at the same location as a profile is defined as the average between the LSL and the RSL.

$$T(JSR; \mathbb{S}; \mathbb{U}) = \frac{LSL(JSR; \mathbb{S}) + RSL(JSR; \mathbb{S}; \mathbb{U})}{2}$$

In summary, the WiFi-based classifier C computes JSR and $T(JSR; \mathbb{S}; \mathbb{U})$ and return the results of the comparison $JSR(\mathbf{S}; \mathbb{S}) > T(JSR; \mathbb{S}; \mathbb{U})$, as defined by Equation 7. To use the classifier for authentication we consider $\mathbb{S}$ as a valid location fingerprint registered by an user and $\mathbf{S}$ as a login attempt at a location. The classifier will determine if there is a match between the profile and the scan, and using the result we can grant or deny access to the user, as described in [6].

$$C(\mathbf{S}, \mathbb{S}) = JSR(\mathbf{S}; \mathbb{S}) > T(JSR; \mathbb{S}; \mathbb{U}) \quad (7)$$

As stated, an objective that the authors of [6] aim to achieve when doing WiFi-based authentication is to preserve privacy. To achieve this the APID is constructed using six features of the WiFi beacon. Those features are defined in [11] and are accessible on Linux via the desktop bus, which is the inter-process communication daemon used in Linux systems.

3. WiFi Authentication Method for the Windows OS

As stated, Linux covers a small part of the total laptop OS market as opposed to Windows. Creating an authentication method for Windows opens the path for mass adoption and will strengthen the resolve for doing additional research into WiFi-based authentication. The main difference between the two OSs is the interface offered to access WiFi data. Linux OS offers unprivileged access through the *desktop bus* to only a few bytes from the WiFi beacon frame. This data consists of eight fields and the RSSI, out of which the MAC address and the SSID cannot be used because of privacy concerns. Windows offers access to WiFi data through the C/C++ Native WiFi API. Through this API we can retrieve the entire beacon frame and the RSSI.

A. Data Collection

In order to assess the reliability and performance of our authentication method we require a dataset. The datasets available online either do not contain the details required or do not contain measurements over a long enough period of time, as required for our analysis. Therefore, we decided to build our own dataset based on measurements we conducted. During the process of data collection we developed a software tool that uses the C/C++ Native WiFi API to record beacons sent by APs for extended periods of time [12], [13].

We developed a web application that queries the OS internals, through the C/C++ Native WiFi API, once every second, for the buffered WiFi beacons the antenna received from the surrounding APs. This is achieved using the tool in [14]. The beacons received from the Native WiFi API are formatted and written into a capture file.

To make the tool easy to use we decided to wrap the C software into a web application with a simple

user interface. To do this we used the Python C API [15] to package the C software as a python module and run it on a python server. Further, we build a server using the Flask web framework and a web interface using the React framework. Also, we modified the C software to save the RSSI value, and the timestamp at which the WiFi beacon was received by the antenna, into the capture file.

To gather the necessary data we conducted measurements during a four months period. The process consisted of finding volunteers to use the software tool to conduct measurements in non-overlapping geographical locations. During this period we constructed a dataset of scans from 33 non-overlapping locations. Each location has measurements collected over a time period between one hour and four hours. In the **Appendix** we show statistics about our dataset, for each location. We computed the minimum, maximum and average number of APs per scan, for each location. We also vary the length of a scan to see how it influences the number of APs. We can see in the table's summary section that as we increase the length of the scan we have more APs in average in a scan. Each additional AP gives us more data that we can use to classify locations apart.

B. WiFi Beacon Fields

As stated, Windows provides access to the entire beacon frame as opposed to unprivileged access in Linux, which provides access to only eight features of the beacon frame. Access to more features in the beacon can increase the performance of our method, but not all fields can be used. When constructing the APID there are several factors to consider when choosing the fields we include. The most important factor is privacy. On Windows we have access to the MAC address and the SSID, as we do on Linux, but those contain privacy sensitive information. To preserve the privacy of the user we exclude those fields from the APID. Another field that contains privacy sensitive information, and we do not include in the APID, is the Country field. This element field specifies the country in which the device operates.

Another factor we consider when choosing the fields to include in the APID is invariance over time. This property is essential because the purpose of the APID is to identify an AP regardless of the moment in time the measurement was taken. To find fields that vary over time we use the dataset we collected. We group the measurements by MAC address so the measurements in each group were taken from the same AP. We then compare the values for each field in each group to identify the variable fields. Our dataset shows that the following 11 fields either change their value over time or are not present at all in subsequent beacons from the same AP: *DS Parameter Set, TIM, BSS load, ERP, HT Information, Secondary Channel Offset, Extended Capabilities, Transmit Power Envelope, Operating Mode Notification, Element ID Extension, Vendor Field, Timestamp, Beacon Interval*.

Another important characteristic of the beacon frame we consider when choosing the fields is that the Vendor Field can appear multiple times in the beacon. The Vendor Field, as defined in the 802.11 standard [10], "is used to carry information not defined in this standard within a single defined format, so that reserved element IDs are not usurped for nonstandard purposes and so that interoperability is more easily achieved in the presence of nonstandard information". To facilitate interoperability, any vendor can register with the IEEE authority to obtain an organization identifier. This organization identifier is included in the Vendor Field for devices to identify Vendor fields they can process. The format of the Vendor Field is show in **Figure 3**.

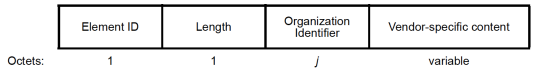

Fig. 3: Vendor field format

A common practice of AP manufacturers is to include their own vendor fields to extend the capabilities of their APs. We cannot include the vendor fields of AP manufacturers as information about the manufacturer of a user's AP is considered privacy sensitive information (the same reason we do not include the MAC address). Also, from our dataset we determined that some vendor fields from AP

Field ID	Field name	Description	Field length	Reason to exclude
0	SSID	Is a user-friendly identifier for the local network	0-32 Bytes	**privacy invasive**
3	DS Parameter Set	Contains information to allow channel number identification for devices	3 Bytes	**variable field**
5	TIM	Element used to signal the timing and availability of data for associated devices	4-256 Bytes	**variable field**
7	Country	Specifies the country in which the device operates	4-256 Bytes	**privacy invasive**
11	BSS Load	Contains information on the current device population and traffic levels in the BSS	7 Bytes	**variable field**
40	Quiet	Used to advertise an interval during which no transmission occurs in the current channel	6 Bytes	**variable field**
42	ERP	It contains the requirement of the ERP element sender (AP, IBSS device, or mesh device) as to the use of protection mechanisms to optimize BSS performance and as to the use of long or short Barker preambles	3 Bytes	**variable field**
61	HT Operation	Controls the operation of HT devices in the BSS	24 Bytes	**variable field**
62	Secondary Channel Offset	This element field is used by an AP when changing to a new 40 MHz or wider channel	3 Bytes	**variable field**
127	Extended Capabilities	The Extended Capabilities element carries information about the capabilities of a device that augment the capabilities specified in the Capability Information field	2-256 Bytes	**variable field**
195	Transmit Power Envelope	Conveys the local maximum transmit power for various transmission bandwidths	4-7 Bytes	**variable field**
199	Operating Mode Notification	Is used to notify devices that the transmitting AP is changing one or more of its operating channel width, the maximum number of spatial streams it can receive, and its LDPC receive preference	3 Bytes	**variable field**
221	Vendor Field	Is used to carry information not defined in this standard within a single defined format, so that reserved element IDs are not usurped for nonstandard purposes and so that interoperability is more easily achieved in the presence of nonstandard information	2-256 Bytes	**variable field**
255	Element ID Extension	Extension field for functionality not included in the 255 fields	3-256 Bytes	**variable field**

Note: variable field = the value of the field changes over periods of time on the order of minutes or even seconds, because the field describes the state of the WiFi network and not one or multiple static AP configurations

TABLE II: Beacon fields excluded from the APID

manufacturers are variable over time, which is also a motivating factor not to include them in the APID.

Besides the vendor fields the actual AP manufacturer includes in the beacon, there are vendor fields from other organizations that can be present in the beacon frame. A vendor field that we identified in our dataset is the Microsoft assigned field with organization identifier **00:50:F2**. Under this organization identifier we find the WEP, WPA and WPA2 secure access protocols. This field is invariant and does not contain private data, therefore we include it in the APID.

In **Table II**, **Table III** and **Table IV** we list all the fields we exclude from the APID, respectively, all the fields we include. We construct the APID by concatenating the fields in **Table III** and **Table IV** into a byte string.

C. Analysis

To determine the performance of our authentication method we compute the *precision*, *recall*, *false match rate* (FMR) and *false non-match rate* (FNMR). Those are common metrics used to benchmark classification methods, and are present in the related literature [9], [6], [16]. To define the metrics we first consider a profile $\mathbb{S}$, which is composed of multiple scans $\mathbf{S}$. We then define the function *locate* that returns the true physical location of a scan or profile, as a unique label that contains no information about the geographical location. If $locate(\mathbf{S}) = locate(\mathbb{S})$ then the measurements in the scan and the measurements in the profile were conducted at the same location. We now define $\mathbb{L}$ as a set of test scans, which were conducted at the same physical location as $\mathbb{S}$, and $\mathbb{U}$ (also called remote scans) as a set of test scans which were conducted at different locations, not overlapping with the location of $\mathbb{S}$. Further, we use C to denote our WiFi-based location classifier. When applying C to a profile $\mathbb{S}$ and a scan $\mathbf{S}'$, then C associates $\mathbb{S}$ a label L and $\mathbf{S}'$ a label L' and return if $L = L'$.

With the above premise we now define: true positives(TP), false positives(FP), true negatives(TN) and false negatives(FN). For a given $\mathbf{S}$ and $\mathbb{S}$ if their measurements were collected at the same location ($locate(\mathbf{S}) = locate(\mathbb{S})$) and C returns $L = L'$, then the classification result is considered a true positive, because the classifier correctly identified that the locations are the same. In the same manner, given $\mathbf{S}$ and $\mathbb{S}$, we define the result as a false positive when $locate(\mathbf{S}) \neq locate(\mathbb{S})$ and $L = L'$; a true negative when $locate(\mathbf{S}) \neq locate(\mathbb{S})$ and $L \neq L'$; a false negative when $locate(\mathbf{S}) = locate(\mathbb{S})$ and $L \neq L'$.

Using the above definitions, we describe our evaluation metrics as follows:

$$Precision = \frac{TP}{TP + FP}. \tag{8}$$

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

$$FMR = \frac{FP}{FP + TN}. \tag{10}$$

$$FNMR = \frac{FN}{TP + FN}. \tag{11}$$

We now calculate those metrics for our classifier C. We take our dataset and group it by location. As mentioned, our dataset consists of 33 non-overlapping locations. We take each location and group the measurements into scans, where a scan is comprised of the measurements collected over a time period of n seconds (scan length). A lower value for n has the advantage of making the classification faster, mainly because we require less time to collect the necessary data, but also it takes the classifier less time to process it. The disadvantage, however, is that less data may worsen the performance of the classifier. To select the best value for the scan length, we compute the precision and recall for different values of $\mathbf{n}$ and use them as guidelines.

Further, we group the scans at each location in two sets, one that will be used for training the classifier (profile $\mathbb{S}$) and one that will be used for testing the classifier(local test set $\mathbb{L}$). For statistical consistency we pick the same length (number of scans m) for $\mathbb{S}$ and $\mathbb{L}$. We also want to lower the length of the profile as much as possible to make profile data collection in real world application (e.g. authentication) less time consuming. The same trade-off applies as for the scan length, if the profile

Field ID	Field name	Description	Field length	Minimum Entropy
1	Supported Rates	Specifies any combination of up to eight BSS membership selectors and rates in the OperationalRateSet parameter	3-10 Bytes	**2.87 bits**
32	Power Constraint	Indicates the maximum transmit power	1 Byte	**0.5 bits**
35	TPC Report	Contains transmit power and link margin information sent in response to a TPC Request element	4 Bytes	**3.53 bits**
45	HT Capabilities	Contains optional information for High-Throughput functionality for stations that support it	28 Bytes	**6.96 bits**
48	RSN	Contains the information required to establish a secure connection to the network	0-256 Bytes	**2.55 bits**
50	Extended Supported Rates	Specifies the rates in the Operational Rate Set parameter	3-256 Bytes	**2.07 bits**
51	AP Channel Report	Contains a list of channels where a device is likely to find an AP	3-256 Bytes	**3.71 bits**
52	Neighbor Report	Contains information about APs in neighboring BSS	15-256 Bytes	**5.27 bits**
54	Mobility Domain	Used by the AP to advertise that it is included in the group of APs that constitute a mobility domain, to advertise its support for FT capability, and to advertise its FT policy information	5 Bytes	**3.69 bits**
59	Supported Operating Classes	Is used by a device to advertise the operating classes within which it is currently configured to operate.	3-256 Bytes	**2.99 bits**
66	Measurement Pilot Transmission	Contains the measurement pilot interval	3-256 Bytes	**0.11 bits**
67	BSS Available Admission Capacity	Is helpful for roaming devices to select an AP that is likely to accept future admission control requests	4-256 Bytes	**1.0 bits**
70	RM Enabled Capability	Signals support for radio measurements in a device	7 Bytes	**3.35 bits**
74	Overlapping BSS Scan Parameters	Used by an AP to indicate the values to be used by BSS members when performing OBSS scan operations	16 Bytes	**0.65 bits**
107	Interworking	Contains information about the interworking service capabilities of a device	3-11 Bytes	**2.6 bits**
113	Mesh Configuration	Is used to advertise mesh services	7 Bytes	**2.0 bits**
191	Very High Throughput Capabilities	Is used to advertise network communication capabilities	12 Bytes	**5.0 bits**
192	Very High Throughput Operation	Is used to advertise network communication capabilities	5 Bytes	**2.92 bits**
200	UPSIM	Is used to advertise power saving capabilities	1-33 Bytes	**1.5 bits**

TABLE III: Beacon fields included in the APID

Field ID	Field name	Description	Field length	Minimum Entropy
221	Vendor Field	Is used to carry information not defined in this standard within a single defined format, so that reserved element IDs are not usurped for nonstandard purposes and so that interoperability is more easily achieved in the presence of nonstandard information	2-256 Bytes	**3.01 bits**
231	Sectorized Group ID List	Includes the information necessary for a receiving device to determine its sectorization group membership	2.5-256 Bytes	**1.5 bits**
	Capabilities	It is used to indicate requested or advertised optional capabilities	2 Bytes	**3.32 bits**

TABLE IV: Beacon fields included in the APID

length is too small the performance of the classifier may worsen. To select a suitable profile length we compute the precision and recall for different values for *m* and use them as guidelines.

To calculate the similarity threshold $T(JSR; \mathbb{S}; \mathbb{U})$ we need two other thresholds, LSL and RSL. First, we want to calculate LSL. For this we require the threshold *d*($\mathbb{S}$), which we calculate using Equation 4 from a profile. Once *d*($\mathbb{S}$) is calculated, we can determine the JSR for each scan in $\mathbb{L}$ using Equation 1 and take the minimum JSR value as our LSL, as defined in Equation 5.

As the next step we compute the RSL. For each location in our dataset we choose at random, from the pool of scans from the other 32 locations, a number of scans equal to 2*m*. We then split those scans into a remote set $\mathbb{U}$ of *m* scans for training and a remote set $\mathbb{R}$ of *m* scans for testing. Finally, we use Equation 6 to determine the RSL value.

After acquiring the LSL and the RSL we calculate the threshold $T(JSR; \mathbb{S}; \mathbb{U})$ by averaging the sum of the LSL and the RSL. We use T to classify each scan in $\mathbb{L}$ and compute the TP, FN and the recall for each location. We then use T to classify each scan in $\mathbb{R}$ and obtain the FP, TN and compute the precision. Also, for each location we do a Monte Carlo cross validation step for the remote scan set. This step consists of repeating the steps to choose the remote scan set 100 times (we choose a number that deemed sufficiently large for testing) and computing the TP, FP, TN, FN, precision, recall, FMR, FNMR. We do this to validate that the authentication has a high recall for any randomly chosen $\mathbb{U}$ and $\mathbb{R}$.

D. Experimental Evaluation

In our analysis we compute the precision and recall for different values for *m* and *n*. We start with the configuration $m = 60$min and $n = 60$sec and decrease the values to see how it affects the precision and recall. From our experiments we managed to obtain a precision of **1.00** and a recall of **1.00** for values up to $n = 30$sec and $m = 30$min, as shown in Figure 4. If we further decrease the profile or scan length the performance also drops. We show in **Figure 5** results for $n = 15$sec and $m = 15$min. For precision we have the minimums of **0.9672** for location 29, **0.9677** for location 32 and **0.9836** for location 27. For recall we see better results, with only two locations dropping to values of **0.983**. We deem the precision and recall for this configuration insufficient for continuous authentication applications, because the impact of a FP is too severe, so the precision should not be less than **1.0**. A FP indicates that an authentication request that should be denied is instead granted. Thus, we recommend using the $n = 30$sec and $m = 30$min configuration to avoid FPs. Also, for a scan length of 15 seconds, we look into increasing the profile length to give the classifier more data in an effort to increase the precision. Even with a 60 minutes

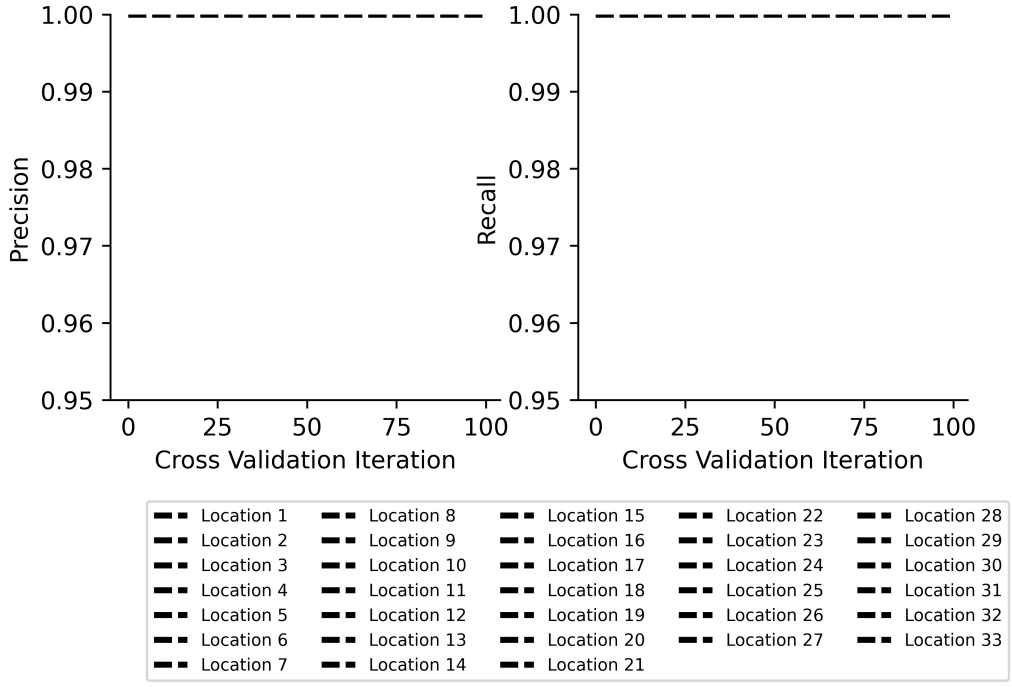

Fig. 4: Classifier precision when m = 30min, n = 30sec, over 100 cross validation runs

profile, we see in **Figure 6** that the precision drops to **0.983** for location 21, to **0.991** for location 16 and to **0.995** for locations 2 and 3. For $m = 60$min we also need 60min of data for the test set. However, we are limited by our dataset, which contains only 22 locations with that amount or more, as shown in **Figure 6**.

We conclude that a scan length smaller than 30 seconds does not contain sufficient data for the classifier to achieve adequate precision.

4. Security analysis

The security of the authentication method is given by its resilience against adversaries. An authentication system has to be secure against different adversaries that try to gain illegitimate access to the system posing as a valid user, known as masquerading. There are multiple scenarios in which an adversary can try to gain access. We look into the chances of an adversary to guess a valid fingerprint, an adversary attacking the communication channel and an onsite adversary. One method for an attacker to gain access is to brute force a valid scan of a user, this scenario allows us to conduct an analysis on how hard it is to guess a valid fingerprint. The hardness of this challenge is given by the amount of information contained in a scan.

In information theory, Shannon's entropy is a measure of information in a random variable. We apply **Equation 12** to calculate the Shannon entropy in a scan using our dataset.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \qquad (12)$$

In the above formula, $H(X)$ represents the entropy of the random variable X. The values x_1 to x_n represent the possible values the random variable X can take. $P(x_i)$ is the probability with which X takes the value x_i.

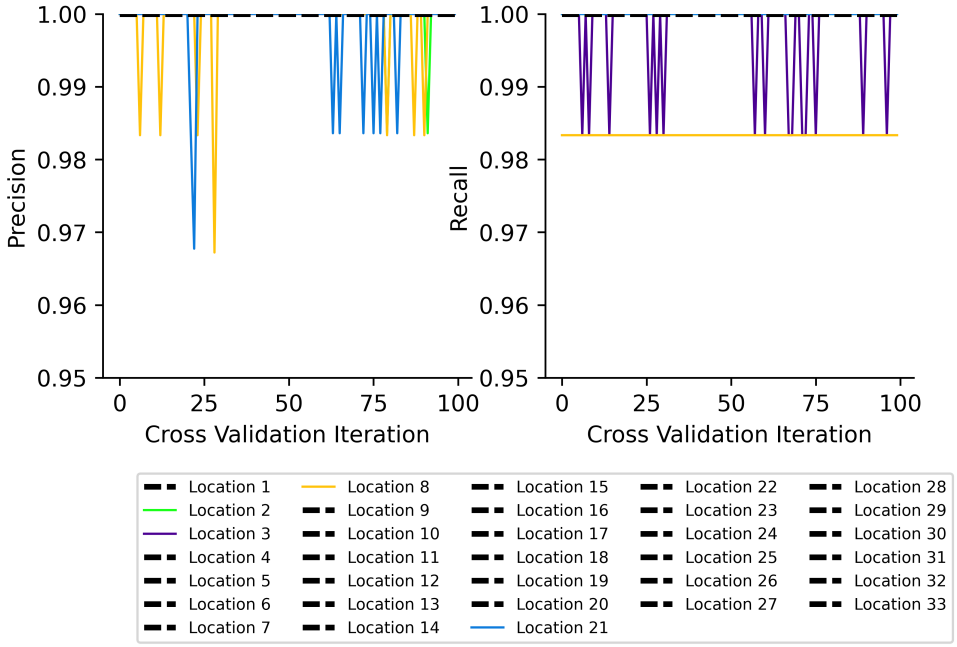

Fig. 5: Classifier precision when m=15min, n=15sec, over 100 cross validation runs

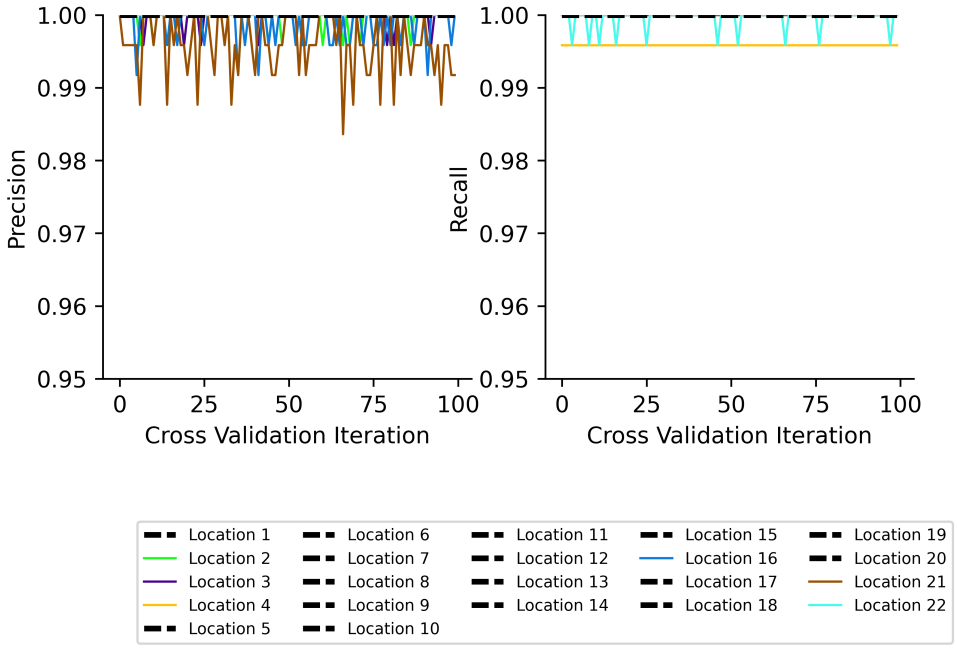

Fig. 6: Classifier precision when m=60min, n=15sec, over 100 cross validation runs

The entropy of a scan is given by the APIDs in the scan. With our limited dataset we cannot compute the exact entropy of an APID (*H(APID)*), but we can calculate an upper bound and a lower bound for the entropy.

The APID is build of multiple fields, where each field can be considered a random variable. The entropy for a set of random variables is given by Shannon's Joint Entropy formula:

$$H(X_1, ..., X_k) = -\sum_{x_1 \in \mathbf{X}_1} ... \sum_{x_k \in \mathbf{X}_k} P(x_1, x_2, ..., x_k) \cdot \log_2 P(x_1, x_2, ..., x_k) \tag{13}$$

In the Joint Entropy formula, $\mathbf{X}_1$ to $\mathbf{X}_k$ represent the sets of values that the random variables X_1 to X_k can take. A property of the joint entropy is that it is always lower or, at most, equal to the sum of the entropy values of the comprising random variables:

$$H(X_1, ..., X_k) \leq H(X_1) + H(X_2) + ... + H(X_k) \tag{14}$$

This implies that the entropy of the APID is less or equal to the sum of the entropies of the comprising fields. We calculate the entropy of a field using our dataset and **Equation 12**. More precisely, we have $P(x_i)$ equal to the number of times field i has value x_i in our dataset divided by the total number of APs. At last, by summing the entropies of the fields given in Table III and Table IV we obtain **51.97** bits as the upper bound for *H(APID)*, as given by **Equation 14**.

The lower bound for *H(APID)* is given by the Joint Entropy (**Equation 13**) of the fields estimated using our dataset. Here $[x_1, x_2, ..., x_k]$ is a unique combination of fields, or an unique APID, in our dataset, and $P(x_1, x_2, ..., x_k)$ is the probability with which that APID appears. We calculate the probability by dividing the number of time the APID appears by the total number of APs. As a result, we obtain a lower bound of **8.99** bits for the entropy.

However our lower and upper bounds for *H(APID)* are dependent on the size of the dataset. In order to make a more precise estimate we built another set of scans [17] from real world measurements collected during walking and cycling around the city. Collecting measurements while travelling over extended areas greatly increases the set of unique APs our new dataset contains. Adding [17] to our 33 locations dataset we increase the number of unique APIDs to **10560**. Using the new dataset the upper bound is now **76.53** and the lower bound is now **10.83**. Also, we show the Shannon entropy for each field, using the new dataset, in **Table III** and **IV**. We observe a significant increase in the entropy as our dataset increases in size.

To get a better approximation of the entropy we use the joint entropy formula and the definition of joint entropy entropy of two or more independent random variables. Two random variables X_1 and X_2 are independent if the value of X_1 does not influence the value of X_2, or otherwise (i.e. knowing the value of X_1 does not give any additional information about X_2). Using domain knowledge of the WiFi protocol we can separate fields into sets that are independent. Given two independent sets $\mathbf{G}_1$ and $\mathbf{G}_2$ (i.e. every field in $\mathbf{G}_1$ is independent from every field in $\mathbf{G}_2$), we have:

$$H(\mathbf{G}_1, \mathbf{G}_2) = H(\mathbf{G}_1) + H(\mathbf{G}_2)$$

Because the two sets are comprised of a smaller part of the total fields in the APID, the maximum entropy they can hold is lower, so they require less data to approximate their entropy.

To split the APID fields into independent groups we need to use domain knowledge. Our split of the fields into independent fields consists of two sets. $\mathbf{G}_1$ contains the Supported Rates, Power Constraint, TPC Report, HT Capabilities, Extended Supported Rates, Measurement Pilot Transmission, RM Enabled Capabilities, VHT Capabilities, VHT Operation, UPSIM, and $\mathbf{G}_2$ contains the remaining fields from **Table III** and **IV**. In $\mathbf{G}_1$ we included all the fields that contain information about the transmission rate and performance capabilities of the AP, or information that is indirectly linked to the transmission capabilities, like the AP's power constraints or the physical medium state. $\mathbf{G}_2$ contain fields that carry administrative information regarding the network structure (e.g. AP Channel Report, Neighbor Report, Mesh Configuration), how to establish a secure connection (e.g. RSN, Vendor Fields) or how

and when to transmit packets (e.g. Measurement Pilot Transmission). The information contained in $\mathbf{G}_1$ is independent from the information on $\mathbf{G}_2$ according to our domain knowledge gathered from [10].

We use the $\mathbf{G}_1$, $\mathbf{G}_2$ grouping to recalculate a lower bound for the entropy of an APID. We now have the following results:

$$H(\mathbf{G}_1) = 7.70 \text{ bits}$$

$$H(\mathbf{G}_2) = 9.89 \text{ bits}$$

$$H(\mathbf{G}_1, \mathbf{G}_2) = H(\mathbf{G}_1) + H(\mathbf{G}_2) = 17.59 \text{ bits}$$

We obtain a much higher entropy using independent sets of fields. Our results show a significant increase over the entropy obtain by the authors of [6], which is **9.1 bits**. This is a direct implication of having access in userspace to more fields on Windows than on Linux. In locations where multiple APIDs are detected in a scan, the entropy is equal to the number of APIDs times the entropy of one APID.

To evaluate the security of the method against brute force attacks we use as reference the NIST [18] entropy requirements for authentication methods. More precisely, we look into the memorized secrets category, for which the requirement is a minimum of 8 characters from the printable ASCII character set. The printable ASCII character set contains 96 characters, thus, the minimum entropy for a memorized secret is $\log_2(96^8) = 52.67$bits. Therefore, we recommend using our method in locations where at least three APs (which will give a minimum entropy of **52.77**), in order to be compliant with the NIST standards.

Another scenario that we look into is an adversary compromising the communication channel between a user and the server that authenticates the user. In order to secure the communication we recommend that authentication is only initiated after an encrypted communication is established between the user and the server. An encrypted channel using TLS is used in most communications over the Internet and such a channel can also be leveraged for our authentication method. A TLS encrypted channel guarantees the user the authenticity of the server and also protects against Man-in-the-Middle attacks and Reply attacks.

Another scenario is protecting against people that have access to the same physical location as the user. In the case of an office building other people that have access are colleagues. We do not know how big the distance between two users must be such that our method can distinguish their scans as being at distinct locations. To protect in such scenarios we recommend a first and second factor for authentication, and then use our method only as additional factor (e.g. for continuous authentication).

5. Comparison with Related Work

In the literature there are other attempts for continuous authentication methods. Most of the methods that are presented in the literature are based on behavioural biometrics, e.g. physical movements of the hand when using a smartphone [16] or finger movements on a touchpad [19]. The authors of [16] use readings from accelerometer, gyroscope, and magnetometer to model user behaviour for authentication. On the other side, the authors of [19] propose a method for embedded devices to determine finger movements from the WiFi channel state information (CSI).

Another approach to continuous authentication explored by the authors of [20] targets IoT devices that do not present peripherals for user input. Their method uses CSI to infer daily human behavioural patterns.

The domain of location based authentication using WiFi was explored in [21] and [22]. The authors of [21] build a template of the WiFi networks that are detected by the mobile phone of the user during a 24h period, in slots of t minutes. For a 30 minutes slot period they have a 48 slots template. Each slot consists of the MAC addresses of the WiFi networks and a score given by the frequency with which each network appear in the same slot during a day, on a 30 day period. Additional to the WiFi location, accelerometer data collected over a period of three seconds is used to identify user mobile phone movement patterns when opening an application.

Method	False Acceptance Rate	False Rejection Rate	Precision	Recall	Uses privacy preserving features
Our Method (n = 30 sec, m = 30 min)	0	0	1.00	1.00	TRUE
MineAuth [22]	-	-	0.982	0.983	FALSE
[21]	0.091	0.091	-	-	FALSE
[16]	0.0095	0.0667	-	-	FALSE
[6] (n = 60 sec, m = 120 min)	-	-	0.98	0.975	TRUE

TABLE V: Comparison of experimental results from related research in the literature

Using readings from the two sensors the authors of [21] authenticate the user. The experiments in [21] show the method has a low performance given by an EER of **0.091**. This means that the false acceptance rate (FAR) is 0.091, which is too high for practical authentication applications.

In [22] the authors also use WiFi to identify the location of the user, but they combine the WiFi data with data from more sensors to increase the performance. They use the reading from WiFi, bluetooth, GPS sensors, and usage data (e.g. calls, SMS) as identify behavioural patterns and location for authentication, based on a user's profile. [22] experiments show a precision of **0.982** and a recall of **0.983** which are good.

To compare our work with the literature we compute for our method the metrics used in other research papers and present the results in **Table V**. We see that our method outperforms all the previous methods from the literature that use WiFi sensor readings for authentication. It also outperforms [6], which is an expected results given that on Windows we have access to more fields than on Linux, which ,in term, increases the entropy of an APID. More over, we managed to decrease the scan size to 30 seconds and the profile size to 30 minutes with minimum impact on performance. This decrease in parameters' size makes the method more user friendly and further increases the probability of adoption. However, we still require half a minute for a fingerprint. Whether this is tolerable must be decided per use case.

6. Conclusions

In our research we managed to build a method that can be used for continuous authentication, based on the research from [6], for the widespread operation system Windows and using the ubiquitous WiFi infrastructure. To develop this method we devised a way to access WiFi data using the Windows C/C++ Native WiFi API. Using the Native WiFi API a user does not need privileged access rights, which is an essential requirement for authentication. We also made preserving the privacy of the user an important requirement in our research and excluded all WiFi data which could classify as PII.

By using the Native WiFi API we have access to the entire WiFi discovery beacon frame which gives us more capability information compared to [6]. This leads to a higher minimum entropy of **17.59 bits** compared to the entropy obtained in [6], **9.1 bits**. As a result of higher entropy, we are able to identify locations based on WiFi data collected over smaller periods of time. We built a proof-of-concept implementation that gives better results than other methods in the literature. We also compare it to [6], and we obtain a similar performance with a four times smaller profile length (i.e. from 120 minutes to 30 minutes), and the scan length also two times smaller (i.e. from 60 seconds to 30 seconds). With lower scan length, continuous authentication can be performed more often. This decreases the

attack windows in which an adversary can exploit a hijacked session key or physical device with an active session. Also, smaller scan and profile length makes our method less onerous for users, which subsequently increases the probability of adoption.

A. Research Limitations

A limitation of our research is the lack of data to better evaluate the performance. Because the online datasets did not match our requirements we decided to collect our own data. This dataset contains 33 locations on which we approximate the capabilities of our method. A larger dataset will give us a better approximation of the authentication performance and also the entropy in the WiFi discovery beacon frame.

Another limitation of our method is being OS specific. On most devices the cumbersome task of handling the low level network configurations is taken away by the operating system. This also comes with limited access to user applications which want to take advantage of the meta data that the network protocols use. To make the method accessible to more devices we need to adapt it for different platforms (e.g. MacOS, mobile OSs).

B. Future work

To further improve our results we identify a few directions for future research. In our method we calculate the RSSI threshold $d(\mathbb{S})$ based on the profile, as the authors of [6] do. Another method that might improve the performance is calculating $d(\mathbb{S})$ for each AP in a profile. This approach aims to obtain a precise mapping of the behaviour of each individual AP in a location and reduce the probability of a false positive.

Another direction for future research that we identify is examining the beacon fields that we excluded from the APID. Most of the fields were excluded because their values variate overtime. This volatility might be restricted to only part of those fields and other parts might present stable values as long as the AP is not reconfigured. If this hypothesis is correct, it may be possible to include information from more fields in the APID, which will lead to a higher entropy per AP and thus a better performance.

REFERENCES

[1] "The U.S. Bureau of Labor Statistics' Information Security Analyst's Outlook. Statistics on Cyber Security jobs." [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/information-security-analysts.htm#tab-6

[2] "ISO/IEC 27000 - Information technology — Security techniques — Information security management systems - Overview and vocabulary," International Organization for Standardization, Geneva, CH, Standard, 2018.

[3] W. Barker, "Guideline for identifying an information system as a national security system," 2003. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-59.pdf

[4] LastPass, "Global Password Security Report - Emerging trends in credential management, access and authentication in businesses worldwide." 2019. [Online]. Available: https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LMI0828a-IAM-LastPass-State-of-the-Password-Report.pdf

[5] T. O. of Management and Budget, "Annual report to congress on the federal information security management act," 2017. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2017/11/FY2017FISMAReportCongress.pdf

[6] P. Jakubeit, A. Peter, and M. van Steen, "Wifi fingerprinting as nonintrusive authentication factor," in *Emerging Technologies for Authorization and Authentication (ETAA) International Workshop*, 2022.

[7] S. McKenna and S. Gong, "Non-intrusive person authentication for access control by visual tracking and face recognition," in *International Conference on Audio- and Video-Based Biometric Person Authentication*, 1997, p. 177–183.

[8] J. Ranjan and K. Whitehouse, "Object hallmarks: Identifying object users using wearable wrist sensors," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, p. 51–61.

[9] L. Zhang, C. Wang, and D. Zhang, "Wi-pigr: Path independent gait recognition with commodity wi-fi," in *IEEE Transactions on Mobile Computing*, 2021, pp. 3414–3427.

[10] IEEE, "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, 2021, pp. 1–4379.

[11] "Network manager's accesspoint data structure documentation." [Online]. Available: https://developer-old.gnome.org/NetworkManager/1.2/gdbus-org.freedesktop.NetworkManager.AccessPoint.html

[12] "Scanning tool server source code." [Online]. Available: https://github.com/victorciresica/wifi-scanner

[13] "Scanning tool interface source code." [Online]. Available: https://github.com/victorciresica/wifi-scanner-frontend

[14] "C program for collecting beacon frames in cap files using the windows native wifi api." [Online]. Available: https://github.com/6e726d/Native-WiFi-API-Beacon-Sniffer
[15] "Python c api." [Online]. Available: https://docs.python.org/3/c-api/index.html
[16] M. Abuhamad, T. Abuhmed, D. Mohaisen, and D. Nyang, "Autosen: Deep-learning-based implicit continuous authentication using smartphone sensors," in *IEEE Internet of Things Journal*, 2020, pp. 5008–5020.
[17] "Wifi measurements dataset collect throughout the city of bucharest." [Online]. Available: https://github.com/vciresica/wifi-research
[18] "NIST. Digital identity guidelines, authentication and lifecycle management." [Online]. Available: https://pages.nist.gov/800-63-3/sp800-63b.html
[19] H. Kong, L. Lu, J. Yu, Y. Chen, and F. Tang, in *Continuous Authentication through Finger Gesture Interaction for Smart Homes Using WiFi*, 2020, pp. 3148–3162.
[20] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, W. Xu, and L. Su, "Towards environment independent device free human activity recognition," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, p. 289–304.
[21] G. Li and P. Bours, "Studying WiFi and Accelerometer Data Based Authentication Method on Mobile Phones," in *Proceedings of the 2018 2nd International Conference on Biometric Engineering and Applications*, 2018, p. 18–23.
[22] X. Pang, L. Yang, M. Liu, and J. Ma, "Mineauth: Mining behavioural habits for continuous authentication on a smartphone," in *Australian Conference on Information Security and Privacy*, 2019, pp. 533–551.

APPENDIX

Location label	Scan Length 15 sec			Scan Length 30 sec			Scan Length 60 sec		
	Min APs	**Max APs**	**Avg APs**	**Min APs**	**Max APs**	**Avg APs**	**Min APs**	**Max APs**	**Avg APs**
Location 1	3	7	4.22	3	7	4.56	3	7	4.93
Location 2	5	31	23.74	12	32	25.66	21	34	27.49
Location 3	7	23	16.60	15	27	19.98	18	28	23.05
Location 4	6	34	25.62	21	41	31.17	27	49	37.30
Location 5	29	52	41.49	38	56	47.43	46	63	53.09
Location 6	24	47	34.74	29	56	40.73	38	66	47.58
Location 7	5	30	9.23	6	30	9.98	7	31	11.00
Location 8	1	24	14.64	11	25	17.31	15	25	19.35
Location 9	3	19	11.97	7	22	14.65	13	23	17.38
Location 10	10	38	25.46	19	42	29.21	23	48	32.24
Location 11	2	5	3.73	3	5	3.87	3	5	3.92
Location 12	20	35	27.56	23	38	30.86	27	41	34.19
Location 13	8	22	14.29	9	23	16.48	13	25	18.65
Location 14	13	29	18.96	14	29	20.97	17	32	23.38
Location 15	21	52	30.57	25	62	34.60	28	67	39.02
Location 16	11	65	22.21	13	78	25.73	13	94	30.07
Location 17	14	33	23.51	20	40	29.05	26	51	34.27

TABLE VI: Dataset scan time distribution

Location label	Scan Length 15 sec			Scan Length 30 sec			Scan Length 60 sec		
	Min APs	**Max APs**	**Avg APs**	**Min APs**	**Max APs**	**Avg APs**	**Min APs**	**Max APs**	**Avg APs**
Location 18	4	62	47.13	43	69	57.08	30	76	65.38
Location 19	3	6	4.55	4	6	4.59	4	6	4.58
Location 20	8	28	21.09	14	32	24.02	20	35	27.12
Location 21	6	45	33.19	26	53	40.98	6	59	47.50
Location 22	6	14	9.61	7	14	10.35	8	14	10.79
Location 23	16	30	22.41	18	32	24.75	20	34	26.59
Location 24	9	14	11.38	10	14	12.12	11	15	12.67
Location 25	20	53	39.12	35	58	47.23	44	67	56.03
Location 26	17	56	38.55	31	67	47.97	41	84	59.18
Location 27	12	43	31.40	17	48	36.42	31	54	42.10
Location 28	8	18	12.37	10	18	13.16	11	19	14.20
Location 29	13	32	24.50	20	37	28.11	25	40	32.02
Location 30	1	4	2.98	1	4	3.03	1	4	3.05
Location 31	1	9	4.52	3	9	5.11	3	11	6.03
Location 32	18	30	23.25	21	33	26.06	24	36	29.27
Location 33	24	71	52.35	29	82	64.15	50	101	77.40
Summary	**1**	**71**	**22.03**	**1**	**82**	**25.68**	**1**	**101**	**29.42**

TABLE VII: Dataset scan time distribution