IEEE 802.11 Chipset Fingerprinting by the Measurement of Timing Characteristics

Guenther Lackner[1] **Peter Teufl**[1]

[1] Institute of Applied Information Processing and Communications (IAIK)
University of Technology Graz
Inffeldgasse 16a, 8010 Graz, AUSTRIA
Email: guenther.lackner@iaik.tugraz.at, peter.teufl@iaik.tugraz.at

Abstract

In this paper we present a technique to create WLAN device fingerprints by measuring timing properties without the use of special-purpose hardware. Our proposed process is absolutely passive and cannot be detected by the targeted device. The timing measurement is based on a delay caused by the hardware implementation of the CRC checksum algorithm at the network interface card (NIC) of the client. This delay turned out to be significant for a large number of different chipset implementations. The ability of identifying connected devices could improve the security of a wireless network significantly. It could help to enhance access control mechanisms and would deliver valuable real time information about the connected clients. As a proof of our concept we present a prototype implementation called WiFinger to evaluate our approach.

Keywords: IEEE 802.11, MAC Address Spoofing, passive Chip-set Fingerprinting, Significant Histograms

1 Introduction

During the last years wireless networking spread into countless fields of application like mobile telephony, wireless computer networks, mobile sensor networks, and many more. This wireless revolution daily pervades new areas of our lives providing an increase in the grade of mobility, usability and comfort. But there seems to be a price to pay. Due to their open-air propagation nature, wireless networks raise a new variety of potential security and privacy risks for attackers. During the development and definition of related industrial standards there was obviously not enough focus on security issues. Some of the most popular and widest spread standards in wireless computer communications like WEP (Wireless Equivalent Privacy) are full of security breaches which open up all gates to attackers (Fluhrer et al. 2001). Further on, even state-of-the art standards like WPA2 (Wireless Protected Access based on AES) begin to crumble (Airtight-Networks 2010).

The aim of our work is to bring more safety into the wireless world by identifying network participants via timing measurements. Our approach focuses on the widely spread standards of the IEEE 802.11 family. We do not intend to improve or alter encryption mechanisms. With our tool called WiFinger, one could be able to detect, and in succession prevent layer 2 MAC address spoofing attacks. If unauthorized participation of attackers in a wireless network can be detected many possible attacks could be prevented. Our work is based on a technique of passive fingerprint creation by observing the timing behavior of IEEE 802.11 compliant devices without the necessity of special purpose hardware like frequency spectrum analyzers.

This paper is organized as follows. Section 2 introduces related work. Section 3 describes our method of creating fingerprints of IEEE 802.11 device chipsets. Section 4 introduces the fundamentals of the classification method developed us, based on Self Organizing Maps. Section 5 illustrates the design and implementation details of the WiFinger software. The real-world applicability and performance analysis is placed in section 6. Section 7 provides a short outlook on future extensions and improvements and concludes the article.

2 Related Work

A straightforward approach for device identification is to utilize the device addresses such as the MAC (Media Access Control) address (layer 2) or the assigned IP address (layer 3). This can easily be achieved by analyzing relevant ARP (Address Resolution Protocol) traffic (Plummer 1982). Unfortunately, this approach has a major drawback. Most devices allow to modify their assigned MAC address with easy to use, free software tools. This problem might be tackled by creating fingerprints of network hardware. This would allow the identification of any device by observing its external characteristics.

Remote Physical Device Fingerprinting: One of the most significant papers in the field of device fingerprinting has been published by Tadayoshi Kohno and his team at UC San Diego (Kohno et al. 2005). Kohno developed a method to identify remote devices by exploiting small, microscopic deviations in the hardware: clock skews. By analyzing the deviation of TCP or ICMP timestamps over a certain period of time it is feasible to distinct different hardware clocks and thus different devices. The main difference to our approach is, that Kohno et al. is not applicable in an encrypted wireless environment as it needs plaintext TCP or ICMP payloads for analysis.

Radio Frequency Fingerprinting: This fingerprinting technique is based on the signal characteristics of turn-on transients of wireless transceivers. These transients are specific to each different transceiver and thus are perfectly suited as data source for fingerprint generation. Transient capturing and analysis requires a special infrastructure for signal capturing which is expensive and has

Copyright ©2011, We want to thank the Austrian Science Fund(FWF), who supported this work as part of the project P21936. This paper appeared at the 9th Australasian Information Security Conference (AISC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 116, Colin Boyd and Josef Pieprzyk, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

to be operated by experts. Hall et al. evaluated the performance of the fingerprinting method with 30 transceivers. For each transceiver 120 signals were captured and used for the performance evaluation. The results indicate that the method is capable of achieving a very low false positive rate (0% during the evaluation) and a high detection accuracy (95% during the evaluation). The biggest disadvantage of this method is the special hardware needed for signal capturing which limits the broad deployment. (Hall et al. 2006)

Passive Data Link Layer Fingerprinting: Franklin et al. (McCoy et al. 2006) identified an imprecision in the IEEE 802.11 Media Access Control specifications that has been interpreted differently by wireless NIC firmware developers. The time between sending two so called *beacon frames* used for network detection is not strictly defined. This method is able to classify different firmware versions instead of the underlying hardware. For creating a meaningful fingerprint a large number of probe-requests need to be captured. Due to the fact that a NIC willing to join a network, usually just needs a hand-full of these requests it could take a rather long time to obtain a suitable amount of data. Another significant drawback is that fingerprinting may easily be avoided by using passive-scanning or altering the device firmware. (McCoy et al. 2006) Some improvements to this approach have been developed by Loh et al. (Desmond et al. 2008).

Active Fingerprinting by Timing Analysis: Bartolomiej Siekas work on device fingerprinting (Sieka 2006) is probably the one closest related to our approach. It uses the time that elapses between the first acknowledgement is sent and the moment the authentication response is sent. For classification purpose, support vector machines are used. The drawback of this approach is its limitation to the authentication phase for measurements. As this phase only occurs during the initialization of the connection, Sieka actively needs to provoke the repetition of it by sending specifically crafted 802.11 frames. This could be detected by an intrusion detection system or the device to fingerprint, allowing it to counteract. As the next section describes, our approach is immune against such countermeasures as it is absolutely passive.

3 Fingerprinting on Layer 2

Creating a fingerprint of a device is the process of identifying it by the observation of its external characteristics. We developed one possibility of creating fingerprints of IEEE 802.11 devices by observing their timing behavior. This section provides a compact overview of the basic principles.
Our approach examines the timing behavior of IEEE 802.11 devices generating so called *acknowledge packets (ACK)*. Due to the fact that IEEE 802.11 standards follow the principle of *half-duplex* communication, a *collision avoidance technique* is generally needed to be deployed. If a participant A (client) has sent a data frame to participant B (access point), A is not able to observe if its message was transmitted correctly or collided with a data frame sent from another participant at the same time. IEEE 802.11 standards are based on the so called *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) mechanism (Brenner 1997). In this paper we just shortly describe the IEEE 802.11 media access system. For further details consult (IEE 1999) and (Kerry 2007).
To inform A that its data frame was transmitted

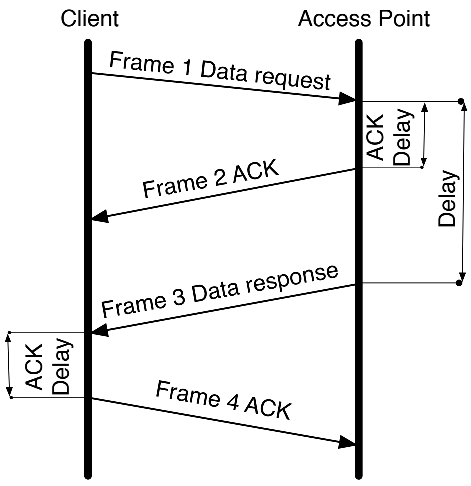

Figure 1: ACK delay

correctly, B generates and transmits an ACK packet after having correctly evaluated the CRC checksum of A*'s* data frame. If the CRC check fails, no ACK will be sent and A retransmits the data frame after a certain time (IEE 1999).

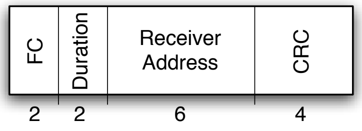

Figure 2: ACK packet

The computing and evaluation of the CRC checksum plus the generation of the ACK packet takes a certain amount of time. This amount depends on the hardware implementation of the CRC algorithm, the firmware and some other components of the used wireless network device. We call this delay *The Acknowledge Delay*. If one regards the distribution of a certain number of ACK delay values the outcome represents a significant property of the used wireless device. This outcome is called *Significant Histogram*.

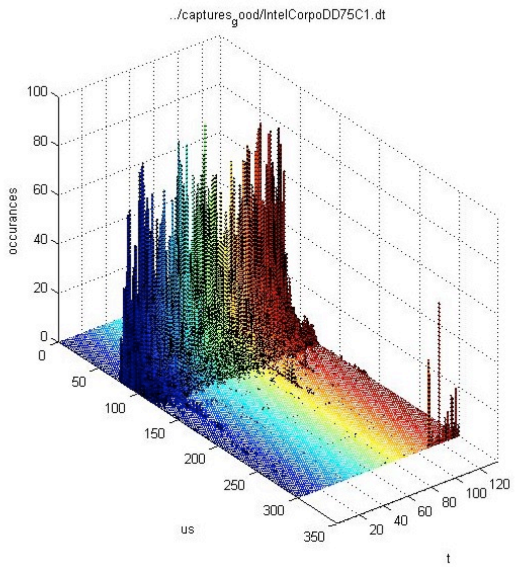

Figure 3: Significant Histograms of 400 ACK delays each, over several time periods t

Based on these Significant Histograms it is possi-

ble to distinct between different IEEE 802.11 device chipsets.
The classification results in lab environment were very promising.

4 Classification

This section describes the classification algorithm based on multiple Self Organizing Maps (SOM) arranged in a tree and implements an extension which improves the quality of SOMs when used for supervised learning.

4.1 SOM based algorithm

Self Organizing Maps (SOM) belong to the broader category of neural networks (Kohonen 2001). They are mainly used for unsupervised learning and the visualization of high-dimensional data. In this paper we employ SOMs for a supervised classifier. Although, other supervised algorithms like neural networks or support vector machines are better suited for classification tasks, we still focus on the SOM due to one main reason: The visual representation of the data in a 2D map allows us to quickly gain insight on the analyzed data (an example is given in Figure 5).

By labeling the SOM units during the training process according to the class labels of the data they represent, the SOM can also be employed for supervised learning. However, due to the unsupervised nature of the SOM, the class information is not taken into account during the training process. Therefore, the accuracy of the trained model might be inadequate for the separation of data belonging to different classes. This data is mapped by the same units and leads to classification errors that decrease the accuracy of the SOM. In order to cope with this issue, our classifier utilizes multiple SOMs arranged in a tree.

Whenever the model of a trained SOM is not precise enough to separate data of different classes accurately, we extract this data, train a new SOM on this data and link the units of the old SOM covering this data to the new SOM. Therefore, we do not need to deal with SOM model complexity manually. If the model of a trained SOM is not accurate enough, the algorithm simply trains a new SOM that is only trained on the data which requires more complex modeling (indicated by a higher misclassification rate).

The multiple SOMs are trained and arranged in a tree according to this algorithm:

1. Train a SOM on the input data
2. Label the units according to the classes they represent
3. Calculate misclassification rates for all classes
4. Extract the data of classes that cannot be separated with an error rate lower than a given threshold
5. Mark the units that cover the extracted data to indicate that the actual classification will be made in the next SOM.
6. Go to step one and train a SOM for new extracted data. Repeat these steps until the error conditions are met or only two classes remain in one SOM.

A simple example with five classes is shown in Figure 4. The first SOM is trained on the complete data set and the misclassification rates are determined. The example shows that classes A/B/C and D/E cannot be separated accurately by the first SOM. Therefore, two data sets for A/B/C and D/E are extracted. For both data sets, new SOMs are trained and the units corresponding to these classes in the first SOM are linked to the newly trained SOMs. In case of A/B/C, the second SOM is able to separate the class C from A/B but the misclassifications rates for A/B are still too high. Therefore, another SOM is trained that increases the classification performance. The picture indicates that the SOM for A/B still has some misclassification errors, which cannot be removed without losing generalization (and thereby overfitting the data).

The trained SOM hierarchy of SOM tree is used for the classification of unknown data in this way:

- Present the data to the root SOM of the tree and determine the best matching unit (BMU)
- If the unit is linked to another SOM further down in the hierarchy, load this SOM and go to the previous step. If the unit is not linked to another SOM, return the class label of the unit.

This procedure is indicated in the example by the two classification paths for data vectors from class B end E.

The described strategy is employed for the classification of the WLAN chipsets. The same technique was already successfully applied to other classification problems, especially for network traffic classification (Payer et al. 2005).
For SOM training, the SOM toolbox (Vesanto et al. n.d.) which is available for Matlab®(*MATLAB - The Language of Technical Computing: Mathworks, http://www.mathworks.com* n.d.) was used. The classification algorithm was also implemented in the tool WiFinger.

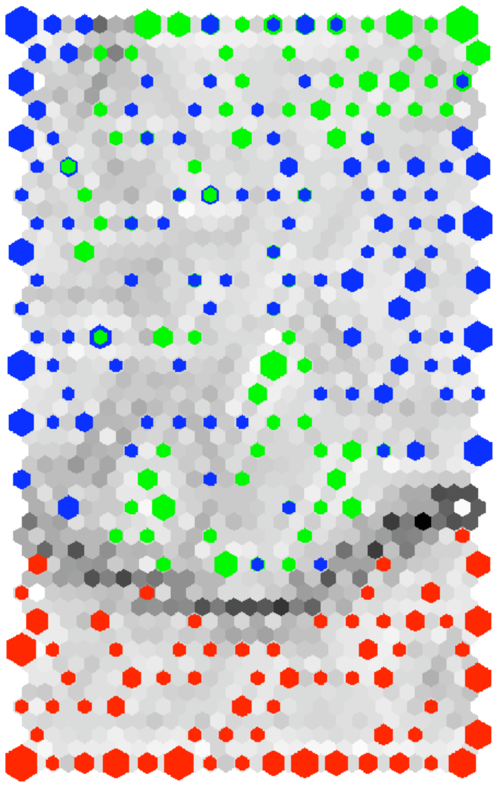
Figure 5: SOM trained with data from three WLAN chipsets. The map helps us to gain a quick insight on the analyzed high dimensional data. In this case we can see at a glance that the "red" chipset can be clearly separated from the other ones.

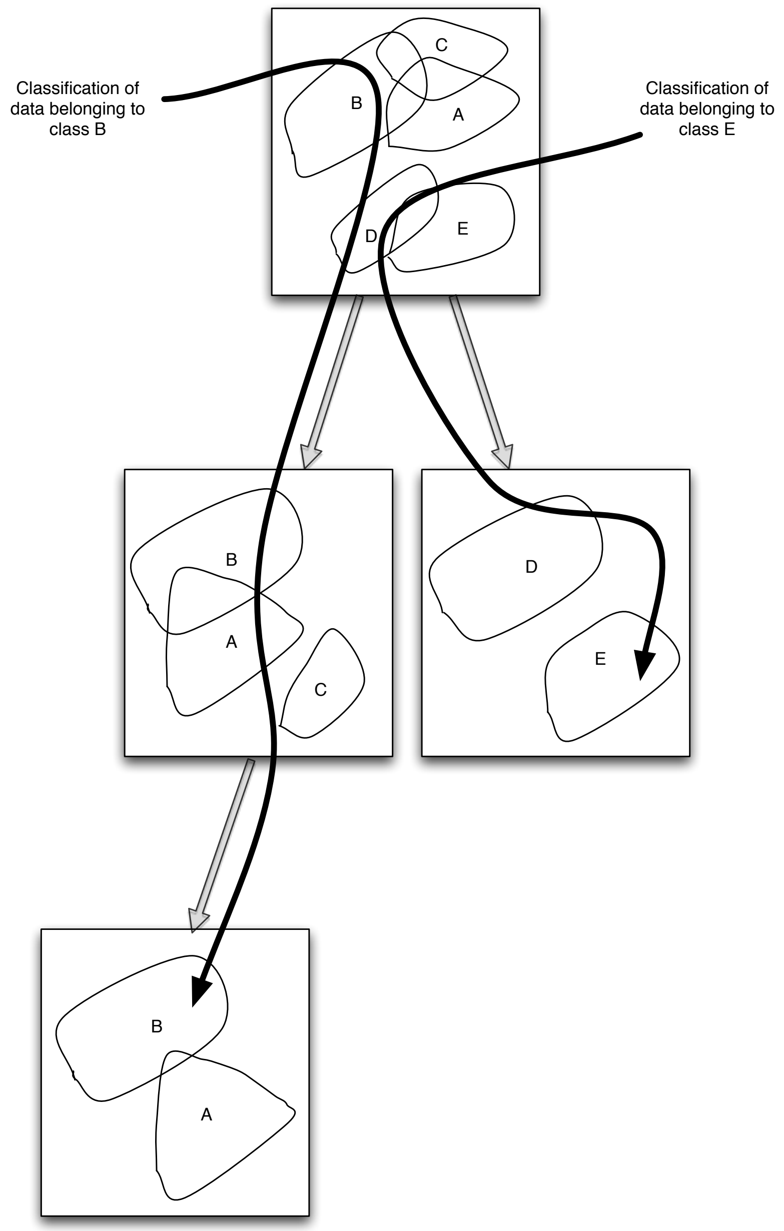
Classification of data belonging to class B
Classification of data belonging to class E
C
B
A
D
E
B
A
C
D
E
B
A

Figure 4: SOM Tree example: The classes A/B/C and D/E cannot be separated within the accepted error threshold, therefore two new SOMs are trained for A/B/C data and D/E data. Another SOM is attached due to a high number of misclassifications for A/B. The two arrows indicate the paths that are used for the classification of unknown data.

4.2 Features

An initial evaluation of WLAN traffic showed us, that the ACK delays of different packets vary from WLAN chipset to chipset and therefore could be used to identify such chipsets. By analyzing the spectrum of the ACK delays of the same chipset we can derive a histogram that represents the number of packets over the various observed delay times. In addition we capture the packet size in order to find out whether the ACK delay also depends on the packet size. The packets of a session – the time frame, where packets of a given chipset are captured – are arranged in the histograms in the following way:

1. Collect the ACK delays for each session of traffic generated by different WLAN chipsets.
2. For each 50 packets, create a 3D histogram which stores the frequency of the packets with a specific ACK delay and packet size. Each histogram is converted into a feature vector used for SOM training and classification.
3. Train a SOM tree with the histograms of the different WLAN chipsets.
4. The trained SOM tree is used for the classification of new traffic.

The length of the feature vectors depend on the number of analyzed ACK delay values (indicated as n) and packet size values (indicated as m). By storing the number of packets for given delay values and packet sizes we gain a 3D histogram that can be converted into a feature vector with $f = n \times m$ entries. In order to keep the feature vectors at a feasible length, we need to map delay and packet size ranges into single values. E.g. by considering ACK delay values from 1 ms to 300 ms ($n = 300$) and packet size values from 1 byte to 1600 bytes ($m = 1600$) we would get feature vectors with $f = 300 \times 1600 = 480000$ entries, which is not feasible. However, this resolution is not needed and on the contrary would decrease the accuracy of the classifier. Therefore, we reduce the number of features by mapping several ACK delay values and packet size values into bins representing value ranges. E.g. if we use a bin size of 10 ms for the ACK delay (then $n = 30$) and a bin size of 40 bytes for the packet size (then $m = 40$) the feature vector length is reduced to $f = 30 \times 40 = 1200$.

In Figures 6 and 7 two histograms based on delay information only (packet size is ignored) are shown. We observe that there is a significant difference between the analyzed chipsets. The role of the packet size combined with the delay values is visualized in Figures 8 and 9. Here we observer that the ACK delay values also depend on the packet size – at least for certain chipsets. In Figure 8 the captured data of the Agere chipset clearly shows that there is such a dependence. In contrast this dependence cannot be observed when analyzing the Edimax chipset (Figure 9).

By integrating both features into the classification process we are able to increase the accuracy compared to classifiers based on the delay information only. For further details and evaluation results we refer to the results section.

5 WiFinger

As proof of concept of our approach, a small linux command line utility called WiFinger was developed and implemented in C/C++. The implementation was kept small in anticipation of possible use on handheld PCs as passive scanning devices. Additionally to

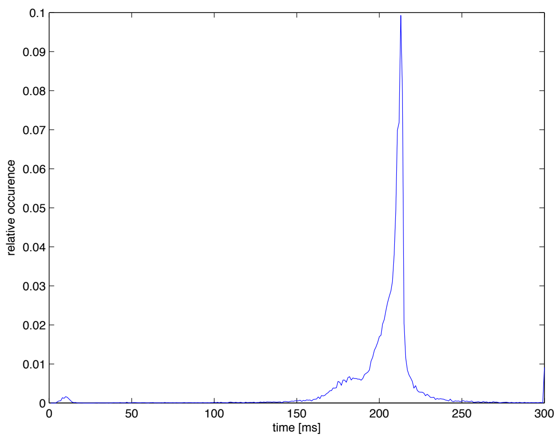

Figure 6: Delay histogram without packet size for an Orinoco chipset

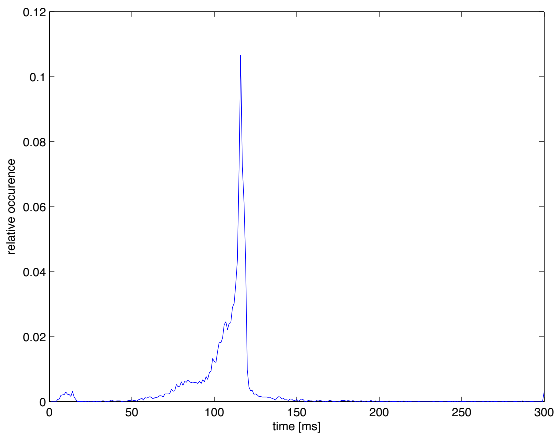

Figure 7: Delay histogram without packet size for a Broadcom chipset

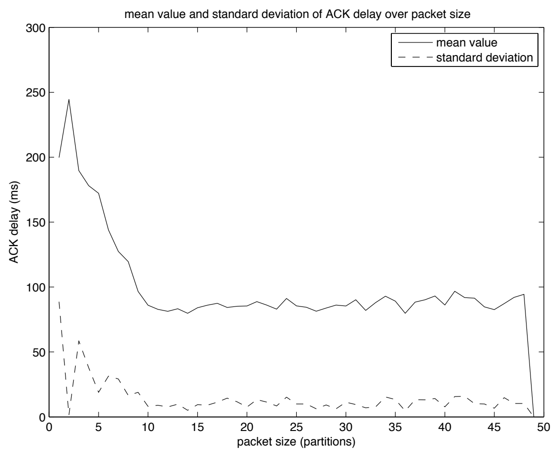

Figure 8: Agere (Chipset 2): Dependency between packet size and ACK delay values

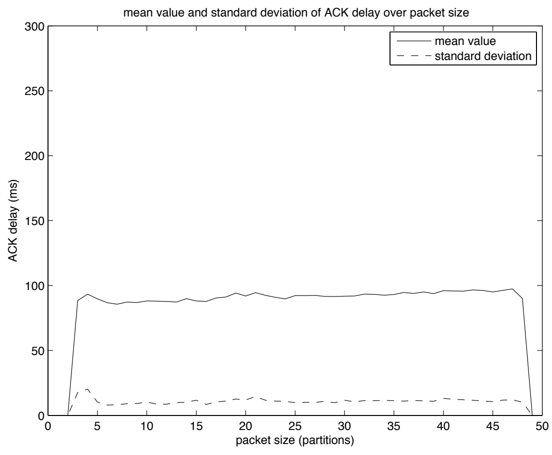

Figure 9: EdimaxTech (Chipset 6): Here, we cannot observer a dependency between the packet size and the ACK delay values

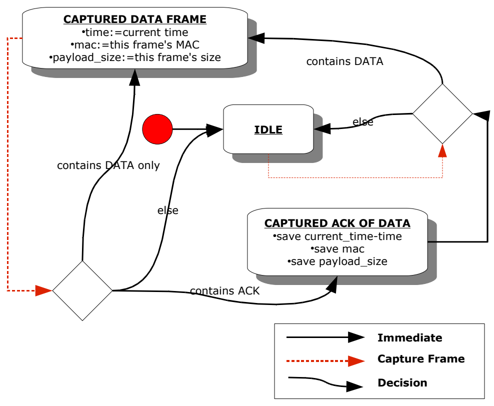

Figure 11: WiFinger Feature Measurement FSM

our code just a small number of libraries was used(*libpcap*[1], *libSom*(Payer et al. 2005) and *ncurses*).

Two possible modes are provided by the application. Frames may either be captured live from the wireless network or be loaded from a previous capture-session file in *libpcap* format. Only the first 24 bytes of each IEEE 802.11 frame i.e. only the frame header bytes are examined. The payload itself plays no role in the classification task but the overall packet size does because of its influence to the CRC processing time. The architecture allows the collection and analysis of data on distributed devices.

5.1 Data Processing

Figure 10 shows how data is processed by WiFinger. After *capturing* and *feature measurement* the measured delay and associated host information is handled in two steps. First, the collected data is saved unfiltered onto the hard-disk and converted in a format that could be imported by Matlab®. Second, the data is filtered and added to the *Significant Histogram*. Hosts are distinguished by their MAC address. The accuracy of the *Significant Histogram* increases with time. Per default, classification is run every 1000 measurements.

5.2 Feature Measurement

Figure 11 illustrates the process of feature measurement. Depending on the type of the captured frame, one of three states is entered: IDLE, CAPTURED DATA FRAME and CAPTURED ACK OF DATA. Recognized types are data frames and frames containing an ACK. During *contention free periods*, DATA frames can contain *contention free acknowledgments*, hence referred to as CF-Ack. During these periods DATA frames with embedded CF-Acks may appear in direct succession of each other. Thus as a special case, the states LAST FRAME WAS DATA and DATA/ACK PAIR CAPTURED can be entered during the same pass. MAC addresses are read from the frame header's *address 1 field* which always contains the wireless destination station and the *address 2 field* which always contains the sending wireless station (IEE 1999). Both fields are 6 bytes long and start at byte 4 and 10 respectively.

The destination station's MAC address, payload size and the time of reception of the latest data frame are saved in temporary variables. If the frame acknowledges the immediately previously sent data packet, the *acknowledge delay* is measured as time between the reception of the last data frame and the reception of the acknowledging frame (see Figure 1). Note that interval is longer than the *Shortest Inter-Frame Space*, since the delay of receiving the data frame is added.

Measurements on broadcast addresses are discarded. Since an ACK frame carries only a destination address (see figure 2), it is possible that on-air data is missed and a later ACK frame is mistaken for an expected acknowledgment. To minimize such mistakes, the sending stations address of the previous data frame is checked against the destination address of the following ACK frame. This only works outside of *contention free periods* since the destination address of frames containing CF-Ack does not need to match the address of the previously transmitting station. The following frame types are relevant during *contention free periods* (IEE 1999):

- CF-End + CF-Ack
- Data + CF-Ack
- Data + CF-Ack + CF-Poll
- NoData + CF-Ack
- NoData + CF-Ack + CF-Poll

5.3 Usage of Matlab®for SOM training

Figure 10 depicts the interaction between WiFinger and Matlab®. For each host WiFinger writes a simple tab and newline delimited text file. Measurements exported for use in MatLab®are not preprocessed by WiFinger. During the experimentation process this provided more flexibility in finding the best parameters for classification. Like in (Payer et al. 2005) the used scripts produce a SOM-tree which is then automatically loaded by WiFinger.

5.4 Usage of libSom for SOM classification

libSom (Payer et al. 2005) provides loading of SOM-trees, datatypes for SOMs and vectors as well as classification functionality. Classification works as described above in section 4. For each host a SOM-Vector is used to save a histogram of ACK delays. The file *somconfig* generated by the scripts was extended to save the parameters used in the training of the SOM-tree. These are:

[1]http://www.tcpdump.org/

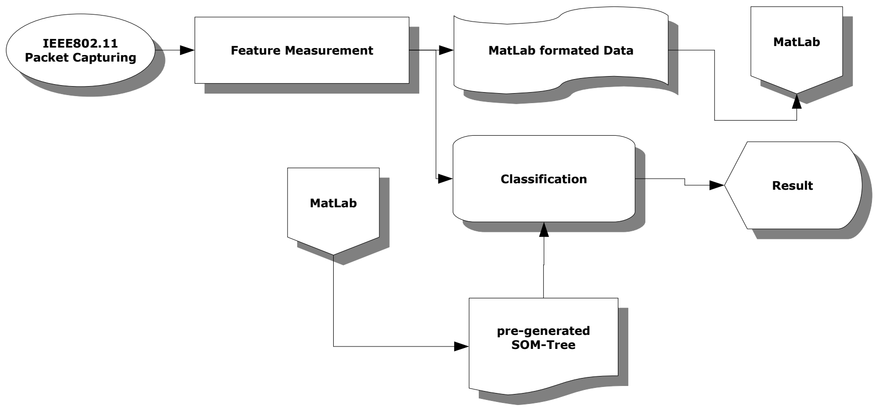

Figure 10: WiFinger Dataflow

- minimum
- maximum
- number of subdivisions

for each of the two features, acknowledge delay and data frame size.

Captured values above the given limits (500ms) are discarded. If the number of values between minimum and maximum differ from the number of subdivision, values are scaled to fit the chosen resolution. After a variable number of measured ACK delays, a copy of the SOM-Vector is normalized and the resulting *Significant Histogram* is classified. Optionally, a number of measurements can be defined after which the histogram is reset.

6 Results

For performance evaluation, the packet size and ACK delay time was taken from real traffic data with WiFinger. To facilitate this, an Aironet 350 access point was set up with Internet access. Traffic was measured using a 802.11b wireless NIC with Orinoco chipset, capable of capturing all layer 2 data in *rfmon* mode. In turn, six different wireless NICs with known chipsets were used to generate the traffic. The access point shows up in the classification results as chipset 3.

60% of this data was used for the SOM training process. The remaining test data was used to create simulated sessions with 500 respectively 1000 packets. This session based classification should give a hint on how much data from a chipset is needed to get an accurate classification result. Table 1 gives details on the evaluated chipsets and the training respectively test sets.

Chipset	**training/test data** in packets	**test data** sessions 500/1000 pkts
1: Atheros	35105/23404	47/24
2: Agere	53006/35337	71/36
3: Aironet	292858/195239	391/196
6: RaLink	19265/12843	26/13
7: BCM4306	149864/99910	200/100
8: Intel2100	41390/27594	56/28
9: PRISM	16076/10714	22/11

Table 1: training/test data sets

For performance evaluation several parameters were evaluated:

- **Number of used features**: By using the packet size information in addition to the ACK delay time information, the feature space can easily exceed reasonable boundaries. Thus, the ACK delay and packet size information needs to be grouped. This is done by specifying the number of delay partitions and the number of size partitions. A short example explains this grouping: If the packet size from 1 byte to 1600 bytes and the delay time from 70 ms to 500 ms are taken into consideration, the size of the feature space would be $1600 \times 430 = 688000$. By using 40 partitions for the delay time and 40 partitions for the packet size, this size can be lowered to $40 \times 40 = 1600$ features. The grouping is done by mapping 1 byte to 40 bytes to the first feature, 41 bytes to 80 bytes to the second feature and so on. The same procedure is applied to the ACK delay information.
- **Time/packet size range**: These parameters set the range of ACK time delay and packet size which is used for feature generation. For the evaluation of WiFinger we used a range of 70 ms to 500 ms for the delay information and 1 byte to 1600 bytes for the size information.
- **Histogram size:** This parameter is used to set the number of packets which are used to create a histogram. For our tests we evaluated a setting of 50 packets per histogram.
- **Session size:** This parameter is used to create sessions from the test data sets. The evaluation of different sessions sizes gives information about how many packets need to be analyzed before an accurate classification can be made.
- **Training factor:** This factor is used to separate the whole data set into training and test data. We used a setting of 0.6 for all tests.

The classification results were obtained in this way:

- Data was collected with WiFinger for seven different chipsets.
- The parameters described above were tuned to evaluate the impact of delay and packet size features.
- The overall classification results were obtained by getting the number of correctly classified time/size histograms for the whole test dataset.

- The dataset was split into sessions with 500 and 1000 packets. This should give an indication on how many packets are needed in order to get an accurate classification result.

- The combination of delay and packet size features which gave the best overall results was evaluated with sessions 500 and 1000 packets.

The first row of Table 2 shows the results when using delay information only. Tests from row 2 - 5 evaluate the performance of different feature sets and indicate that adding packet size information can significantly increase the classification performance. In case of 40 time slots and 40 packet size slots 80% of all histograms were classified correctly, which is an increase in classification accuracy of 64% over the first version, which only classifies 51,4% of all histograms correctly. These parameters result in a feature vector with 1600 entries, which is quite large. However, row 2 shows, that the classification performance only drops slightly when using just 10 features for packet size. The feature vector size is also reduced to 25% (400 instead of 1600), which is even lower than the feature vector size used in row 1, where only delay information is taken into consideration.

It is necessary to be careful with the number of features used for the packet size. As real data is used for SOM training, it is not guaranteed that this data has an equal distribution of packet sizes over all chipsets. Thus, by using a too fine resolution for the packet size (meaning a large feature space), the algorithm learns to classify the chipsets according to the packet size.
The parameters which gave the best classification accuracy (row 5) were used to rerun the experiment with 500/1000 packets per session. These sessions were created by using data from the test sets. The results of this evaluation can be seen in Table 3 and 4.
The following conclusions can be drawn from the results:

- There is only a very small performance increase if 1000 packets instead of 500 are used per session. The session size needed for an accurate classification result depends on the type of analyzed data. Generally, increasing the session size increases classification accuracy as noise is reduced.

- Most of the classification errors are made, when chipsets are classified as chipset 7 (PRISM 3). It seems that this chipset is quite similar to the other ones tested. Furthermore, the training set for chipset 7 was rather small. As the classification is based on the hits on the SOM, noise plays a larger role when smaller training sets are used.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**89,4**	0	2,1	8,5	0,0	0,0	0,0
2	9,9	**16,9**	0,0	1,4	5,6	28,2	38,0
3	0,0	0,0	**93.6**	0,0	6,4	0,0	0,0
4	0,0	0,0	0,0	**34,6**	0,0	0,0	65,4
5	0,0	0,0	0,0	0,0	**100,0**	0,0	0,0
6	0,0	3,6	0,0	0,0	0,0	**82,1**	14,3
7	0,0	0,0	0,0	0,0	0,0	0	**100,0**

Table 3: Confusion matrix for 500 packet sessions. E.g. 89,4 % of chipset 1 (in row 1) sessions are classified correctly as chipset 1, 0 % as chipset 2, 2.1 % as chipset 3, etc.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**95,8**	0	4,1	0,0	0,0	0,0	0,0
2	11,1	**13,9**	0,0	0,0	5,6	22,2	47,2
3	0,0	0,0	**93.4**	0,0	6,6	0,0	0,0
4	0,0	0,0	0,0	**30,8**	0,0	0,0	69,2
5	0,0	0,0	0,0	0,0	**100,0**	0,0	0,0
6	0,0	0,0	0,0	0,0	0,0	**85,7**	14,3
7	0,0	0,0	0,0	0,0	0,0	0	**100,0**

Table 4: Confusion matrix for 1000 packet sessions. E.g. 95.8 % of chipset 1 (in row 1) sessions are classified correctly as chipset 1, 0 % as chipset 2, 4.1 % as chipset 3, etc.

7 Conclusion and Future Work

This article describes the possibility of chipset fingerprinting based on timing characteristics in wireless networks in IEEE 802.11 standards compliance. The presented approach is absolutely passive and thus not detectable by the fingerprintee or any other party. All measurements are carried out by off-the-shelve low-cost hardware. The approach is resistant against any kind of standardized cryptographic routines like WEP and WPA due to the fact that management frames are not encrypted by these standards. As a proof of concept we implemented a tool called WiFinger to validate our approach in real world scenarios and obtained promising results in identifying single devices and chipsets. This paper further on describes the basics and application of the applied classification method based on *Self Organizing Maps* (SOM).

Some improvements are still possible. Beside the optimization of the feature selection for the ACK delay classification, we intend to add an implementation of the fingerprinting approach proposed by Jason Franklin et al. (McCoy et al. 2006) (see section 2 for further details) to our software. We plan to integrate the SOM training algorithm into the WiFinger tool which would eliminate the time consuming process of exporting/importing the data from/into Matlab®. This integration allows for a better evaluation by using larger training/test sets with a larger number of different chipsets and will help to explain why several of the chipsets cannot be classified with high accuracy. The next step will then be to use openWRT based IEEE 802.11 compatible access points for data collection and a centralized analysis and classification server for network wide WLAN MAC spoofing detection. This approach would allow the usage of existing WLAN infrastructure to apply our method.

We further on need to validate if it is possible to classify different devices with the same chipset. Hardware properties like *clock skews* could render a timing offset that affects the Significant Histograms and al-

feature range	time slots	size slots	vector size	results
70-500 ms (time only)	430	1	430	51,4%
70-500 ms, 1-1600 bytes	**40**	**10**	**400**	**74,4%**
70-500 ms, 1-1600 bytes	10	40	400	69,8%
70-500 ms, 1-1600 bytes	20	20	400	68,7%
70-500 ms, 1-1600 bytes	**40**	**40**	**1600**	**80,0%**

Table 2: Comparing the impact of different features. The results show that the delay information is more important than the packet size. However, adding packet size information increases the classification accuracy.

low their classification.

References

Airtight-Networks (2010), 'Wpa2 hole196 vulnerability', *Blackhat DEFCON 18* .
URL: *http://www.airtightnetworks.com/WPA2-Hole196*

Brenner, P. (1997), A technical tutorial on the ieee 802.11 protocol, Technical report, BreezeCOM Wireless Communications.

Desmond, L. C. C., Yuan, C. C., Pheng, T. C. & Lee, R. S. (2008), Identifying unique devices through wireless fingerprinting, *in* 'WiSec '08: Proceedings of the first ACM conference on Wireless network security', ACM, New York, NY, USA, pp. 46–55.

Fluhrer, S., Mantin, I. & Shamir, A. (2001), Weaknesses in the key scheduling algorithm of rc4, *in* 'RC4, Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography', pp. 1–24.

Hall, J., Barbeau, M. & Kranakis, E. (2006), Radio frequency fingerprinting for intrusion detection in wireless networks, *in* 'IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING'.

IEE (1999), *IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology Telecommunications and Information Exchange between Systems Local and Metropolitan Area Network Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*

Kerry, S. J. (2007), *IEEE Std 802.11-2007 Edition, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE 802.11 Working Group, Secretary, IEEE-SA Standards Board 445 Hoes Lane Piscataway, NJ 08854 USA.

Kohno, T., Broido, A. & Claffy, K. C. (2005), 'Remote physical device fingerprinting', pp. 211–225.

Kohonen, T. (2001), Self-organizing maps, *in* 'volume 30 of Springer Series in Information Sciences', Springer-Verlag, Berlin.

MATLAB - The Language of Technical Computing: Mathworks, http://www.mathworks.com (n.d.).

McCoy, D., Randwyk, J. V., Tabriz, P., Sicker, D., Neagoe, V. & Franklin, J. (2006), Passive data link layer 802.11 wireless device driver fingerprinting, *in* 'Proceedings of the 15th conference on USENIX Security Symposium - Volume 15'.

Payer, U., Lamberger, M. & Teufl, P. (2005), Traffic classification using self-organizing maps,, *in* 'INC 2005 5. International Networking Conference Workshops, Samos Island, Greece'.

Plummer, D. C. (1982), 'Rfc 862 - an ethernet address resolution protocol or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware'.
URL: *http://tools.ietf.org/html/rfc826*

Sieka, B. (2006), Active fingerprinting of 802.11 devices by timing analysis, *in* 'Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE', Vol. 1, pp. 15 – 19.

Vesanto, Himberg, Alhoniemi & Parhankangas (n.d.), Som toolbox for matlab, technical report a57, Technical report, Helsinki University of Technology.