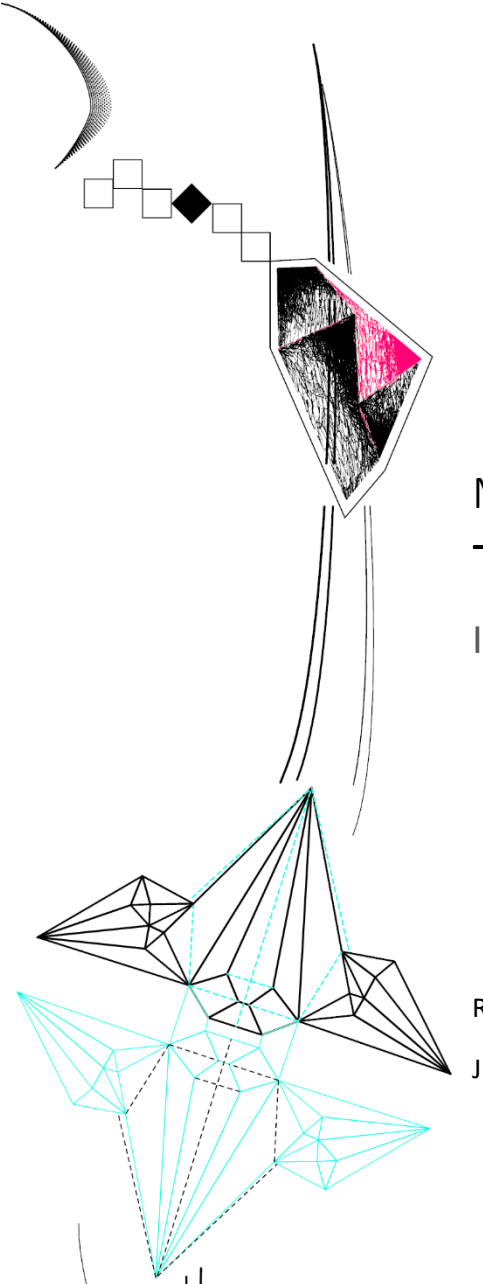


Master's thesis

Tracking of wireless devices:

Is it possible and solvable?

Remie Löwik

July, 2018

Supervisors:

Dr.ir. P.T. de Boer

Prof.dr.ir. G.J. Heijenk

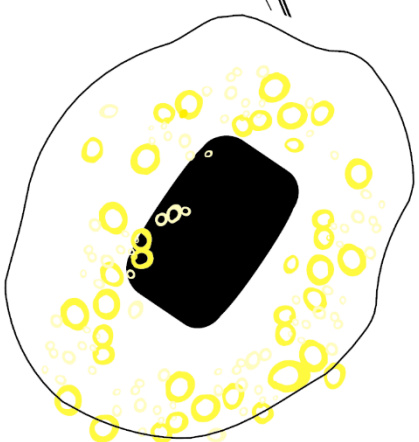
Dr. M. Baratchi

Computer Science

Faculty of Electrical Engineering,

Mathematics and Computer Science

UNIVERSITY OF TWENTE.



Abstract

A shared research experiment is performed to prove that tracking of Wi-Fi enabled clients is possible. Then individually a solution for the problem is developed, verified and tested. In this case the goal was to alter the protocol in such a way that tracking was not possible any more but to keep interoperability with older devices.

In the shared experiment students are asked to track their household occupancy and devices that collect Wi-Fi data are placed in student households for a week. Using the collected Wi-Fi data an occupancy schedule is generated and compared to the actual schedule created by the student. Unfortunately the schedule could not be generated because the students were all on the same network (Eduroam) which made it impossible to separate households and their devices from their neighbours.

The individual part started by determining which parts of the protocol allow tracking of the clients. The protocol is then implemented in Proverif to verify that the data is actually leaked by the protocol. The protocol was then altered in such a way that it would not leak any of the data whilst staying as compatible with older devices as possible. Lastly the solution is implemented on devices to verify the compatibility of the new protocol with the old one. Determining which parts of the protocol were the problem, proving this in Proverif and creating a solution in Proverif were all successful. The last part could unfortunately not be successfully implemented. Thus, the interoperability could not be verified but due to the limited changes made to the structures defined by the protocol it is expected to not be a problem.

Table of contents

Figures 1
Tables 2
Abbreviations 3
1 Introduction 4
2 Shared research 5
2.1 Introduction 5
2.2 Background 6
2.3 Method 8
2.4 Results 14
2.5 Problems and solutions 28
2.6 Conclusion 31
2.7 Discussion 32
3 Related solution research 34
3.1 Introduction 34
3.2 Awareness 34
3.3 Passive probe 34
3.4 Active probe 34
3.5 Passive mac 35
3.6 Active mac 36
3.7 Comparison 36
3.8 Conclusion 37
4 Goal 38
5 Approach 39
6 Identifying the problem 40
6.1 Frame types 40
6.2 Generic MAC frame 41
6.3 Sequences 45
7 Solution 56
7.1 Consideration 56
7.2 Protocol changes 60
8 Proving solution 66
8.1 What is Proverif 66
8.2 Implementation choices 67
8.3 Prove trackability 67
8.4 Altering implementation 68

9 Implementing solution 77
9.1 Place for changes 77
9.2 Implementation setup 78
9.3 Implementation 79
9.4 Problems 80
10 Limitations 83
10.1 Statistical analysis 83
10.2 Analogue information 83
11 Conclusion 84
12 Discussion 86
12.1 ACK control 86
12.2 Usage of PSK key 86
References 87
Appendix 88
Appendix 1: Small scale research 88
Appendix 2: Probe Proverif implementation 94
Appendix 3: Authentication & association Proverif encryption 96
Appendix 4: WPA key exchange Proverif implementation 98
Appendix 5: Multicast data transmission Proverif implementation 101
Appendix 6: Beacon Proverif implementation 103
Appendix 7: Power management Proverif implementation 104

Figures

Figure 1: Confidence level vs sample size for the university campus household list 7
Figure 2: Measurement equipment 9
Figure 3: Example of a timesheet day before and after initial processing 11
Figure 4: Signal strength distribution of the measured devices in 1 household 17
Figure 5: Detected presence of two visually matched devices against the user's schedule 19
Figure 6: Dataset 1, comparison between network traces and user's schedule 21
Figure 7: Dataset 1, comparison between the user's schedule and measured absences 21
Figure 8: Dataset 2, comparison between network traces and the user's schedule 22
Figure 9: Dataset 3, comparison between network traces and the user's schedule 23
Figure 10: Dataset 3, comparison between the user's schedule and measured absences 23
Figure 11: Dataset 4, comparison between network traces and the user's schedule 24
Figure 12: Dataset 5, comparison between network traces and the user's schedule 24
Figure 13: Dataset 6, comparison between network traces and the user's schedule 25
Figure 14: Dataset 7, comparison between network traces and the user's schedule 25
Figure 15: Average false and correct vacancy prediction rate versus device count 26
Figure 16: Average false and correct vacancy prediction rate versus device count #1 27
Figure 17: Average false and correct vacancy prediction rate versus device count #2 27
Figure 18: Average false and correct vacancy prediction rate versus device count #3 27
Figure 19: Example timesheet 29
Figure 20: privacy preserving discovery (Lindqvist et al. 2009, figure 1) 35
Figure 21: Probability of tracking devices (Vanhoef et al. 2016, figure 6) 35
Figure 22: SlyFi Protocol (Greenstein et al. 2008, figure 1) 36
Figure 23: Generic MAC frame header 41
Figure 24: Frame control field 41
Figure 25: Ack only sequence of frames 45
Figure 26: Acknowledgement frame 45
Figure 27: Clear to self sequence of frames 45
Figure 28: Clear to Send frame 45
Figure 29: Request to send sequence of frames 46
Figure 30: Request to Send frame 46
Figure 31: Beacon sequence of frames 46
Figure 32: Beacon frame 47
Figure 33: Beacon frame body 47
Figure 34: Probe sequence of frames 47
Figure 35: Probe request frame 47
Figure 36: Probe request frame body 47
Figure 37: Probe response frame 48
Figure 38: Probe response frame body 48
Figure 39: Open authentication sequence of frames 48
Figure 40: Shared key authentication sequence of frames 48
Figure 41: Authentication frame 49
Figure 42: Authentication frame body 49
Figure 43: Association sequence of frames 49
Figure 44: Association request frame 50
Figure 45: Association request frame body 50
Figure 46: Re-association request frame 50
Figure 47: Re-association request frame body 50

Figure 48: Association response frame 51
Figure 49: Association response frame body 51
Figure 50: Disassociation frame 51
Figure 51: Disassociation frame body 51
Figure 52: EAP key exchange sequence of frames 52
Figure 53: EAP frame 53
Figure 54: EAP frame header 53
Figure 55: EAP-Key frame 53
Figure 56: EAP-Key frame body 53
Figure 57: Key information field 54
Figure 58: Data transmission sequence of frames 54
Figure 59: CCMP frame format 55
Figure 60: CCMP header 55
Figure 61: PS-Poll sequence of frames 55
Figure 62: Power-save Poll frame 55
Figure 63: Comparison between key sizes (Ajay Kumar et al. 2013, table 4) 56
Figure 64: Performance comparison between RSA en ECDH (Levi and Savas 2003, figure A) 57
Figure 65: Used data structures 60
Figure 66: Adding MACs to the encrypted MAC list 60
Figure 67: Updating encrypted MAC list 61
Figure 68: Location of changes 62
Figure 69: Beacon transmission handling 63
Figure 70: Beacon receive handling 63
Figure 71: Probe and authentication transmission handling 64
Figure 72:Probe and authentication receive handling 64
Figure 73: PS-Poll/Other transmission handling 65
Figure 74: Data/Other packet receive handling 65
Figure 75: Overview of execution path 77
Figure 76: Overview of the setup 78
Figure 77: Beacon data structure 79
Figure 78: Connection data structure 79
Figure 79: Kernel Wi-Fi stack 81
Figure 80: Wireshark trace of authentication 81

Tables

Table 1: User presence results with their respective standard deviations 19
Table 2: Overview of comparison 37
Table 3: Overview with new solution 38
Table 4: Type colour coding 40

Abbreviations

ACK	Acknowledgement
AES	Advanced encryption standard
AID	Association id
ATIM	Announcement traffic indication map windows
BSSID	Basic service set identifier
CBC	Cipher block chaining
CCMP	Counter mode cipher block chaining message authentication code protocol
CFB	Cipher feedback
CTR	Counter mode
CTS	Clear to send
CTS	Clear to send
EAPOW	Extensible authentication protocol over wireless
ECB	Electronic codebook
ECDH	Elliptic curve Diffie-Hellman
FCS	Frame check sequence
KCK	Key confirmation key
KEK	Key encryption key
MAC	Media access control
MIC	Message integrity code
MiTM	Man in the middle
NAV	Network allocation vector
OFB	Output feedback
PSK	Pre shared key
PTK	Pairwise transient key
RC4	Rivest cipher 4
RTS	Request to send
SSID	Service set identifier
TIM	Traffic indication map
TK	Temporal key
TKIP	Temporal key integrity protocol
WDS	Wireless distribution systems
WEP	Wired equivalent privacy

1 Introduction

In a world where the digital world becomes ever more important, the devices we use to access that world also changes. In 2007 less than a third of the users were mobile users ("Mobile marketing statistics 2018," 2018), but after 2014 this already grew to more than half of the users and still continued to grow afterwards. What all these mobile devices have in common is the methods of how they communicate, the most common methods are mobile connections like 3G and 4G and Wi-Fi. Although users are more privacy-aware nowadays, little is known by those common users about how much information is leaked by especially the latter of the communication methods.

Very low awareness in combination with high usage and very interesting information makes it a very good target for less friendly usages. In the case of Wi-Fi, which is widely used in households, this could be a very interesting area for example for burglars. As burglars should be very interested in knowing when homes are empty as those moments are opportunities for them. Luckily there are no known incidents of burglars using these possibilities to their advantage, but we think that with little effort presence of people could be detected with very cheap hardware. Big companies are already using these kinds of techniques to track costumers in department stores and on large festival areas (Verbree et al., 2013). Therefore we (me and two other students) started this research, first we did this for a course given on the University of Twente. This successfully proved that it could be done with a high certainty. Though this was done using family members and friends as a target group. Because of the results we were asked to extend this research further with a larger and more random group of people. We then proposed to make this our shared research topic. To extend the research a little further, a proposition was made that we would jointly research the problem and separately research a solution. In the end a division was made to use one third of our time to research the problem and two-thirds in researching a solution. The solution proposed by me will be to change the 802.11 protocol in such a way that no information is leaked about the user any more whilst keeping the compatibility with other devices, more about this is discussed in chapter 4. But first the shared research is discussed in chapter 2.

2 Shared research

2.1 Introduction

This chapter covers the research into trackability of household occupancy using the Wi-Fi network. This research is a follow-up of an earlier small-scale research (see appendix 1) performed by the same researchers among the households of relatives. The usability of that research was very limited due to the scale and potential bias. This research tries to prove the potential of Wi-Fi eavesdropping to track occupancy in households.

The execution of this research is a joint effort between Remie Löwik, Ruben Lubben and Tim Kers. These researchers performed their own research into potential solutions against Wi-Fi tracking. This chapter, assessing the potential risk of eavesdropping on Wi-Fi networks is a joint effort between Remie and Tim and will be identical between their respective theses.

The research is divided into 2 parts. Due to practical reasons, the measurements are conducted in the living quarters on the campus of the University of Twente. These living quarters feature a shared Wi-Fi network called Eduroam. Instead of separating the devices per household by their used network, as would be possible in normal households, this shared network throws all devices on one pile. Or at least from the burglar's perspective.

The first research step, would be to use other parameters to determine the critical devices for the participating household. After this step, the situation is again similar to normal households where only relevant devices are registered. At this point, the trackability of the network can be determined.

This chapter therefore knows two research questions:

- Is it possible to determine which Wi-Fi devices belong to a certain household in a shared network with only passively detectable parameters?
- Is it possible to reliably track occupancy in a household with passive eavesdropping on its Wi-Fi traffic?

2.2 Background

As stated in the introduction, this research was preceded by a small-scale experiment in 2016. In this small-scale research, borrowed laptops were used as measurement devices which limited the group of participants to relatives and friends. Unfortunately, the stability of the borrowed hardware and the many configurations onto which the software had to work proved to be a problem. Combining this with a very limited timeframe, limited the experiment to 12 households. This in turn limited the statistical relevance of the research.

The results, however, did indicate a potential problem with household Wi-Fi networks. On average, 86.7% of predictions were correct. The 13.3% faulty predictions were made up of false occupied (10.5%) and false vacant predictions (2.8%). For a burglar, false occupied predictions are potentially missed opportunities. However, as long as other opportunities are available, this is not really a problem. The false vacant predictions are problematic for a burglar. These are the times they would think the house was vacant while it was not and would risk getting caught.

Most of these false vacant predictions occurred at night, partly due to households having limited Wi-Fi coverage in the bedrooms causing residents to turn their Wi-Fi off at night. When the 00:00 to 07:00 timeslot was removed from the analysis, correct ratings increased to 89.3%, false occupied declined to 10% and false vacant diminished to 0.7%.

Although less relevant to this research, a small social study was conducted as well. It showed that participants felt slightly less safe in their neighbourhood, with safety grade lowering from 7.5 before and 7.33 after the research, on a scale of 10. More people had the feeling of being unsafe in their homes (50% before to 58.33% after) and the likeliness of a burglary happening to them in the next 12 months was graded 1.6% higher than the 25% before the research.

The social part of the previous research was not included in the new research. This was mainly due to the amount of time and effort it involved to get all participants to fill in the forms. The forms also required more work from participants, which was deemed as a potential deal breaker for them. Additionally, this research focuses on the technical side of this potential problem. The social study is not regarded as relevant for this part.

Unlike previous research, this one was intended to prove the potential of eavesdropping on household networks in a statistical relevant matter. This required larger datasets and a non-biased group of participants. The latter is tackled by randomly choosing households out of a list of living quarters on the campus of the University of Twente. This is further explained in paragraph 2.3.1.1. This yielded a list of 556 potential participating households. We estimate that a quarter of the potential participants will be willing to participate. To retain a level of randomness in the selection of the participants, we will use a maximum of 50% of this list. This leaves an upper limit of around 70 participants.

In statistical experiments the required number of samples can be determined by (Lisa Sullivan, PhD, n.d.):

$$n = (\frac{Z\sigma}{E})^2$$

Where, Z is dependent on the confidence level. In this case, 95% yields a Z of 1.96. σ Indicates the standard deviation, which is fairly unknown at this point and therefore set to 50%. E is the margin of error which is plotted against the sample size (n) in Figure 1 below

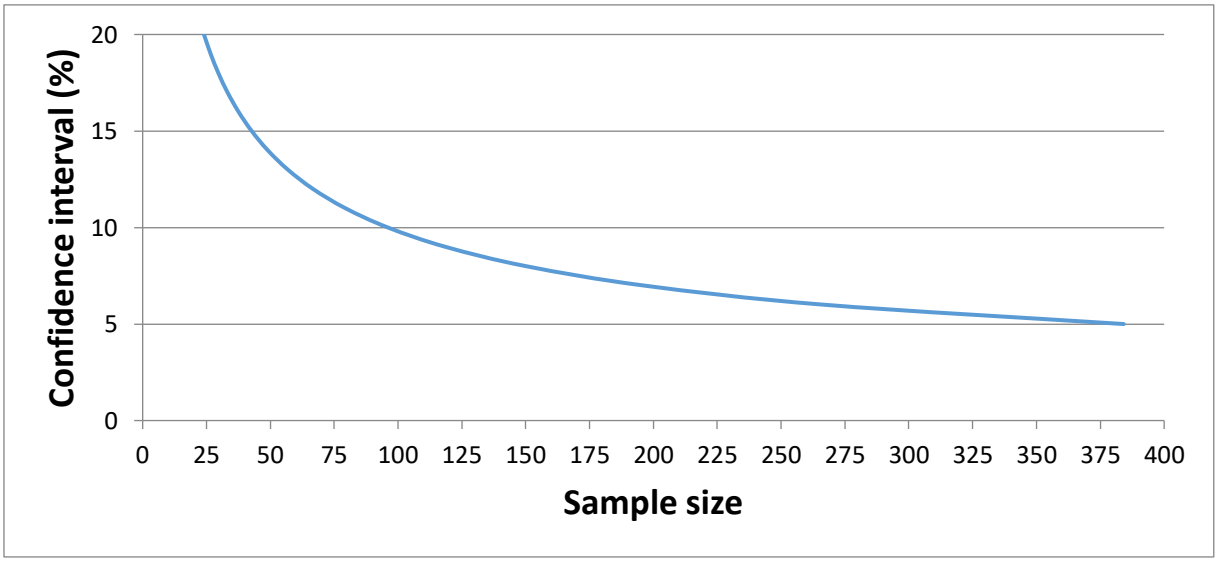


Figure 1: Confidence level vs sample size for the university campus household list

To reach sub-10% intervals, sample sizes of 100 and higher are required, which is not feasible with our pool of participants. Therefore, a compromise was made to aim for a 15% or better confidence interval and the accompanying requirement of 43 or more datasets. This was deemed feasible with the available time and equipment and keeping in mind some problems on the way.

2.3 Method

This experiment is split into three parts. First, measurement equipment is placed in the homes of participants to gather network traces to be used in the later parts. The residents receive a form on which they are asked to keep their presence to be compared with the retrieved data afterwards. After retrieval, the filled-in timesheet and trace data are pre-processed to prepare for the next parts.

In the second part, the pre-processed data is processed to remove any device not belonging to that household.

The third part would then aim at extracting an occupancy schedule from the network trace and compare this to the schedule filled in by the participant.

2.3.1 Part 1: gathering network traces from households

In the original experiment, datasets of multiple weeks were recorded to try and recognize recurring patterns in people's lives. In this research, the datasets are chosen to be only one week long, to try and reach a higher number of datasets in the available time frame for this research. The focus therefore lies on reliable occupancy detection instead of pattern recognition. When occupancy detection can be performed reliably, pattern detection should not be a problem.

As the research potentially involved privacy sensitive data of the occupants, the research proposal was reviewed by the Ethical board of the EEMCS faculty at the University of Twente. This gave some restrictions on target groups and data storage that will be explained further down in this chapter.

2.3.1.1 Target group

A problem with the earlier experiment was the use of relatives as test subjects, this gave potentially biased data and therefore should be avoided in the new experiment. For this new experiment, subjects should be chosen at random from a large pool of potential candidates.

The ethical board gave an important restriction on the potential candidates. All occupants in a participating household must be able to understand and consent to the potential privacy risk. This prohibits measuring households with for example underage children or mentally challenged people.

Eventually the aim was set on student housing. This gives an easily containable set of candidates, almost no underage people and very small chances of children living in and/or visiting the household. This left two possible groups: Dormitories and individual living quarters. Dormitories posed a couple of potential problems.

- When measuring a complete dormitory, all students living there must consent. With living groups up to 16 people, it is not unlikely that at least one would refuse.
- Standard measurement equipment would probably lack the range to cover the complete dormitory, thus requiring more equipment and opening the door for synchronisation issues and/or potential blind spots.

Alternatively, measurements could focus on individual occupants in a dormitory. A measurement device could then be placed in the room of the participating student. However, this gives a similar range problem. When a student leaves his room to eat in the shared living room, he or she is likely to be out of range. This system would consider this as "absent". The student should therefore note his presence in the actual room, which quickly becomes a hassle and error prone.

Ultimately, the choice fell on individual living quarters on the university campus. The housing agency provided us with a list of 556 individual housing quarters found across campus. These are divided in full apartments, studios and standard sized rooms with personal facilities. These areas are all coverable with standard Wi-Fi products and are usually occupied by one or two people.

2.3.1.2 Privacy considerations

As this research involves privacy sensitive information about people and their household, some precautions had to be taken:

- No user data is stored by the measurement device at all
- The device identifier (the MAC address) is only stored as a hash to stop anyone from finding easily finding the original device. Although scanning the whole campus could still be easily done, preventing such action is fairly hard whilst keeping usable data. Additionally, anyone with such interest would be better suited with gathering newer data instead of trying to crack the old.
- The retrieved timetable is linked to the measurement device its device number. However, this number is never linked to a house address, phone number or email address. This means that there is no way to link a dataset or timetable back to a household or individual.
- After retrieving the measurement device, all data is removed from the SD-card before reusing it for another household. Although the stored data would be barely usable for any adversary, this prevents other people from retrieving the data.
- All research data is to be permanently removed no later than 1 year after completing the research, as stated in the original research proposal (see appendix 1). The data is only accessible to the researchers and supervisors stated in the proposal and brochure.

2.3.1.3 Measurement equipment

For these measurements a device was required to capture network traffic. As student housing is covered with the Eduroam Wi-Fi network, monitoring this network is sufficient in most cases. The network is divided over the three Wi-Fi super channels (channel 1, 6 and 11) thus requiring 3 network interfaces. The choice fell on the Orange pi lite minicomputer. This creditcard sized computer features an onboard Wi-Fi module (XR819) and two additional USB ports for two additional USB Wi-Fi card (Ralink RT5370).

An important parameter was the support for monitoring mode on the Wi-Fi interface. This was a problem with selecting a Raspberry pi. Its on-board module does not support monitoring mode requiring us to add 3 external Wi-Fi modules. Furthermore, the cost of a raspberry pi is almost double that of the Orange pi Lite.

For the OS (Ubuntu) and measurement data, a 16GB micro SD card is used. With data compression used in our system, this would easily cover measurement data for multiple weeks.

Figure 2: Measurement equipment

2.3.1.4 Data gathering

When the device is started, it places all three Wi-Fi modules in monitoring mode. In this mode, the module will listen to all traffic on that frequency regardless of destination or network. In this case the modules will be set up to listen all three super channels. By using monitor mode, the module does not have to be associated with any network to listen to the data that is transferred on that channel.

The traffic is monitored for each interface separately by creating a TCPdump instance for each of the interfaces. TCPDump was configured to return only the data we required, in this case the following information was stored to file for each packet:

- Source device
- Destination device
- Timestamp
- Signal strength
- Packet type

The output of TCPDump was then parsed by a java program and then processed further for storage.

Due to privacy concerns, instead of storing the MAC addresses of the source and destination device, an anonymized hash is created and stored. Furthermore, user data in the packet is not stored. It would not be relevant for the research and take a lot of storage space, but it is also a privacy concern.

Each interface writes its data to a set files. Then after an hour, a new set of files is started and the old files are flushed and closed to make sure that all packets are committed to storage. This technique also helps in preventing data loss. If a device loses power suddenly, depending on the current activity of the system, data could be lost. By storing the data in chunks, this data loss is limited to a maximum of 1 hour.

Furthermore, the choice was made to split up the information into three different files: data-, mac- and extra packets file. The first file is the data file. In this file the mac addresses, a timestamp, signal strength and packet type is saved for each packet that is received on the interface and is directly compressed with the GZIP compression algorithm to minimize the size of the data. Because mac addresses are the biggest portion of the data, the choice was made not to rely on the compression algorithm but instead to make a lookup table in which all the MAC addresses are given an ID. This ID is then used in the data file instead of the longer MAC address.

The second file is the content of the lookup table: an ID with its assigned MAC address. But before saving the mac addresses, the macs will first be hashed using the SHA256 hash function. In the end this lookup table did not only save storage space but also minimized the chances of errors: Hashing and storing the MAC addresses only once minimizes the change for errors. Furthermore, extra processing is saved by only having to hash each MAC address once instead of having to hash the macs for each received packet.

The last file is used to save unknown packet types, because there might be a chance that the output of the TCPDump program is not correctly interpreted. Therefor if a packet is not correctly recognized by the program it creates a new “packet type” assigns a new type id, adds some extra formatting information and saves this to this file. If this packet type is encountered again it could use the information saved with the previous packet to identify it as the same type.

2.3.1.5 Measurement procedure

From the original list of households, a random selection of 60 households at a time is chosen by a Matlab script using the standard *rand()* function with a random seed of 42. These households receive an introductory letter about the research to give them some time to consider participating. Then, after approximately a week, the houses are visited and the residents asked if they would like to participate in the research. If required, additional information can be given. If nobody is home at that time or the participant wishes some extra time to consider participating, the household is tried again at a later time. Obviously, a resident is free to decline participation without reasoning, after which the house is removed from the list.

When a resident chooses to participate, one of the measurement devices is handed over and plugged into a power socket inside the house. Additionally, the subjects get a form with a timetable on which they are asked to keep their presence log during the measurements. This timetable is used as a reference to validate the conclusions drawn from the measurement data. For extra information about the research, the privacy concerns and proper actions, should they want to stop the measurements, an informational brochure is handed over for them to keep. Finally, the participant is asked for contact information such as a phone number or email address so that, after a week of measuring, the participant can be contacted for retrieval of the device and timetable.

The introductory letter, blank timesheet and informational brochure are added in appendix I of this report.

2.3.1.6 Initial data processing

After retrieval of the measurement device and timetable, their data has to be processed before it can be used to identify occupancy.

Timesheet processing

All timesheets are scanned and digitally processed. Initially, the “marked” fields are made uniformly black to prevent reading error by the automated processor. An example of this is shown in Figure 3.

Figure 3: Example of a timesheet day before and after initial processing

After this step, the images are loaded into an automated processor, created in Matlab. This program lines up the filled in timesheet with a reference (empty) version and determines the light level of each data field (white or black, indicating unmarked or marked). For this, predetermined coordinates are used, derived from the reference timesheet.

Participants were allowed to choose if they preferred to mark for “absent” or “present” as long as they indicated their choice on the timesheet. Additionally, participants sometimes mixed up days or started marking at a different day than the first one on the form. All these factors were manually entered into the processor, which (where applicable) inverted the derived schedule or rearranged the days.

The result of each timetable is a text file with 7 lines (days) of 96 characters (quarters). For each character, a ‘0’ symbolizes vacancy and a ‘1’ occupancy.

As discussed in saving data part, the device saves three files per interface per hour. The choice was made to do some pre-processing on this data to lower the amount of data that had to be processed every time. To do this a program was written that would read and uncompress this data and summarize the presence for each device. This was done by creating blocks of 5 minutes in which packet type count, the minimum, maximum, average signal strength and to whom each client was talking to was saved. This data was then exported to a csv file to allow further processing in Matlab.

2.3.2 Part 2: Automated filtering of relevant devices

2.3.2.1 Selecting devices within the household

In a normal household environment, a burglar can select a certain network and therefore household to track. This allows him to only track devices using that network. Unfortunately, just as many universities, the University of Twente uses the Eduroam network across the entire campus including the living quarters. As a lot of students will be using this, the distinction between houses disappears. This means that other steps have to be taken to extract devices belonging to the targeted household. If this step succeeds, the remaining trace only contains legitimate devices for that household and the situation is again similar to a normal household.

Two factors were used to determine devices belonging to that household. The measurement device logged the signal-to-noise ratio of every received device throughout the week. With the device placed within the household, the devices with the highest ratings will most likely belong to that household.

As a second factor, the interaction between different devices is checked. The idea behind this is that devices within the same household may often communicate with each other. For example, a laptop checking the availability of a network printer, or a mobile phone streaming a video to a smart tv. With this second step, a device tucked away in a corner or cupboard but belonging to that household may still be recognized while its SNR values would imply it is a device from another household.

2.3.2.2 Selecting devices with usable characteristics

Nowadays, many different devices can be present in networks. A burglar will probably be best served with smartphone availability, as this device is mostly carried around with the residents. Laptops, tablets and other devices could give similar information.
But a stationary device like a network printer, being active all day long, would not be very interesting to determine occupancy. Therefore, some extra filters are added to separate usable devices from the trace.

- Discard devices with high active or inactive rates
- A device that is communicating continuously or barely does not give much insight in any resident's schedule. Therefore, any device that is active for more than 95% of the time or less than 5% of the time is discarded. The likelihood of a resident having such a schedule is almost zero.
- Session lengths
- Schedules differ between people, but some factors are fairly constant. Over the period of a week, one can expect the residents to be home for some lengths. For example, because they sleep at home. Therefore, a filter is created that looks at the occurrence of certain session lengths. For example, if a device is never present for a couple of hours, it is very unlikely that its trace will represent the residents schedule
- Session counts
- Similar to session lengths, session counts can be used as a parameter as well. A real person would not come home and leave every 10 minutes (for example), nor would they stay at home for 5 days and then disappear for the weekend. In the first situation, it is more likely that it involves a device connecting periodically. In the latter, it looks more like a stationary device, but it is turned off when the resident leaves for the weekend. Although exact boundaries for "legitimate" devices are hard to draw, the extreme situations as stated above can be removed relatively safe.

2.3.3 Part 3: Extract household occupancy from network trace data

In a normal household, the Wi-Fi network would be used by the people and devices belonging to it. This makes tracking much easier as the trace would not be influenced by neighbouring devices. In the chosen Eduroam environment, all households share the same network. But after extracting the appropriate device traces from the dataset, the situation should again be comparable to a normal household.

The next step is to generate occupancy schedules from the network trace and compare this to the schedules filled in by the participants. A burglar will aim to minimize risk. As he will need only one free moment, it is less relevant if other potential moments go unnoticed due to an overly safe technique.

The safest options to start with is to regard every captured device as relevant. Only when all devices become silent, the house is regarded empty. In addition to that, a burglar would not be interested in free windows of a couple of minutes. Instead, only continuous vacancies of 15 minutes or more are deemed relevant.

As with all of these predictions, the burglar would be looking for an absolute minimum false vacant predictions. These are the moments he could be detected. As long as not all potential moments are lost, no technique is "too safe".

2.4 Results

2.4.1 Part 1: gathering network traces from households

Gathering the network traces from the households proved to be a very time-consuming process. Apart from all the hours distributing introductory letters, asking for participation and retrieving devices, a lot of time was consumed by software issues on the measurement devices and to process the data.

2.4.1.1 Start-up phase:

Before being able to distribute any device, software had to be created for the measurement equipment. In this step, multiple test rounds were conducted to test the software for functionality and reliability. Some problems were found and resolved in this phase, like occasional failure to initialize a network interface. In these cases, one of the interfaces became unusable for the data logging software. As this problem was detectable and re-initialization of the module was sufficient, this problem was effectively resolved.

2.4.1.2 First measurement round:

After multiple rounds of short and long tests, the system was deemed ready for deployment. Unfortunately, after the first round of real-world tests, the resulting data from all 10 participating households came back corrupted. The cause of this was found to lie within the LZMA compression algorithm used to compress the recorded data.

The problem turned out to be a memory allocation issue and finding a solution within the compression software proved difficult. Fortunately, storage space turned out to be plenty for a week of data allowing a switch to the more commonly used but less efficient Gzip compression algorithm. This solution was tested in multiple networks for multiple days and proved reliable.

2.4.1.3 Final measurement rounds:

After the problem in the first round of measurement was resolved, multiple successful measurement rounds were performed before the holidays put a stop to this research step. In total, 45 households participated in these rounds before the holidays brought a stop to them.

Of these 45, 8 were lost due to administrative mistakes. 6 of them were found to be checked off, but never actually retrieved. Due to the long period between data gathering and processing, this discrepancy went unnoticed. The participants were contacted when this problem was found. The device was successfully retrieved from two residents. one admitted the device was never retrieved, but lost it while moving to a new house. The other three never responded.

Two other devices remain unaccounted for. It could be that they are also still out there with participants, but we were not able to find out whom. The strict separation between consent forms (with personal information) and devices and their data may be good for privacy concerns, but did prevent us from backtracking which consent forms were never met with data.

On top of the administrative error, one dataset became unusable as its accompanying timesheets went missing. With that, only 36 datasets remained before processing even began.

Although the major issues were resolved, some measurements still developed problems. Some of the found problems were:

- Measurement devices missing data from one of the network interfaces. This looks similar to the earlier initialization error, except that the software never found an initialization error nor were there any problems reported in the system's logs. Normally, a problem with one of the network interfaces should trigger a system reboot to try to re-initialize everything. However, this did not happen and the system continued its operation with two interfaces. This problem only occurred in one of the measurements making a not completely plugged in USB Wi-Fi modules plausible.
- Measurement devices seized to record any data during the measurement period. Although the device was placed for a minimum of 7 days, the trace would only cover a couple of hours or days in some cases. Similar to the previous problem, no evidence of it was to be found in the systems logs. A possible cause could be a loss of power. Maybe a resident moved the device causing the power jack to become loose or unplugged an extension cord while forgetting the device that was placed there. In total, 5 devices showed these kinds of problems with their active time varying between 26 and 95 hours. One of these devices had its data split with a reboot in between. As the device does not have a real time clock, there is no data on the amount of downtime between these two sessions.
- Measurement devices developing corrupted files within the data. This could have been caused by a power loss or other reboot event. This problem affected two devices, but only influenced a couple of files. The software was created to store data in one-hour blocks to prevent large data loss in such cases. Therefore, the datasets remained usable, although missing an hour somewhere.
- Devices not logging any data. In total, three devices came back without any measurement. In one of the cases, this was due to the SD card not being inserted properly. Although powered all week, the device never measured or even booted. The second device did boot up and created the initial logging files and system log entries, but the device probably stopped working soon after that. No further logging files were created (which should have happened every hour) and system logs did not show any more data. The last device had its power jack not inserted properly due to the improvised (cardboard box) case used for 10 devices.

Eventually, the holidays limited the available time for measurements as a large amount of the residents moved away for some time. In the end, after removing all faulty datasets, only 25 datasets remained to be processed further. Unfortunately, this is far less than the aimed minimum of 43, limiting the statistical relevance of the outcome of this research. The confidence interval was now limited to 19.2%, assuming no further problems arose.

2.4.2 Part 2: Automated filtering of relevant devices

Due to the choice of an area with a single large Wi-Fi network, it was expected that neighbouring devices would be picked up in the measurement. The first step would be to remove these from the trace. The resulting dataset should ideally only include all devices belonging to the participating household. This situation would be similar to a measurement in a normal household where devices are separated by their used network.

2.4.2.1 Original approach

While processing the data, the number of unique devices recorded in the measurements proved to be extremely high. As the experiment was conducted in the Eduroam environment, it was expected that large amounts of devices would be found from neighbouring households. However, it was not expected that most datasets would contain hundreds of recorded devices and some which even went up to hundreds of thousands.

One cause for this huge number of devices is people passing by the house. This would result in a registration of their device (if active on Wi-Fi) for a short amount of time. Additionally, the MAC randomization scheme of some versions of IOS and Android would create a lot of "fake" devices as long as the devices has its Wi-Fi capabilities enabled but is not connected to a network.

Multiple rounds of filtering were used to try and remove any unwanted device from the traces. Initially, 5 datasets were picked as training set to adjust the filters. These filters would then be applied to the other datasets.

Remove extremely short and long presences

People walking by or devices with MAC randomization create a lot of data that is not usable for occupancy tracking. Therefore, all devices that were picked up for a total of less than 5% of the total measurement duration, or approximately 8 hours out of the week, were removed from the trace. This includes MAC randomizing devices, people walking by and someone visiting during the week.

Additionally, devices that were present for more than 95% of the time were also removed. These devices include access points and stationary devices. These devices yield no information about the resident's presence and are therefore fairly useless for a burglar.

This filter removed a major part of "unusable" devices from the trace and reduced the datasets mostly to sizes between 25 and 75 devices.

Group devices together by mutual communication

The idea behind this filter was that devices belonging to the same household are more likely to communicate with each other. For example, video streaming from a laptop to a TV, or sending a document to a network printer.
Unfortunately, devices proved to be much more talkative than that. Intercommunication happened everywhere in the dataset making distinction between different device "groups" impossible. Therefore, this filter was not used any more.

Remove devices with low signal strength

Devices within the household are in close proximity of the measurement device and should therefore read high SNR values. Finding the exact threshold after which a device does not belong to that house is going to be difficult due to all the different circumstances in and around the households. However, it can be used to filter out “distant” devices and reduce the dataset by a significant amount.

Figure 4 shows the signal strength distributions in one of the datasets gathered in this experiment. Most devices reside in the far left of the graph, making them most likely to be distant. However, it is difficult to select proper thresholds to distinguish devices actually belonging to the household. Manually comparing the dataset to the filled-in schedule revealed 1 perfectly matching device. However, when looking at the average signal strengths, that device came second with the first device showing no relation to the schedule. When looking at peak values, the matched device fell down to 16th place.

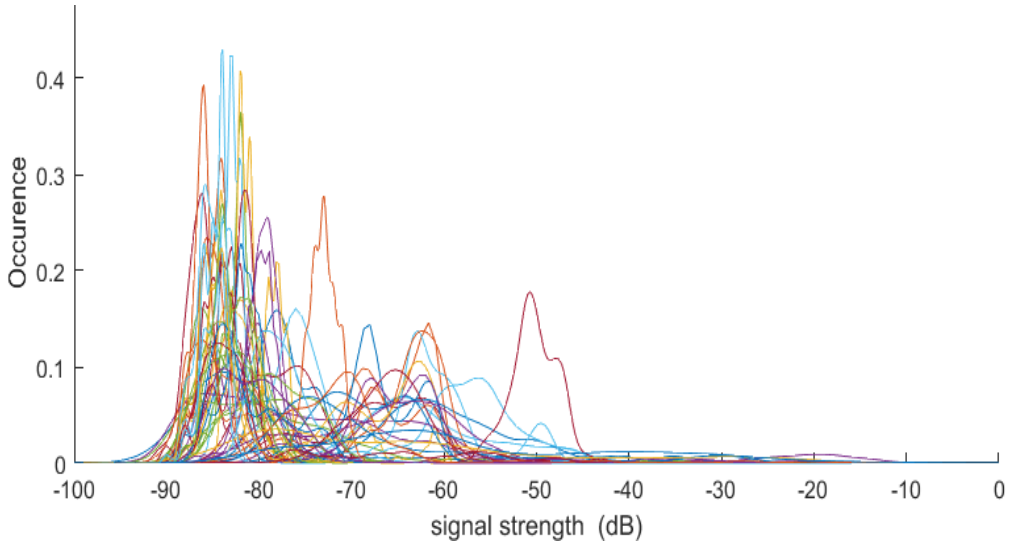


Figure 4: Signal strength distribution of the measured devices in 1 household

No similarity in the results was found across the datasets. The original training set of 4 datasets was even doubled to 8, to try and find the best matching filter settings. However, the filter was not able to remove all “unwanted” devices without losing genuine ones as well.

Another problem that arose, was the lack of “matching” devices in a lot of datasets. Although some devices showed high signal strengths, they would not be comparable to the schedule that the resident filled in. This problem is further worked out in 2.4.3: Alternative approach.

Session lengths

Analysis of the datasets showed some interesting characteristics in some devices. For example, some devices would show enormous amounts of activity, but all in short bursts.
Although it is unclear what kind of devices these actually are, but it is not likely to reflect the schedule of a resident. An actual resident would normally have periods of presence and absence. To try and filter for those characteristics, session lengths were checked. It would be likely that a resident would have multiple presences of a couple of hours during the week, for example to sleep, study or relax.

This filter proved reasonably effective. Many devices with the behaviour talked about above were filtered out. Specific filter settings proved to be only mildly influential. Any setting for a couple of presences of a couple of hours was reasonably effective. The filter was only effective in removing unusual devices, not in selecting devices for a specific household.

Session counts

This filter had a similar aim to the previous one. During a week, a resident would probably leave a number of times. But to the rapid transitioning devices mentioned earlier showed extremely high numbers. Other stationary devices that had 1 period of absence would pass through the <95% filter, but would show very low session counts.

This filter was set out to filter out unrealistic low and high session count numbers. Although reasonably effective, it did not have any influence over the session length filter. Therefore, this filter was eventually dropped.

2.4.2.2 End result

In the end, a uniformly applicable filter was not achieved. The filters, when combined, gave a reasonable decline in device count, but returned both genuine and neighbouring devices. Even within the test group, with prior knowledge of the schedules, no acceptable result was achieved.
As mentioned earlier, many datasets appeared to be lacking “genuine” devices at all, when comparing to the residents’ schedules. Of the original 4 datasets selected as a training set, only one showed clearly matching devices and one other showed similar (but not perfectly matching) devices. This raised the question if it was even possible to extract occupancy information from these datasets.

Therefore, the original filtering approach was halted, and the focus now came on verifying if there was actually usable data in the datasets before continuing.

2.4.3 Alternative approach

As mentioned before, a lot of datasets appeared to be lacking any devices matching to the schedule. This raised the question if occupancy tracking was even possible with the devices picked up by the measurement devices.

Therefore, instead of using a training set, all datasets were manually compared to the schedules to find any (seemingly) matching devices. Although time-consuming, the easiest method proved to be to plot (a subset of) the devices together with the schedule and visually match them together. Automated versions were tried, but they would occasionally miss devices or incorrectly match them. Sorting the devices by their mean signal strength proved to be effective. The matching devices would (as expected) usually occur in the top part of the selection. In the end, potentially matching devices were identified in only 14 of the remaining 25 datasets. In most households one of the identified devices would closely match a device. Any other would have a lot of resemblance, but also errors. Figure 5 shows a comparison between two visually matched devices and the accompanying schematic.

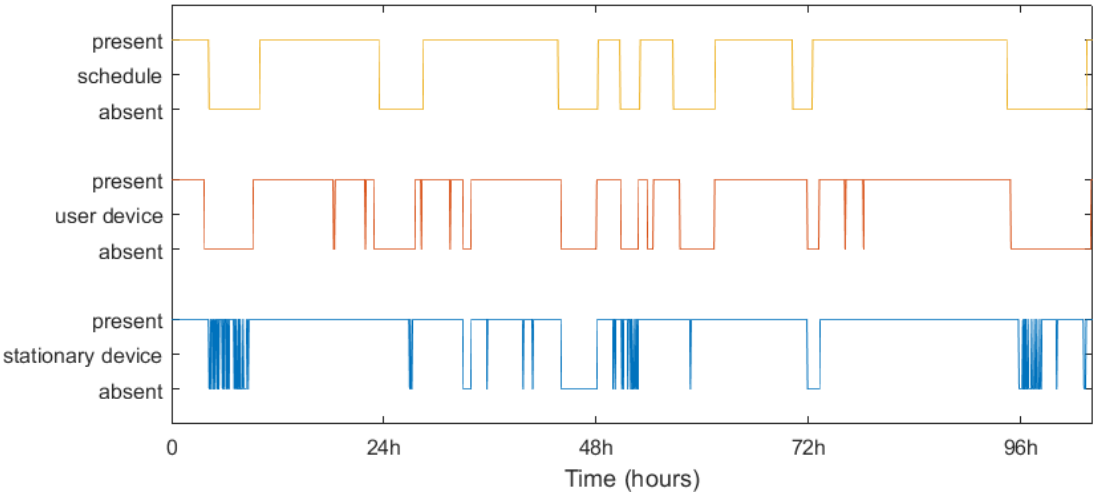


Figure 5: Detected presence of two visually matched devices against the user's schedule

Both devices behave similar to the schedule. However, the bottom device often becomes intermittent when the user is supposed to be away. This is likely to be the behaviour of a stationary device periodically checking the return of known devices. The real "user" schedule appears to be the middle graph.

To get an impression of reliability between the schedule and trace data, the visually best matching device of each household was selected and scored. These devices are likely to be smartphones and similar devices, closely representing the user's presence. These results are presented in Table 1: User presence results with their respective standard deviations

below.

Correct occupied prediction	Correct vacant predictions	Total correct predictions
90,4 % ± 8,9%	87,6% ± 11,9%	87,8% ± 9,8%

Table 1: User presence results with their respective standard deviations

This result does indicate that occupancy could be determined from Wi-Fi data, if the correct devices can be selected from the dataset. However, this result only covers 14 datasets out of 25.

2.4.4 Part 3: Extract household occupancy from network trace data

As explained in part 2, the automatic filtering of devices proved problematic. The proposed method of only selecting relevant devices with filters and extract occupancy out of that is therefore difficult.

Instead, this part is split into 2 parts. First, all visually matched devices of the household are combined and scored. These devices are the most likely to reside within the same household. This combined dataset is compared against the user's schedule to see if usable data has remained. Additionally, some of the filters of part 2 are reused. Although the filters were not able to remove all "wrong" devices, they may still be usable. If genuine devices are present in the dataset, combining them with "wrong" devices only removes potential vacant moments. But it does not add false vacant readings.

Unfortunately, this technique is only applicable to the datasets in which at least one device was recognized. As the measuring equipment lacked any means of measuring date and time, there is no way of lining up the measurements with the schedule without visual checks. A rough estimate can be made, but the manually checked datasets showed various amount of offset remaining.

2.4.4.1 Combining visually matched devices

This technique was only applicable to 7 of 14 the households with visually matched devices. In the other 7, only one device was matched to the schedule. The single device matches were already covered in part 2. The remaining datasets had two (4 times), three (twice) or five (once) devices matched to their schedules.

For each dataset, the traces of all devices are combined into one. Combining the devices effectively performed an "OR" operation on the traces. If any of the devices is present at that moment, the combined trace is too. From a burglar's point of view, this is the safest option. Only when no device is active, the house is regarded empty. The combined trace is added to the first figure presented for each dataset, this to give an overview of the used data.

Afterwards, short absences are removed from the combined trace as a burglar would not be interested in those. In the second graph, three versions of this filtered combined trace are then presented with different minimum absence settings.

2.4.4.2 Dataset 1

In the first dataset, 2 devices were recognized. Figure 6 shows their behaviour compared to the schedule. The 2 devices share a number of absences which in turn match roughly with the schedule. However, there is a slight offset between the absences in the schedule and the devices at some times. This could be down to small errors when filling in the schedule.

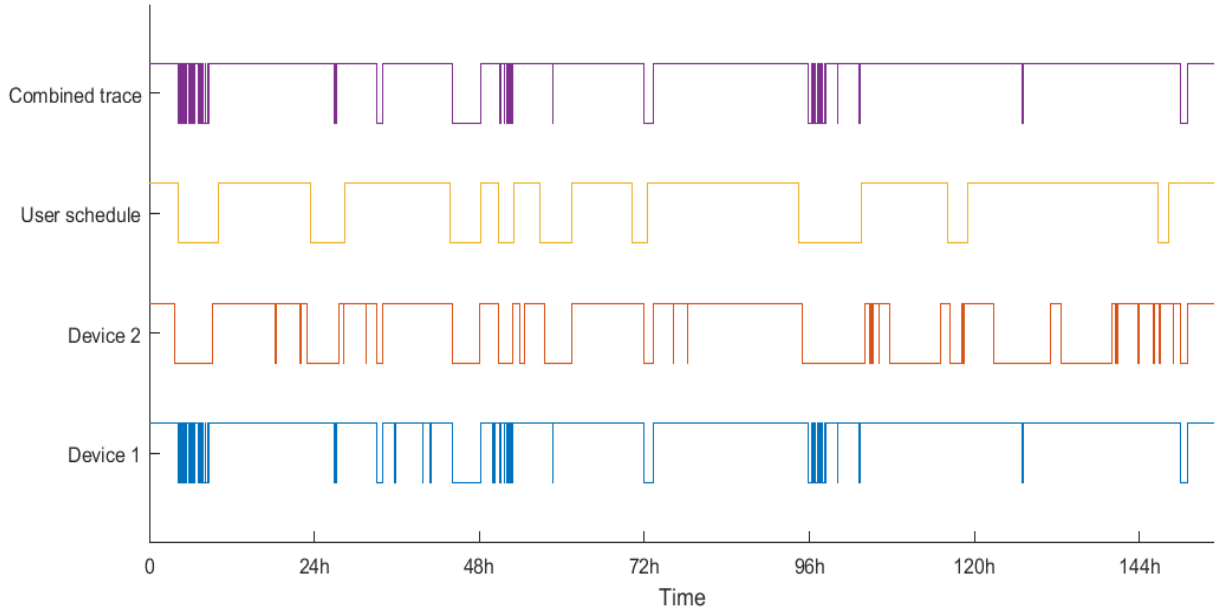


Figure 6: Dataset 1, comparison between network traces and user's schedule

As a burglar would not be looking for absences of mere minutes, some additional filtering was required. Figure 7 shows the original schedule and the combined trace, filtered for absences of more than 15, 30 and 60 minutes.

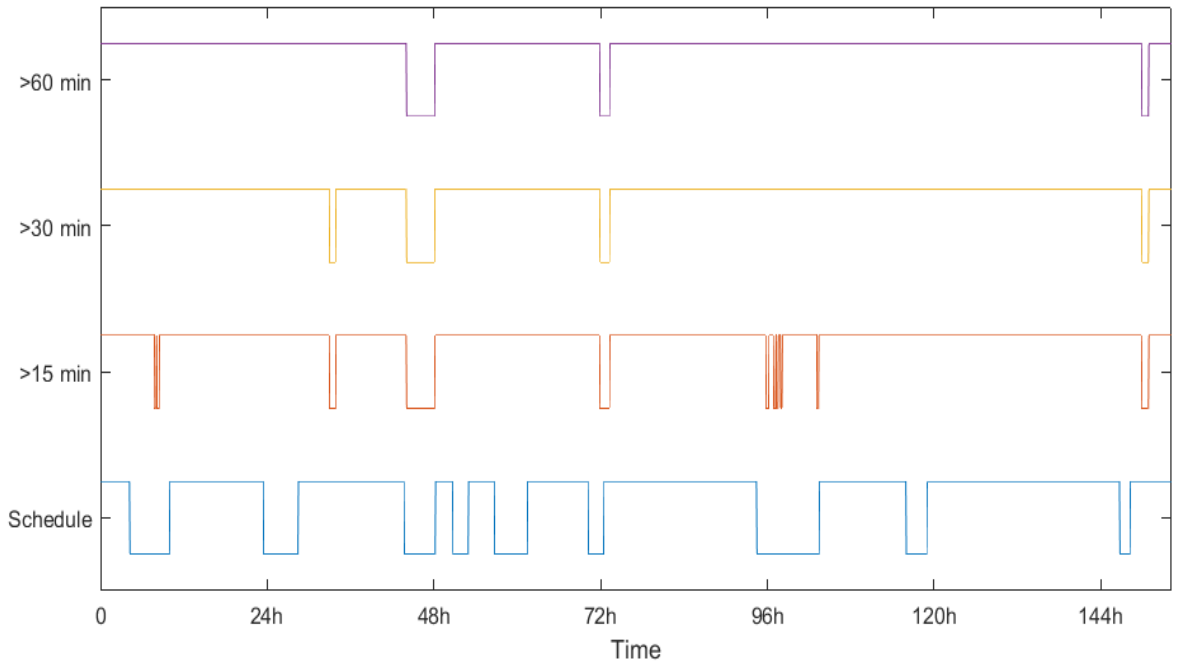


Figure 7: Dataset 1, comparison between the user's schedule and measured absences

At this point, it is a bit problematic to decide which offset between measurements and schedules can be regarded as still valid. For example, the absence at 72 hours is measured slightly later than the schedule states, but there is a reasonable overlap. Completely at the right of the graph, the measured absence is shifted free of the schedule. They are reasonably similar in length and a schedule error is not unlikely, but there is no definitive answer. At the other hand, the measured absence at approximately 33 hours is shifted a lot more from the long-scheduled absence starting at 24h. Additionally, the duration is completely different as well.
These uncertainties make it impossible to capture the result in numbers, but they do give an impression. In this dataset, the longest measured absence (just before the 48h mark) matches perfectly with the schedule. Should the burglar's measurements have returned this data, picking the longest absence would have been "safe".

2.4.4.3 Dataset 2

The second dataset yielded 5 potentially matching devices although none of them prove to be a perfect match. The schedule did not give much room for comparison as it only showed two absences. It is not unlikely that the resident forgot to register some (maybe shorter) absences.

However, Figure 8 shows that combining these devices still give useful information. The long absence from the schedule largely returns in the combined trace. The smaller absence in the combined trace also matches with the large vacant slot of the schedule, giving this prediction an almost perfect score.

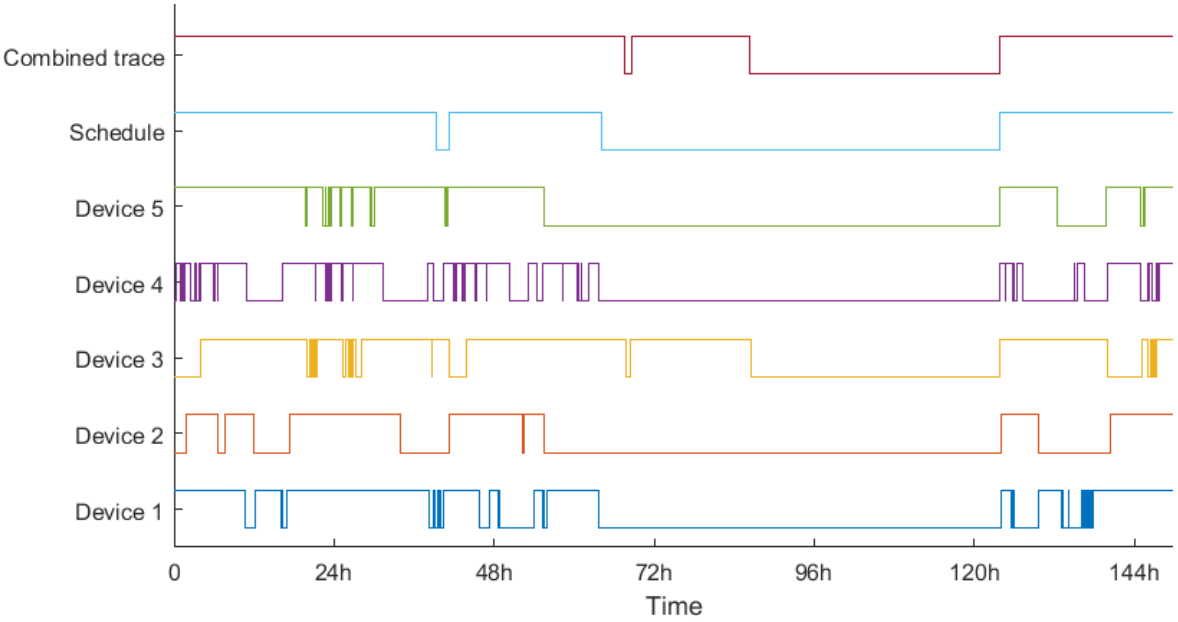


Figure 8: Dataset 2, comparison between network traces and the user's schedule

Filtering on absence length does not make a difference in this dataset. The small absence in the combined trace is still an hour long. The 3 filtered traces (15, 30 and 60 minutes) therefore yield exactly the same graph.

2.4.4.4 Dataset 3

Dataset 1 showed some “unstable” presence like a stationary device could create. In that dataset, it did not prove to be a large problem. This dataset however, has a device that influences the combined trace a lot.

Figure 9 shows the two devices recognized for this trace. One of which displays periodic activity when de resident is away from home.

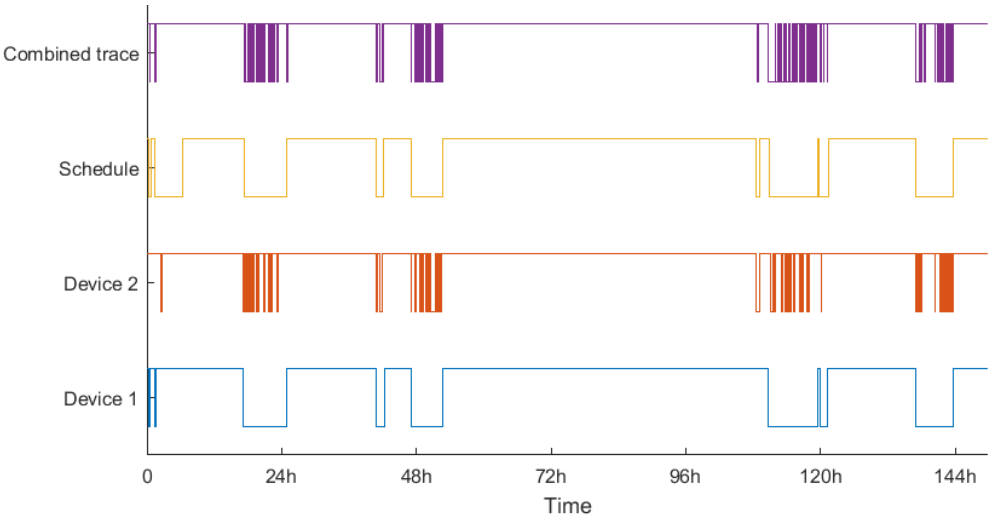


Figure 9: Dataset 3, comparison between network traces and the user's schedule

When filtering this combined trace for periods of 15, 30 and 60 minutes, only a couple of options remain with a maximum length of just over an hour. Meanwhile, the schedule shows plenty of opportunities.

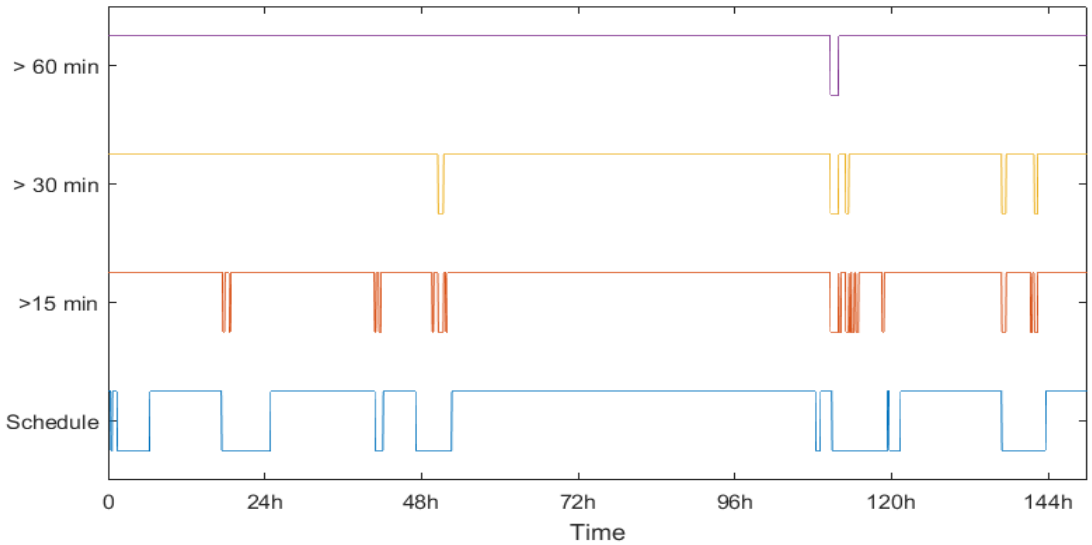


Figure 10: Dataset 3, comparison between the user's schedule and measured absences

Fortunately, for a burglar, the stationary device is recognized easily. Additional filtering or manual adjustments could still reveal the real absences which device 1 clearly shows.

2.4.4.5 Dataset 4

Also, with 3 recognized devices, dataset 4 also shows some “unstable” behaviour, especially in device 1. However, the influence is a lot smaller. Figure 11 shows that the large absences are still recognized, although the largest absence is divided in multiple pieces.

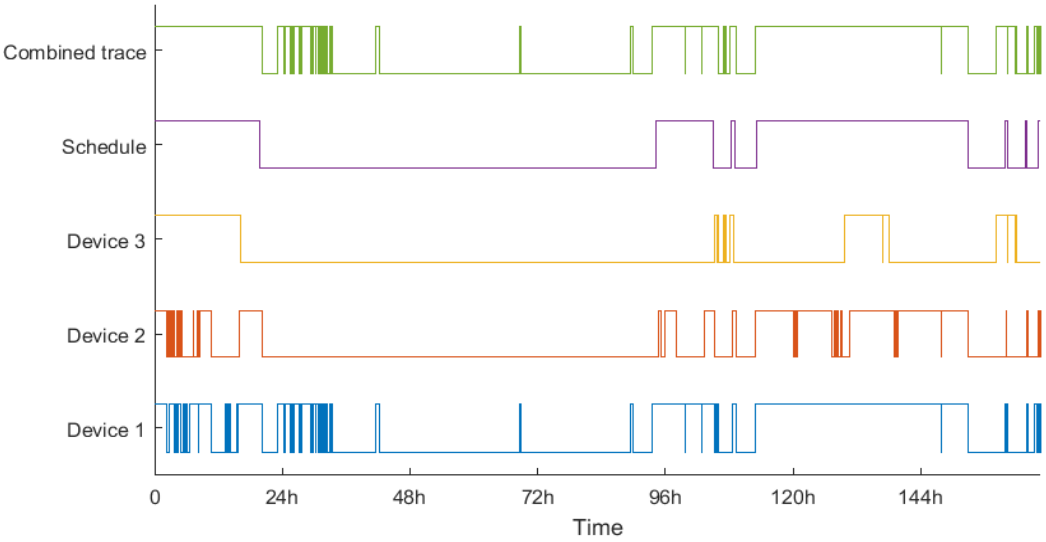


Figure 11: Dataset 4, comparison between network traces and the user's schedule

Filtering with 15, 30 and 60 minute thresholds barely influences the combined trace apart from removing some of the fast switching. However, a burglar would have already chosen the large absence.

2.4.4.6 Dataset 5

In this dataset, two devices were found to be matching the schedule. However strangely, both traces were virtually identical to each other and the schedule. The combined trace of Figure 12 therefore needs no further filtering. The data already matches the schedule without any mistakes.

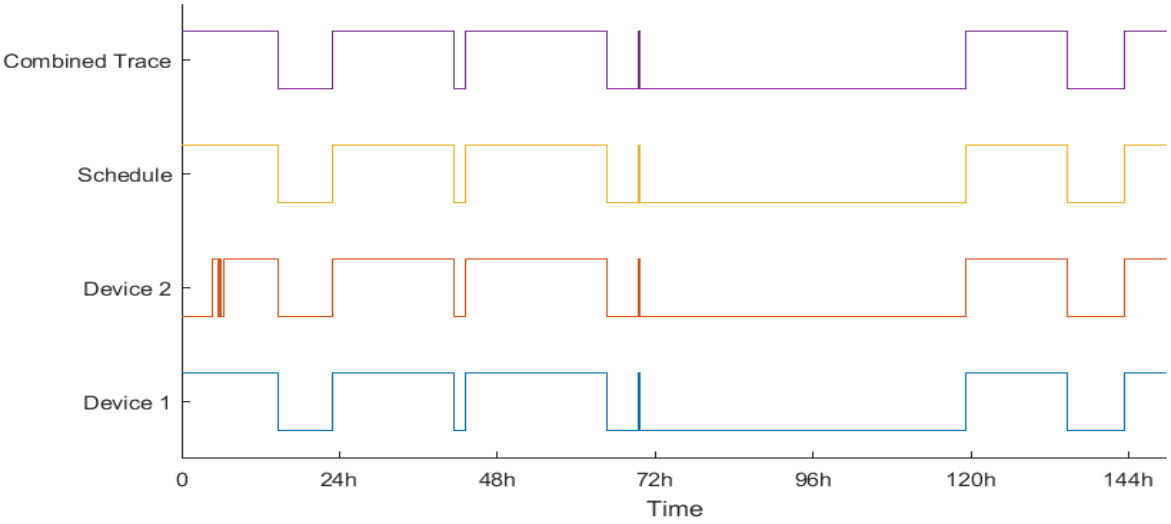


Figure 12: Dataset 5, comparison between network traces and the user's schedule

2.4.4.7 Dataset 6

Similar to dataset 5, both devices in this dataset are similar to the schedule. The combined trace therefore matches very well. However, Figure 13 shows the potential risk of using this kind of presence tracking. The schedule states that the resident was home at approximately the 130-hour mark, but both devices were silent. This would be a risk, should the burglar decide to abuse that "absence".

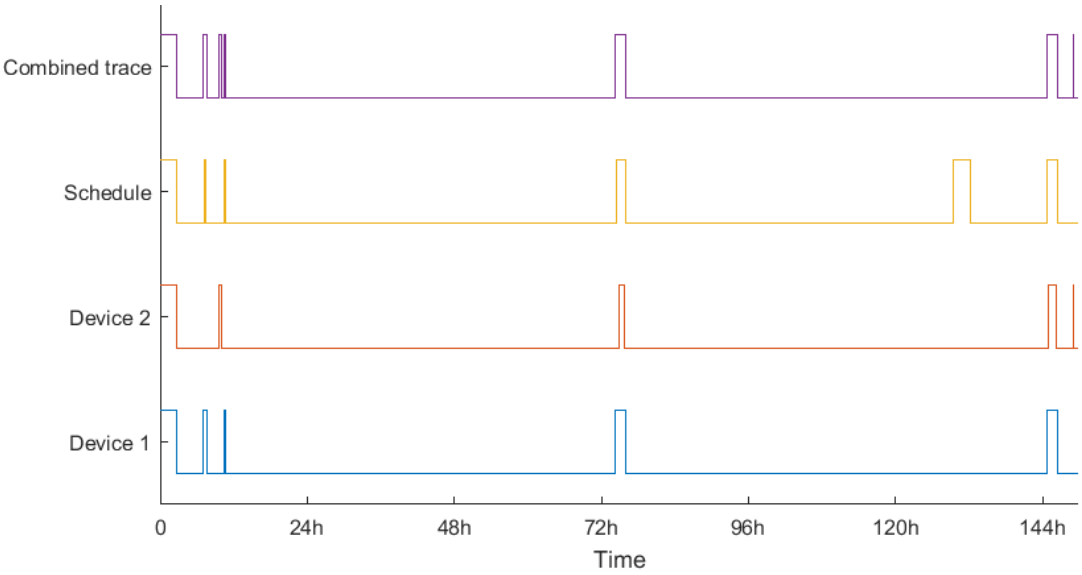


Figure 13: Dataset 6, comparison between network traces and the user's schedule

2.4.4.8 Dataset 7

The last dataset had 3 matching devices as shown in Figure 14.

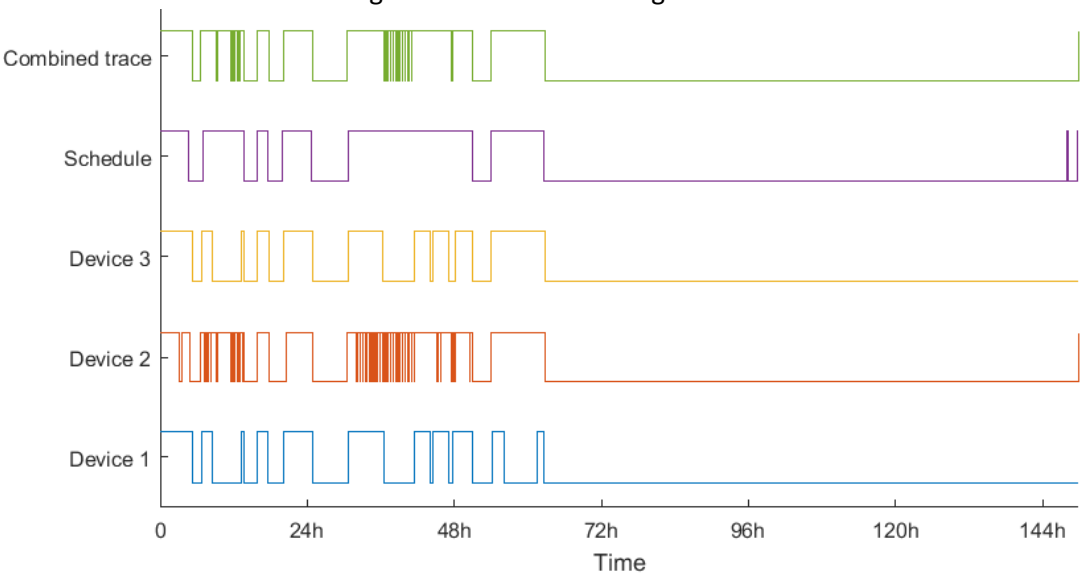


Figure 14: Dataset 7, comparison between network traces and the user's schedule

Device 2 introduces some "unstable" behaviour, but this time it prevents false vacant predictions at approximately 12 and 40 hours. The remaining absences all match with the schedule, especially the main absence of a couple of days.

2.4.4.9 Combining top SNR devices

The usability of the previous results is severely limited. It only shows that device data could be used to determine occupancy. But to be able to do that, a burglar still has to extract the right devices from the full dataset without the prior knowledge of the schedule. When operating in a normal house network, the network itself will only be used by residents and maybe visitors removing this problem. Unfortunately, reliably extracting the appropriate devices from the large Eduroam dataset has proved impossible. This prevents us from proving that these techniques are reliable.

However, some use may still be present in the dataset. As stated earlier, a burglar is only interested in one opportunity, as long as it is reliable. Maybe, the filters were not perfect, but still good enough. To test this, the dataset is initially sorted by signal strength. Afterwards, the first device is taken and compared to the schedule, then the first 2 devices are taken together and compared and so on. The manually recognized devices mostly resided in the top part of the dataset when sorted by signal strength.

All these combinations produce a certain relation between the correct and false vacancy predictions. This relation, with the imperfections of "wrong" devices, may still be able to deliver usable data for a burglar. Figure 15 shows these combinations and their average false vacant and correct vacant scores as a part of the total vacancy displayed by the schedule.

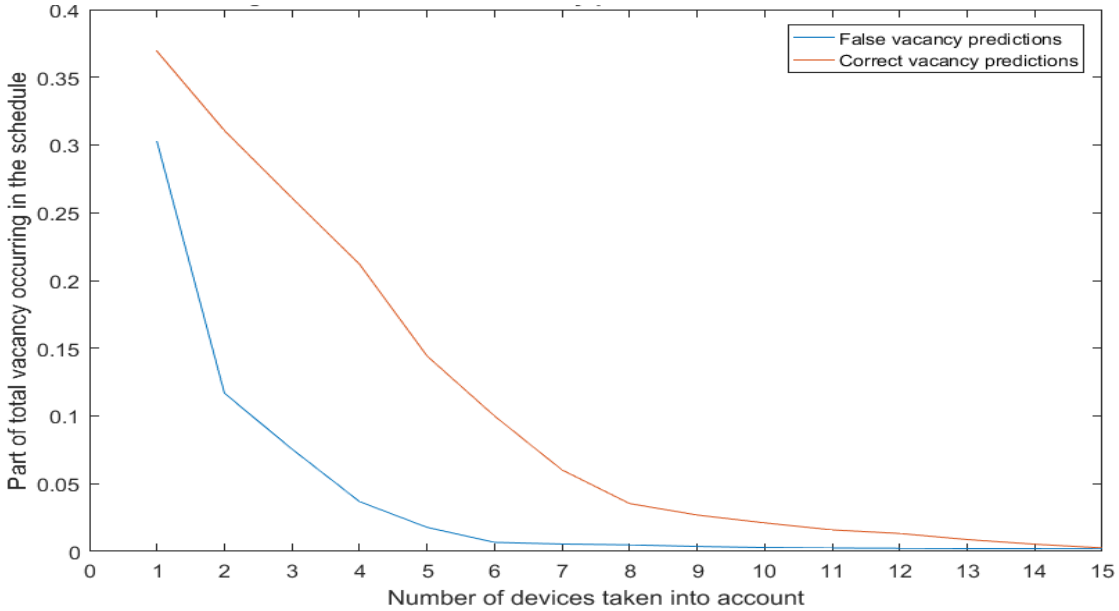


Figure 15: Average false and correct vacancy prediction rate versus device count

The graph clearly shows that the false vacant occurrences decline as the number of devices increase, but so do the correct vacant occurrences. Instead of getting a sweet spot where the false vacant predictions become negligible and true vacant predictions still occur often, both factors follow reasonably similar declines.

The relative sweet spot seems to lie at 6 devices, but only 10% of the actual vacancies is still measured. And of those vacancies, 6% is false. Note that this graph shows an average across the 14 datasets with manually recognized devices. In a lot of household, the burglar will run a lot more risk as the spread between these are very large. Figure 16, 18 and 19 show three widely different examples of these datasets.

It is rather obvious that a usable uniform tactic does not work here.

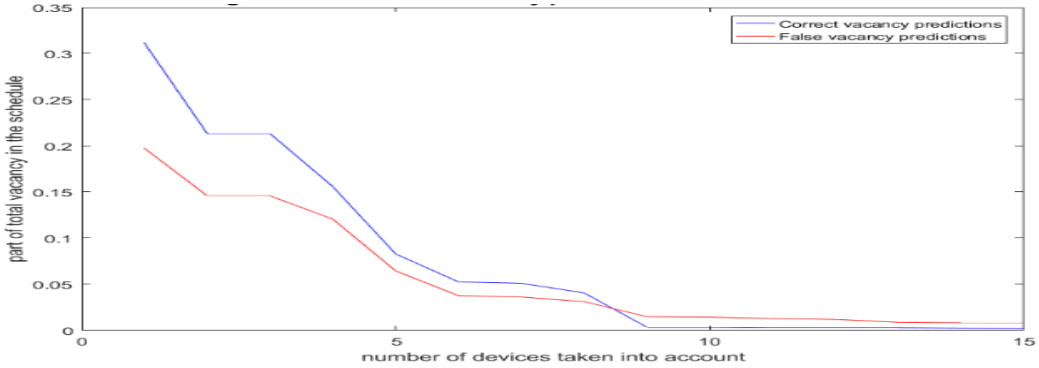


Figure 16: Average false and correct vacancy prediction rate versus device count #1

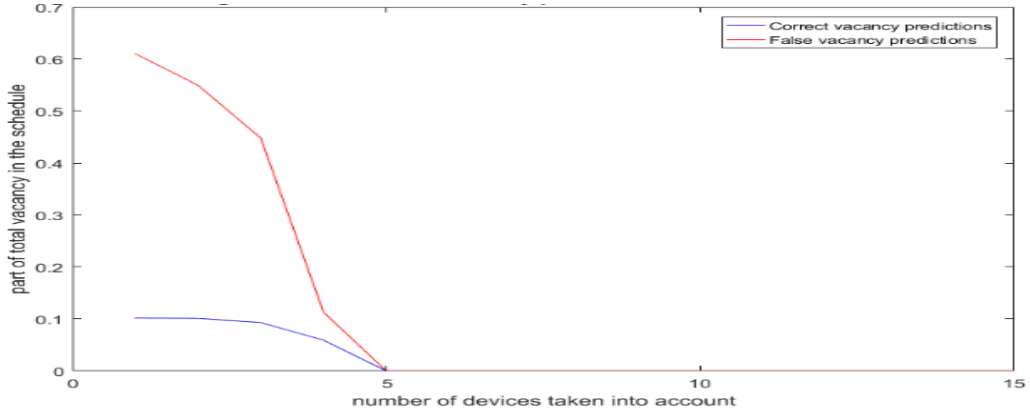


Figure 17: Average false and correct vacancy prediction rate versus device count #2

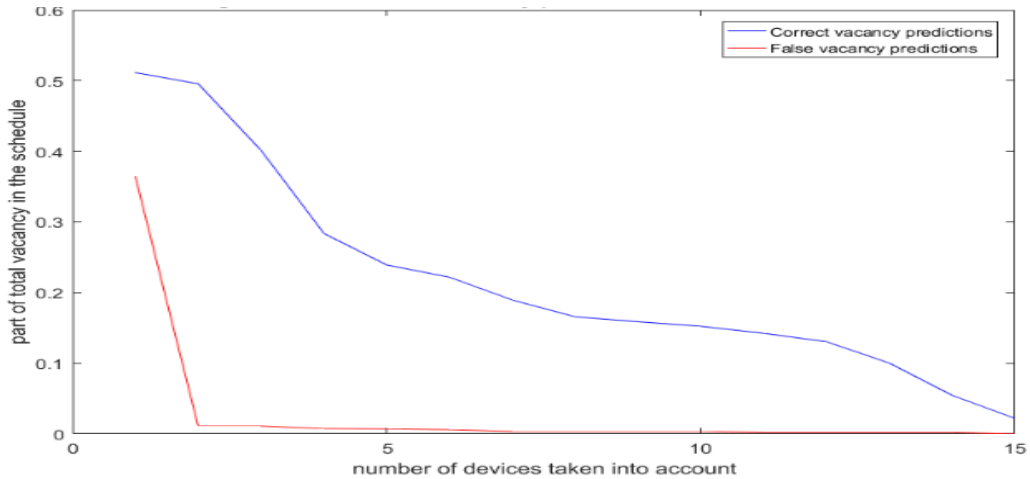


Figure 18: Average false and correct vacancy prediction rate versus device count #3

Unfortunately, the filters introduced in part 2 are not very helpful here either. The session count and session length filters require significant difference between “good” and “bad” devices, but almost all devices in the top segment of the list (with high signal strength) display a reasonable pattern for a person.

With that option not working either, the possibilities are rather exhausted. Unfortunately, isolating certain households from the near-public Eduroam network has not succeeded. As all households used this network, no data remained to test the main hypothesis that household occupancy can be determined by that house’s Wi-Fi data.

2.5 Problems and solutions

During the research, several issues arose with different consequences.

2.5.1 Limited effective time for data gathering

Initially, creating and testing the measurement equipment's software took more time than expected. Afterwards, the first measurement round unveiled some unknown issues making this measurement round useless. Combined with the time limit of the upcoming summer holiday, at which most residents would be away from home for prolonged periods of time, limited the amount of measurements that could be performed. Due to limited available time for the research project and all the other work that needed to be performed, continuation after the summer holiday was not a viable option.

To achieve larger amounts of datasets, a larger timespan would have been needed. More equipment would not have made such a difference in the current setting as on most occasions, not all equipment would be distributed at the same time. The distributions of letters, visiting the residents and collecting of the gathered data required large amounts of time. Improving this part, especially the visits, would free a lot of time.

A potential option would be to increase the size of the household lists substantially. In the current experiment, random selections of 60 households were made. After a substantial amount of these were tried (and participated or declined), another 60 were added. These household would be scattered all around the campus' living quarters, which takes a lot of time to cover. Increasing this list drastically or maybe even dropping the randomized part (although this may interfere with the defendability of the research) increases the number of households in every building and therefore saves a lot of "travel time" between them. This approach may require extra measurement equipment for extra effectiveness as this large set of households will yield more participants in a single round.

Putting the initiative for participation at the residents would also save a large amount of time. Instead of visiting and asking for participation, the introductory letter could ask people to contact us via, for example, an email, a web form, etc. This would save huge amounts of time, but it is expected that people will be less inclined to participate if they have to put in effort. However, with more than 500 households on the campus, it may be a valuable addition to get an initial batch of participants. Visits to ask for participants could still be performed afterwards to the households that did not respond.

2.5.2 The shared Wi-Fi network (Eduroam)

All buildings on the University's campus are fitted with wireless access points distributing the Eduroam network. This induced the problem that devices are no longer linked to a specific household as they would be in most normal households. This issue was known beforehand, but was expected to be countered by using SNR readings and other factors to determine "in-house" equipment.

Unfortunately, this proved to be problematic. In the households where devices were recognized manually with the resident's schematic, the devices would rank very high in this classification. But in a lot of households, the difference between them and some other devices was marginal. Other devices would also frequently be classed higher than a device actually belonging to that household. With these datasets, that meant that reliable filtering of "correct" devices was impossible without the prior knowledge of the resident's schematics.

Another issue was the apparent lack of matching devices in a large number of datasets. Although exact reasons are unclear, part of it seems to be connected to the limited coverage of the Eduroam network. Some participants stated that the Eduroam network in their living quarters was very weak and unreliable. This resulted in residents disabling their Wi-Fi functionality on their devices and reverting to the cellular network. Other residents created a personal Wi-Fi network in their homes. As long as this network would reside on one of the three super channels, also used by the Eduroam network, the measurement device would still be able to trace the network. But if another channel was used, they would fall outside of our measured frequencies and remain invisible.

There is also a possibility of residents using the wired network for devices like their computer. However, this is also an expectable factor in normal households.

2.5.3 Reliability of timesheets

The gathered network data was to be compared to the timesheet filled in by the occupant. However, there is no real way to determine the reliability of the schedule. In the datasets where devices were manually recognized, the timesheet was obviously comparable to the gathered data. But in the other datasets, this is an unknown factor.

Some timesheets show a very unusual schedule. This does not definitively say, that the sheet was filled-in incorrectly, but it does give that impression. Most notably is the timesheet of which one day is shown in Figure 19 below:

Figure 19: Example timesheet

The rest of the timesheet shows similar absences of only 15 or 30 minutes, a couple of times a day. Although there is no definitive way of determining if this timesheet is correct, the strange schedule and the fact that no matching devices were present do give an impression that the schedule is not correct.
Without the shared Wi-Fi network, a burglar would not have to find matching devices out of a list. If the devices would represent this schedule, it would be quickly apparent that there are not really any usable timeslots in which the house is vacant. It is most likely that the burglar would skip this household in favour of an “easier” one.

2.5.4 Absence of RTC

As already mentioned earlier in the report, the chosen measurement devices lacked a way of keeping time while unpowered. So, every time a device was placed, it would start measuring at the first of January at 0:00. Although this does not influence the measurements themselves, it does remove any synchronisation possibility with the schedule. A rough start time of the measurement may be known, but there is a lot of possibility for offset to occur. This was also seen in the datasets with visually matched devices. Data was shifted manually in relation to the given schedule, but the amount of shifting varied a lot.

This unknown timestep between the data and schedule also prohibited the use of datasets without obviously matching devices. Without knowing what offset to choose, the results would not be defendable.

2.6 Conclusion

This research was intended as a follow-up on a similar experiment. Instead of a small-scale experiment among (potentially biased) relatives, this research would be able to prove the risk of presence detection by Wi-Fi eavesdropping with enough statistical relevance.

The main question for this was similar to that earlier experiment.

- *Is it possible to reliably track occupancy in a household with passive eavesdropping on its Wi-Fi traffic?*

The experiment originally yielded 55 participating households where a minimum of 43 was set. Due to soft- and hardware problems, an administrative error and incorrectly filled in or missing forms the resulting number of datasets stuck at only 25.

Apart from the lower-than anticipated number of usable datasets, the research question proved impossible to answer in this experiment. This was mainly due to the chosen circumstances. Because of ethical considerations, households with underage or mentally challenged people were off limits. Therefore, student housing was chosen which gave the added complexity of a shared Wi-Fi network as compared to per-household networks.

This gave a second research question to be answered:

- Is it possible to determine which Wi-Fi devices belong to a certain household in a shared network with only passively detectable parameters?

Unfortunately, this proved to be difficult. The filters created to separate the devices belonging to the household from the rest were only partially effective at best. They reduced the number of devices, but optimal settings varied between households and “external” devices often remained in the dataset. Stricter settings resulted in correct devices being filtered out. Looking at communication between devices in the same household did not help either. It turned out that intercommunication happened everywhere in the network, regardless of the devices belong to the same household or not.

Eventually, manual selection of devices matching the schedule was performed as a last resort to get some usable data. Of the 25 remaining datasets, similarly behaving devices were only found in 14 of the datasets. Retuning the filters with this knowledge still did not return any usable filter settings. Matching devices usually showed high SNR figures as expected, but there often would be others too. This made predictions without prior knowledge completely unreliable.
Comparing the matched devices to their accompanying schedule does show that Wi-Fi data can represent actual occupancy. The predictions from the Wi-Fi data showed a match rate with the schedule of 87,80 percent. Unfortunately, this was only possible with visually matched devices using prior knowledge of the user’s schedule.
In the end, the chosen Eduroam environment proved to be very difficult to deal with. Although devices did show remarkable similarities with the user’s actual schedule, separating those devices from different households was not successful. This prevented a definitive answer to the main question of this research.

2.7 Discussion

The main goal of this research was to prove that occupancy detection from a household's Wi-Fi traffic was possible. Unfortunately, this question proved impossible to tackle with the chosen circumstances. This was mainly caused by the shared Eduroam network in the chosen student living quarters. A normal household would have its own private network, making all devices on it relevant for the burglar. In the chosen situation, it proved impossible to reliably separate devices from each other based on their household. This prevented any proper research into the main goal.

Visually matching devices did show that network traces can show occupancy, but these results are not really defendable as it only covered a small amount of household and required prior knowledge.

To allow for better results, a number of potential improvements have been thought of on the way.

The first and most straightforward one would be to stop using the shared network environment and revert to normal households. The reason this was not done in this experiment was a limitation posted by the ethical board preventing us to use houses with underage or mentally challenged people. In normal neighbourhoods, this omits a large number of houses making the experiment a lot more cumbersome.

However, the given limitation is, from our point of view, not really necessary. The main reason for this restriction would be that these people would not be able to understand the privacy risk that they are exposed of. However, again in our opinion, there is no realistic risk in this. The data stored is not linked to any house or person. It also does not say anything about the timeframe in which the experiment was conducted. Should someone obtain the data and somehow find out which house it belonged to and who carries which device in that house, it would still be old data of unknown age. If someone is going through so many lengths to obtain data, they are better of gathering it themselves.

When looking at the experiment as it was conducted, a couple of other improvements should be put in place.

To improve the efficiency of deploying the devices, larger groups of people should be contacted at the same time. In this case, a group of 60 households was chosen at a time. After a reasonable part of that participated or declined, an extra 60 households were added. As these households were randomly scattered around the campus, it took a large amount of time to visit them all. Instead, adding more households to the "active" list makes the rounds more efficient. Some research has to be done into the maximum possible list size while preventing a bias in the list.

Additionally, a passage could be added to the introductory letter to ask people to contact us when they would like to participate. This would not replace going to all the housed, but could give an initial list of participants to go to. The sooner a device is set out, the sooner it is back and can be set out again. As the number of devices is limited, this addition could aid in a more efficient use of the equipment.

As for the measurement device, a real-time clock system would be advisory. As each device starts at the first of January 1970 (zero Unix time), there is no synchronisation with the schedule. This required manual verification of the offset by comparing the schedule to visually matching devices. Although an indication of the offset was often possible to give, the exact values would be a guess. A real-time clock omits this issue as the data timestamps comply with the times on the schedule sheets.

Another option would be to add a form of communication to the device. In this experiment, Wi-Fi would be the most obvious. Initially, the devices had such functionality. When booted, they would set up a management network to connect to. This allowed some checking of functionality. Unfortunately, this system introduced more reliability problems after which it was disabled. This system was also limited to the initial boot. A better system would be continuous or periodic communication options. This would allow for periodic checks, preventing small errors to ruin complete datasets. However, this functionality would require an extra network adapter or a periodic downtime on one of the channels while the communication path is opened temporarily.
A potential way to be able to still use Eduroam for this kind of measurements would be to apply triangulation. This would require more devices to be placed in the building simultaneously. With the various snr readings, triangulation may be usable to determine the exact location of the measured device. This would however require multiple devices for one household. A superior option would then be to spread numerous devices across one building/street and measure all living quarters at once. This would however, be a nightmare to arrange with all the residents.

3 Related solution research

3.1 Introduction

There are already many implementations, which reduce the trackability of Wi-Fi enabled devices. In this chapter these implementations are divided into different categories. The implementations are then shortly explained: how they work and their similarity with other solutions. After which a comparison is made between the different categories. This comparison is then used to define the goal of this research.

3.2 Awareness

The first category is not about solving the trackability problem but simply about improving the awareness of the trackability/privacy problems.

The paper “PriFi beacons” (Könings et al., 2013) is not actually a solution to the trackability problem but more a solution that improves the awareness of the problem. This is done by adding information about privacy implementations onto beacons transmitted by the access point. They then implemented an application, in android, which reads this information and shows it to the users. This is not an actual solution to the problem but is still important because it tries to improve the awareness of privacy problems with the current protocol.

3.3 Passive probe

The second category is called passive probe, this is because it involves probe requests/responses and it does not actually change the protocol in any way (that is why it is passive in a sense). This is mostly done by randomizing the sender mac address and removing SSIDs from the probe request.

The paper “How talkative is your mobile device?”(Freudiger, 2015) tries to quantify how many probe requests an average smartphone shares with the world. It also researches how effective the implementation of mac randomization is by testing the implementation of an Iphone with IOS 8, which includes these features. Their results show that mobile phones send on average about 55 probe requests per hour. This part of the research shows mainly that the current implementation of the Wi-Fi protocol makes tracking of phones very easy. Furthermore, the effectiveness of the randomizing the mac addresses is shown to be very ineffective in its current implementation. The first problem with its implementation is that it only works while the devices are asleep, it does not randomize its mac while awake. The second problem is that it does not change other fields in the header that could be used to link packets together.

3.4 Active probe

The third category is called active probe because changes are made to the protocol to the probe requests and responses to decrease the trackability and improve privacy.

The paper “Security analysis and authentication improvement for IEEE 802.11i specification” (Xing et al., 2008) tries to improve the authentication method of the 802.11i standard by using asymmetric cryptography, which protects the frames from linker layer up.

The paper “Privacy-preserving 802.11 access-point discovery” (Lindqvist et al., 2009) is about changes in the 802.11 protocol to preserve privacy in access point discovery. The proposed changes are minimally enough that interoperability with the 802.11 protocol is kept. Currently a client transmits a probe request to which an access point can respond with a probe response. After which authentication and association requests are transmitted back and forth to complete the connection. The problem with these packets is that identifiable data is sent in all these packets. This paper solves this by removing the identifiable data from the probe packets and uses randomized MAC address for the association and authentication packets (as shown in Figure 20).

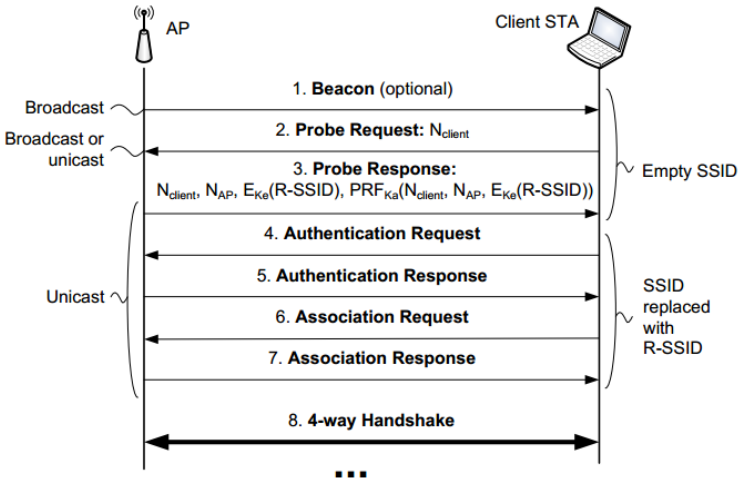


Figure 20: privacy preserving discovery (Lindqvist et al. 2009, figure 1)

3.5 Passive mac

The fourth category improves on the probe categories as it actually improves multiple parts of the 802.11 standard that leak information about the device. This again is done without changing the protocol. In the paper “Enhancing location privacy in wireless LAN through disposable interface identifiers” (Gruteser and Grunwald, 2005), they try to reduce trackability by using disposable MAC addresses and renewing these addresses regularly.

In this implementation, they periodically generate a new MAC address after which they reconnect to their network if they were connected. This solution does have some drawbacks, since a new connection is made every time the address is changed; no active connections are possible while changing identifiers.

The paper “802.11 user fingerprinting” show that mac randomization does not work alone as other identifiers in the MAC header make it possible to still track a device (like packet size)

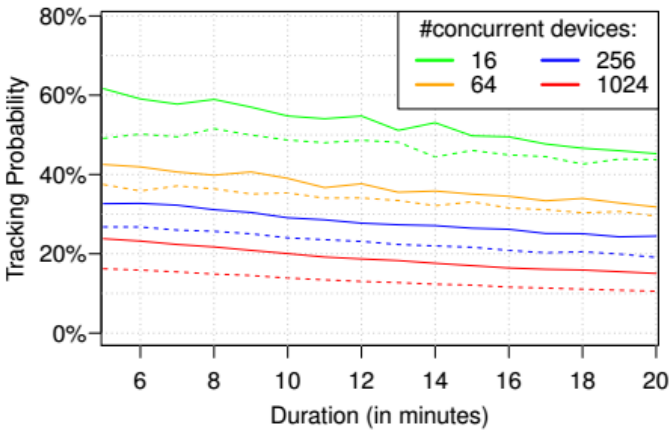


Figure 21: Probability of tracking devices (Vanhoef et al. 2016, figure 6)

As does the paper “Why MAC Address Randomization is not Enough “(Vanhoef et al., 2016) show that using random mac addresses are not enough to prevent the client from being tracked, as shown in Figure 21 the probability of a device being tracked is above 50% with 16 devices connected. This is due to other information that is leaked by the client. Furthermore, they show that they could get the real mac address using two methods that are described in the paper.

3.6 Active mac

The last category does the same as the previous categories but changes the protocol to get the result.

In the paper "Improving wireless privacy with an identifier-free link layer protocol " (Greenstein et al., 2008) they solve the trackability of the client by obfuscating all the data that could be used to identify the client. This is done by encapsulating all the packets with their own layer that encrypts these packets. They created two mechanisms for this, one for discovery and binding (probe and authentication packets, Tryst in Figure 22) and another mechanism to encapsulate the data packets (Shroud).

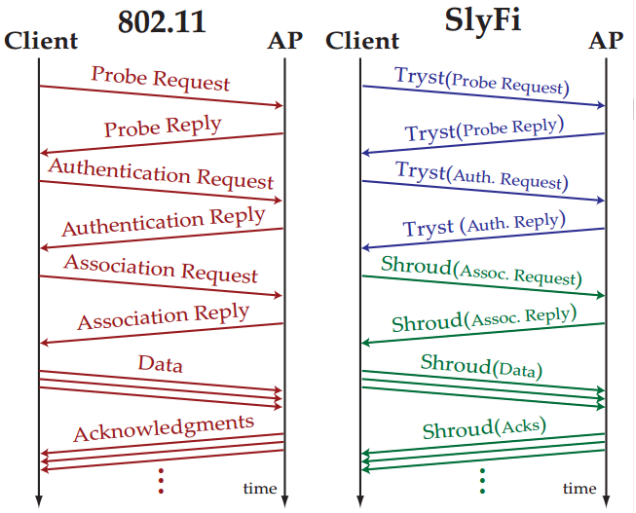


Figure 22: SlyFi Protocol (Greenstein et al. 2008, figure 1)

They split the mechanisms because there are different requirements between these types of packets. The first one is sent sparsely and there is no connection between the access point and client. The second mechanism is implemented to make sending large amounts of packets possible, though this uses data shared in the authentication phase.

And in another paper called "Header Encryption of IEEE802.15.4"(Dalal et al., 2016) they use AES encryption to encrypt the mac header. The goal is to prevent attacks against the network by encrypting the header, but by doing so they also make tracking of devices harder.

3.7 Comparison

To make a good comparison of each category, criteria to which they are compared are defined:

1. How well does the solution solve the trackability problem?
2. How much does it change to protocol?
3. Could it be rolled out easily onto current devices?
4. How much impact it has on connectivity?

The first solution will be skipped in this comparison due to it not actually changing the trackability.

3.7.1 Trackability

The passive probe category does not actually impede the trackability much due to restrictions on when the randomizing is used. In addition, it only changes the addresses and no other fields thus those other fields could be used to make tracking possible. Furthermore, it is only used on probe requests, when a client transmits data to the network they will still be trackable.

The active probe category does minimize trackability but only for probe and authentication part of the protocol thus when the client is connected to a wireless access point it could still be tracked.

The passive mac category minimizes tracking of the clients but it might still be possible. It all depends on how often changing of the identifiers is done.

The last category: active mac solves the tracking of clients.

3.7.2 Protocol change & roll out

Passive probe and passive mac category both do not change the protocol thus do not hamper roll out.

The active probe category makes minor changes to the protocol; these changes are interoperable with the normal protocol. This makes roll outs for this implementation very easy since a gradual roll out of this technique will be possible.

The active mac category makes large changes to the protocol and there is no talk about interoperable, but due to the changes made, it is expected it will not be interoperable with the older protocols. This of course also means that roll out of this technique will be harder since no gradual roll out is possible.

3.7.3 Impact on connectivity

Passive probe, active probe and active mac categories all have minimal impact on connectivity.

The passive mac category impact on connectivity is largely dependent on the frequency of changing the identifiers. No connection is possible whilst changing the identifier thus no persistent connections on the client are possible.

3.8 Conclusion

Table 2: Overview of comparison

	Trackability	Protocol change	Roll out	Connectivity
Passive probe	--	++	++	++
Active probe	-	-/+	+	++
Passive mac	+	++	++	--
Active mac	++	--	--	++

The summery of the results in the table above clearly shows that all categories have drawbacks and only two of the four categories even propose a solution to the problem. However, both categories also have big drawbacks: Passive mac, in the fact that connectivity is severely hampered by the solutions and active mac, due to that it requires big changes to the protocol that will hamper interoperability and roll out of the solution.

4 Goal

In the individual part of the research, a solution is researched to solve the problem described in the previous chapters. Thus, a research question needed to be formulated:

- Is it possible to find a solution that prevents tracking of clients whilst keeping interoperability with current implementations?

In other words, the goal would be to place the solution in between the passive and active mac solutions that were described in the previous chapter. It will still have some downsides in the fact that some pieces of the protocol have to be changed/adapted, but the goal would be to make no changes to the actual packet layout but instead only encrypt the content of fields inside the packet. This encryption would then prevent the attackers from reading the fields, though it would still allow older devices to still read and understand the packets but it would then ignore these because it does not understand the content of it (interoperability). Furthermore, some newer devices have parts of the packet handling implemented in hardware, which limits the amount of changes possible to the protocol. By not changing the packet layout, it is expected that new solution would also work on these devices (also interoperability with current implementations). It also needs to give comparable privacy protection as the passive and active mac solution but without the downside of losing active connections.

Translating this into the table shown in the previous chapter that summed up already researched solutions to this problem would alter the table to the following:

Table 3: Overview with new solution

	Trackability	Protocol change	Roll out	Connectivity
Passive probe	--	++	++	++
Active probe	-	-/+	+	++
Passive mac	+	++	++	--
My solution	++	-/+	+	++
Active mac	++	--	--	++

5 Approach

Too systematically work to a solution that prevents users from being tracked which still is interoperable with older implementations an approach would have to be defined. But before doing this, the goal has to be to be further refined. In this case the goal is to decrease the trackability of devices whilst staying interoperable with older implementations. The idea is that this can be achieved by encrypting the data that makes tracking possible in such a way that the original format will not change and thus keep interoperability with older devices. To research this, the goal is divided into multiple smaller parts:

First, the data that makes user trackable in the current 802.11 protocol implementation needed to be researched and identified.

Secondly, the 802.11 protocol had to be implemented in Proverif. The data that was thought to leak information would then be tagged as private data in the Proverif implementation. Proverif would then be run to check that it could actually check if the data was actually leaked.

Thirdly, the protocol would then be changed in such way that it would still be interoperable with the current implementation but would not leak any information any more. Proverif would then be used to verify that it does not leak any of the tagged data any more.

Lastly, to verify that the new protocol would still be interoperable with the old 802.11 implementation it would have to be implemented on an access point and client. Interoperability could then be tested using these devices and devices that have not been changed. Furthermore, the protocol could then be analysed with normal Wi-Fi analytic tools (like Wireshark) to verify that the solution was correctly implemented.

6 Identifying the problem

The 802.11 protocol defines a messaging format which is used to facilitate the communication between wireless devices. Earlier research shows that parts of these messages could be used to track clients. To determine which parts of the protocol gives an attacker the ability to track these clients a method had to be devised. It was chosen to start with the premises that all fields in the protocol leak information about the client and only mark them as unusable if proper argumentation could be given why it couldn't be used to track the client. A differentiation was to be made between following types:

- Unusable, the field could not be used to track the client.
- Wi-Fi usage tracking, this field would allow an attacker to determine whether a Wi-Fi network was in use by any clients.
- Client tracking, this field would allow an attacker to track the clients on a Wi-Fi network separately.
- Required, either of the above are true but changing this field would impact the working of the protocol in such a manner that legacy clients would not be able to join the network any more.

Many of the frames below have an overview of which fields are present in that frame. In this overview the fields are colour coded to quickly show to which type it belongs, the colours and associated types are shown in Table 4.

Table 4: Type colour coding

	To be determined
	Unusable
	Wi-Fi usage tracking
	Client tracking
	Required

6.1 Frame types

In the 802.11 standard, three different frame types are defined. The first is the data frame, the reason to use this protocol, to transmit and receive data, which is done in this packet. The second are control frames, these control access to wireless communication medium and thus help delivery of data frames. And lastly the management frames. These provide services like identification of networks, authentication, association with networks and other important features. As will be seen in these next chapters, these frames will be transmitted in a certain sequence to get a working communication protocol.

6.2 Generic MAC frame

First a generic frame will be discussed, which is shown in Figure 23, after which we will group the frames together in their most common uses (later called a sequence of frames) which will give us an insight into how these devices communicate wirelessly but also important to determine which types of frames and their fields need to be altered to prevent wireless clients from being tracked by adversaries. Then all fields not already discussed in the generic frame will be discussed on a frame per frame base used in that particular sequence.

Bytes	2	2	6	6	6	2	6	0-#	4
Field	Frame Control	Duration ID	Address 1	Address 2	Address 3	Seq Cont	Address 4	Frame body	FCS

Figure 23: Generic MAC frame header

The generic frame consists of the following fields:

- Frame control
- Duration/ID
- Address 1-4
- Sequence ID
- Frame body
- Frame check sequence

6.2.1 Frame control

The frame control field consists of multiple control bits, which are portrayed in Figure 24 and listed below with further explanation:

Total	2		2		4				1	1	1	1	1	1	1	1
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Protocol		Type		Sub type				To DS	From DS	More Frag	Retry	Pwr Mgmt	More Data	WEP bit	Order

Figure 24: Frame control field

- Protocol version, is used to determine the version of the protocol but there is only one version in used today, thus cannot be used to track a client.
- Type and subtype, determines the type of frame, this in itself cannot be used to differentiate between clients. But it might be used to statistically determine what kind of clients are available in the network which might be used to determine if a network/house still holds active clients. But due to its importance in the protocol, it defines how the data is interpreted by the client, it cannot be encrypted/hashed in a way that keeps it compatible with legacy clients therefore this is tagged as a required field for the protocol.
- ToDS and FromDS, determines the destination of packet. There are 4 different options that will determine the actual source and destination of the packet:
    - Management and control frames that are not destined for clients, this for example includes station to station communication.
    - Traffic from an access point to a client
    - Traffic from a client to an access point
    - Packets that include all four mac addresses, which are usually used in wireless distribution systems (WDS) like a wireless bridge or mesh network.

Option one and four do give information about the layout of the network, as the first could show that the network consists of multiple stations and the fourth shows that a WDS is used. Option two and three are used mainly by clients as that is the communication between access point and clients, thus it should not leak any information about connected clients. Combining this with the fact that these fields are used by clients to determine the format of the frame that is received, the choice was made to tag this field as required.

More fragments bit, is set to one when the packet is fragmented into multiple packets. All clients are able to transmit big packets which in term will mean many fragmented packets. Thus, this does not provide much data to able to clients.

- Retry, used to determine if a packet is retransmitted. This does implicate a previous transmission has not been received but it only links it to that packet. Therefore this bit does not provide much data to be used in tracking client's other than a link to the previous failed packet.
- Power management bit, indicates if the sender will go into a power saving mode after the frame exchange is finished. This allows clients with limited power supply to preserve energy but still stay connected. This bit therefore might indicate if a client is a client which does not get its power from the mains. This does not give the ability to track clients but might give an indication about what kind of clients are currently connected to the network. Therefore, this field is tagged as to aid Wi-Fi usage tracking.
- More data bit, this signals clients that more data is available from the access point. Clients that preserve power will use this to determine if they need to stay awake any longer or that it could go to a power preserving mode. Just like the previous bit, it does not give the ability to track clients but might give an indication about the topology of the network therefore this is also tagged as being used to track Wi-Fi usage.
- WEP/encryption bit, formerly known as the WEP bit because it indicated that WEP encryption was used to protect and authenticate the data but now it is used to signal the use of encryption. Now a day the use of encryption is widely enforced by the base station and thus would be the same for the whole network. Furthermore, without encryption enabled the attacker would be able to see the data that is transmitted by the client. Since this is highly likely to contain data that could identify the client it is not useful to use the new protocol without encrypting the data. Therefore, it is concluded that it cannot be used to track clients since without it being set to true (as encryption enabled) the new system would not have any impact on the trackability of the client.
- Order bit, determines if strict ordering is used to reorder the transmitted frames and fragments. As with the more fragments bit this cannot be used to track clients.

6.2.2 Duration/ID

The duration field is a field, which depending on the frame type can hold two different values. The first is the PS-Poll frame, in this frame the field is used to send the client associate ID to the base station. This type of frame is used in the energy saving features, whenever a client comes out of sleep it asks the access point for any messages which were buffered by the access point. The ID is used by the access point to identify which packets need to be transmitted to the client. In all other frame types the field is used for setting the Network Allocation Vector (NAV). This NAV is used to determine if the communication medium is used by another client or that it is available to send data. This allows the sender to reserve the communication medium for set amount of time, therefore the field is set to the expected time the sender will use the communication medium. The duration field cannot be used by an attacker to differentiate between clients in all other frame types, but the client is able to track clients with the PS-Poll frame type because an ID associated to the client is sent. Therefore encrypting this header for the first type is very important but not for the others, which is important when looking at the compatibility. Because normal clients use the value as a way to prevent more clients from using the same medium at the same time.

6.2.3 Address 1-4

A maximum of four MAC addresses are allowed in the protocol containing any of the five following values:

- Source address, this is the mac address of the original sending client
- Receiver address, the address of the receiving client/access point, this does not have to be the same as the destination address as the destination address might be out of range and thus a relay might be used in such cases.
- Transmitter address, this is the client which transmits the data, as with the receiver address this doesn't have to be the same as the source address.
- Destination address, the MAC address to which the packet is addressed to.
- Basic service set identifier (BSSID), the unique identifier of the network to which the client is associated, in most home cases this will be equal to the address of the access point.

Only 4 of the 5 addresses give the ability of an attacker to track a client but the fifth is also important because that one can be used to determine the usage of a network. In the case of the problem we researched this is very important because you only need the fifth to determine the activity of a network and thus possibly the presence of a user. Therefore all addresses are useful in some way and thus all need to be obscured to prevent tracking of said clients.

6.2.4 Sequence ID

The sequence ID field is divided in two parts; the first is the fragment number, which is used if original packet is fragmented into multiple packets. The number is used on the receiving end to put the original packet back together in the correct order. Identification of the client by the fragment number can be debated, on the one hand it can be used to link the fragments together and therefore the attacker will be able to track a client across the fragmented packet. On the other hand, the fragmentation field length is only 4 bits long and thus can only hold 16 unique values which limits the time the attacker can track the client to a maximum of 16 packets. The second part is the sequence number, this is field is used to discard duplicate frames and for reordering/defragmentation of the packets to make sure the client receives its packets in order if required. The content of the field is a simple counter which is increased every time a new packet is sent, if a packet is fragmented the same sequence number will be used but then the fragment number is used to reorder the packet into the correct order. The impact of this field on the trackability of the client is fairly big, a study has shown that even if all other fields are randomized this could still be used to track the client's due to its

predictability. Furthermore, the field itself can hold is 12 bits long and therefore can hold 4096 different values, which means that even if there are many clients the probability of the clients having the same sequence numbers is fairly small and thus could make tracking possible.

6.2.5 Frame body

Content depending on what kind of frame is transmitted, therefore this can only be discussed per frame type which is done in the next chapters.

6.2.6 Frame check sequence

Before a frame is transmitted, the frame check sequence (FCS) is calculated, it is used to determine the integrity of the data at the receiving end. Because the method of calculating is determined by the protocol, and thus fixed, it cannot be used to identify clients.

6.3 Sequences

6.3.1 Ack only

The first sequence, depicted in Figure 25, encompasses the acknowledgement (ACK) frame. This frame is used after a transmission of frame that needs to be acknowledged by the receiver. This is used to signal the sender that it successfully received and checked the frame for errors by checking the frame control sequence.

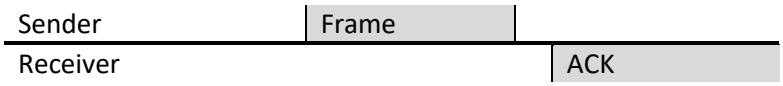


Figure 25: Ack only sequence of frames

6.3.1.1 Acknowledgement

After each transmission the receiver signals to the sender that a packet has successfully been received. This packet does not contain any additional fields as can be seen in Figure 26.

Bytes	2	2	6	4
Field	Frame Control	Duration ID	Receiver Address	FCS

Figure 26: Acknowledgement frame

6.3.2 Clear to self

The second sequence, depicted in Figure 27, is an extension to the previous. The client will first transmit a clear to send (CTS) message to announce a bigger transmission is coming. This allow the client to reserve transmission time so it can then transmit a bigger frame without worrying about collisions.

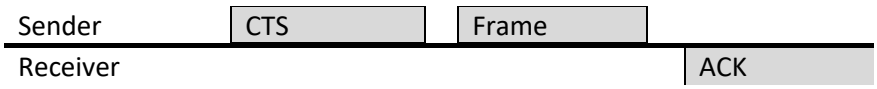


Figure 27: Clear to self sequence of frames

6.3.2.1 Clear to Send

The clear to send frame reserves the transmission medium for the receiver of the packet. As with the previous frame it does not contain any additional fields, as shown in Figure 28, other than the fields already discussed in chapter 6.2. This packet is either transmitted after a client sends a request to send (RTS) frame or if the client itself want to reserve the medium. The duration field is set either by taking the duration field of the request to send packet and then subtracting the time already passed. Or is calculated by adding the time to send this frame, the data frame and the acknowledgement frame together in case of the client sending the clear to send frame.

Byte	2	2	6	4
Field	Frame Control	Duration ID	Receiver Address	FCS

Figure 28: Clear to Send frame

6.3.3 Request to send

The third sequence, depicted in Figure 29, is again an extension to the previous sequence. As shown in Figure 29, the sender will ask for permission to send the frame, the receiver will then respond with a clear to send message and thus reserving the transmission medium for the sender.

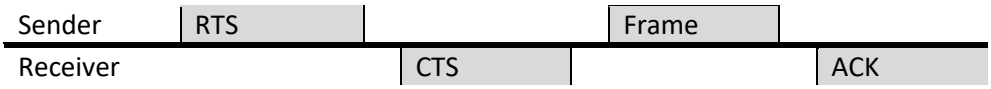


Figure 29: Request to send sequence of frames

6.3.3.1 Request to Send

As shown in Figure 30, the request to send frame does not contain any additional fields relative to the generic MAC frame. But there is a difference in meaning of the duration field: Instead of containing the NAV time for only this frame it will instead contain the time of the complete transmission, thus the expected time it takes to send the request, the station to grant permission to send data, the client to send the data and an acknowledgement of the station.

Bytes	2	2	6	6	4
Field	Frame Control	Duration ID	Receiver Address	Transmitter Address	FCS

Figure 30: Request to Send frame

6.3.4 Beacon

As depicted in Figure 31, an access point will repeatedly broadcast its beacons, the interval between transmissions is set by the beacon interval.

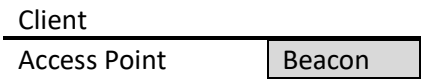


Figure 31: Beacon sequence of frames

6.3.4.1 Beacon

The beacon frame is a regular transmitted frame that announces the existence of the network. It is important to the network because it allows clients to quickly connect to the network and provides the clients with essential data about the network. As seen in Figure 32 and Figure 33, this frame contains four fields which are mandatory:

- Timestamp, used to synchronise the clocks on all the devices. Connected devices will use this to synchronise their timers with that of the base station.
- Beacon interval, the interval in which beacons are broadcasted by the base station. Because the beacon frame sends useful information the clients will need to receive this information and thus need to be awake.
- Capability information, used to advertise the capabilities of the network. This is important because all stations need to support the capabilities of the network otherwise they will not be able to join the network.
- SSID, name of the network. Clients use this to identify with which network it wants to connect.

And five optional fields:

- FH parameter set, this element is included when frequency hopping is enabled on the network. This will give the client all the information to joining such network possible.
- DS parameter set, contains the channel the network is using.

- CF parameter set, contains information about the contention free period when contention free access is used in the network.
- IBSS parameter set, contains the announcement traffic indication map windows (ATIM). This indicates the time between ATIM frames.
- Traffic indication map (TIM), used by clients that have enabled power saving features. These clients allow the base station to accumulate messages for them and thus allowing that client to sleep for periods of time. This field is then used to signal to these clients that data is available for them. This is therefore information that can be used by attackers to determine which clients the base station thinks are connected to the network.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	DA	SA	BSS ID	Seq Cont	Frame body	FCS

Figure 32: Beacon frame

Bytes	8	2	2	#	7	2	8	4	#
Field	Timestamp	Beacon Interval	Capability Info	SSID	FH Param set	DS Param Set	CF Param Set	IBSS Param Set	TIM

Figure 33: Beacon frame body

Beacons are only transmitted by the access points and with exception to one field (TIM) only transmits information about the network itself. Therefore is, except for the TIM field, tagged as not useful for tracking.

6.3.5 Probe

The client will start with sending a probe request to the access point which in turn will respond with a probe response (Figure 34).

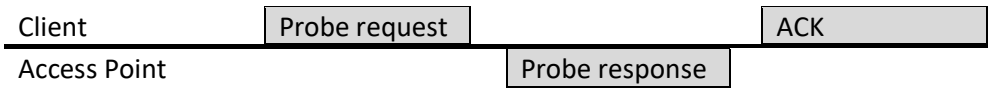


Figure 34: Probe sequence of frames

6.3.5.1 Probe Request

The probe request is used by clients to search for networks without waiting for the base stations to send beacons. It only contains two extra fields (depicted in Figure 36):

- Service set identifier (SSID), network name the client tries to connect with. Therefore it leaks information about to which network the client tries to connect. Thus allowing for the tracking of how much each network is used and how many clients are connected.
- Supported rates, defines which data rates the client supports. Since this is different for many types of devices it could be used for tracking of the client.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 35: Probe request frame

Bytes	#	#
Field	SSID	Supported rates

Figure 36: Probe request frame body

6.3.5.2 Probe Response

The probe response is the response of the base station when it receives a request with compatible parameters. It contains, as shown in Figure 38, nearly all the fields the beacon frame contains except for the TIM field as that is only useful for clients that are already connected. Although it did not say much in a beacon frame. In this case it gives information about the usage of a network since this is a response from a client that tries to discover a network. Therefore, instead of tagging the data as unusable for attackers it is now tagged as being able to be used to track the usage of a network.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 37: Probe response frame

Bytes	8	2	2	#	7	2	8	4
Field	Timestamp	Beacon Interval	Capability Info	SSID	FH Param set	DS Param Set	CF Param Set	IBSS Param Set

Figure 38: Probe response frame body

6.3.6 Authentication

A client will start the authentication process after it has detected a network that it wants to connect to. The client will try to authenticate with the AP by either using open authentication (Figure 39), which is used in an open network or with a network that uses WPA/WPA2 encryption.

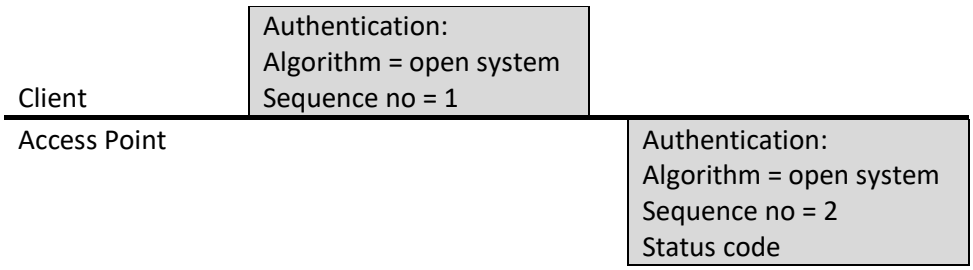


Figure 39: Open authentication sequence of frames

Or with shared key authentication in case of a network which uses WEP encryption (Figure 40).

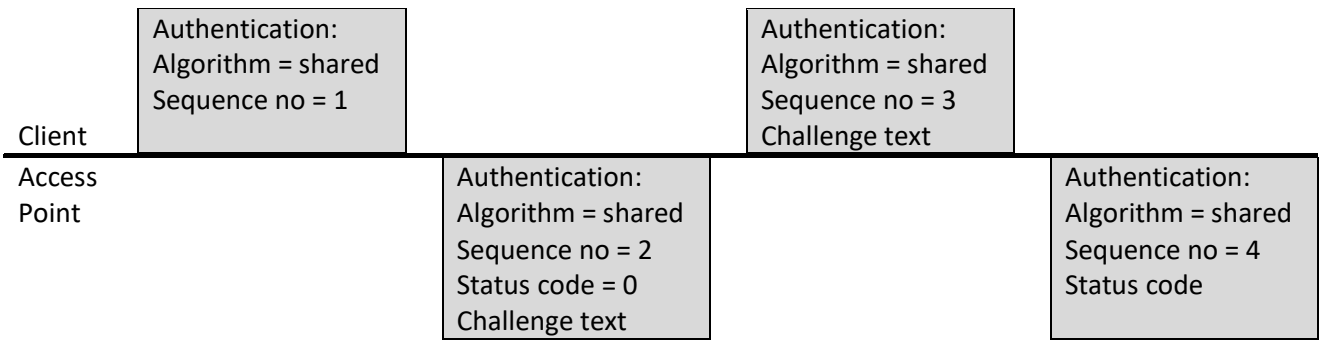


Figure 40: Shared key authentication sequence of frames

6.3.6.1 Authentication

The authentication frame is the only frame used in the authentication process and has 4 extra fields in it:

- Authentication algorithm number, used to identify which type of authentication is used. Currently only two types are defined: an open system and shared key authentication. The second one is used in shared key authentication which is mainly used in a WEP encrypted connection. WPA2 uses the open system authentication but then encapsulates the key exchange in the 802.11 data packets. Because of this it will first authenticate and associate openly and then upgrade the connection to a WPA2 encrypted variant.
- Authentication transaction sequence number, used to track progress of the authentication process. This is due to the authentication process being a multistep process.
- Status code, indicates if the authentication has succeeded or not and if it did not it contains the code with a reason why it did not.
- Challenge text, filled with data depending on which type of authentication algorithm is used.

As discussed above in the encryption bit section, only a limited amount of possibilities for these variables are possible since a good encryption is required. Therefore, all these fields will be tagged as unusable.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 41: Authentication frame

Bytes	2	2	2	#
Field	Authentication Algorithm Number	Authentication Transaction Sequence no.	Status code	Challenge text

Figure 42: Authentication frame body

6.3.7 Association

After the client authenticated with the network it will then try to associate with the access points to complete the joining of the network (Figure 43).

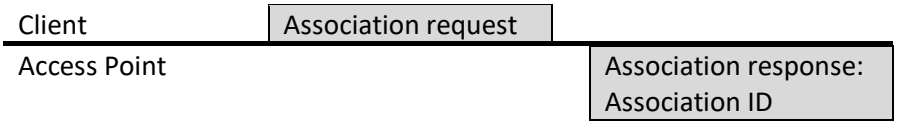


Figure 43: Association sequence of frames

6.3.7.1 Association Request

The first step in joining the network is for the client to send an association request. This request consists of several fields, depicted in Figure 45, which are used in the process:

- Capability information, used to advertise the capabilities of the client. The client needs to support the capabilities of the network to be able to join. These capabilities give an adversary a lot of useful data which could be used to track when a client joins the network.
- Listen interval, used in case a device uses power saving measures. It indicates to the base station how many beacon periods the device wants to stay asleep, the access point broadcasts these beacon frames periodically thus this is a measure of how long the device can stay asleep. This will determine in turn how much data an access point has to store before it can signal to a client to retrieve the data. Therefore the access point has to take this into account when accepting or rejecting a new client. The listen interval might also be used by an attacker to track clients.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 44: Association request frame

Bytes	2	2	#	#
Field	Capability Information	Listen Interval	SSID	Supported Rates

Figure 45: Association request frame body

6.3.7.2 Re-association Request

The re-association request is nearly identical to the association request with one difference which can be seen in Figure 47:

Current AP address, indicates the address of the base station which it was previously associated with. When a client connects to a different base station in the same network this is used to transfer the association and buffered frames to the new base station. Unfortunately, this makes tracking usage of a network possible and it could also be used to track the direction a client is move in, in a bigger network.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 46: Re-association request frame

Bytes	2	2	6	#	#
Field	Capability Information	Listen Interval	Current AP address	SSID	Supported Rates

Figure 47: Re-association request frame body

6.3.7.3 Association Response and Re-association Response

After the client has requested to associate with the network by sending either an association request or a re-association request the access point will respond with the association response. This response contains multiple fields, as shown in Figure 49, but only one that has not been discussed before:

- Association ID, a unique ID given to the client after said client has been associated with network. Especially used when clients use power management features of the network. As it is unique to the client it can be used to track the client.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 48: Association response frame

Bytes	2	2	2	#
Field	Capability Information	Status code	Association ID	Supported Rates

Figure 49: Association response frame body

6.3.7.4 Disassociation and De-authentication

These frames are used to disassociate and de-authenticate a client from the network, the frame contains only one extra field:

- Reason code, code that explains why the client was disassociated or de-authenticated from the network.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 50: Disassociation frame

Bytes	2
Field	Reason code

Figure 51: Disassociation frame body

6.3.8 EAP key exchange

To connect to a network with WPA2 protection some extra steps need to be taken as depicted in Figure 52. After association the access point will initiate the extensible authentication protocol over wireless (EAPOW) key exchange. This is done in the following four steps:

- The access point will start by creating a nonce(aNonce) and transmit that to the client
- The client will then also nonce (sNonce) and use both these nonces to create the Pairwise Transient Key (PTK). The client will then derive the various keys depending on which encryption method is used. In the case of WPA2-CCMP(more about this in the chapter 6.3.9.2) the following keys are derived from the PTK:
    - Key Encryption Key (KEK), used to encrypt EAPOW data.
    - Key Confirmation Key (KCK), used to generate the MIC field.
    - Temporal Key (TK), session key, used as a key to encrypt data.

  The client then uses the KCK to generate the Message Integrity Code (MIC). The MIC together with the sNonce will then be transmitted back the access point.
- The access point is now able to also generate the PTK from both nonces. It then uses the derived KCK to check the validity of the message by calculating the MIC and comparing it to the received MIC. After the message has been verified the client will then send another key message. This message sends the Group Temporal Key (GTK), a key that is used to derive keys to encrypt broadcast/unicast messages, encrypted by the KEK key. Furthermore, another MIC is generated using the KCK to ensure integrity.
- The client will then start checking the integrity of the message by checking the MIC and if correct will save the GTK for later use. It then generates the last message which is a conformation message to the access point that the association has been successfully concluded. This message will also include a MIC to ensure integrity.
- In the last step, the access point will check the MIC and if correct the client will then be successfully associated to the network

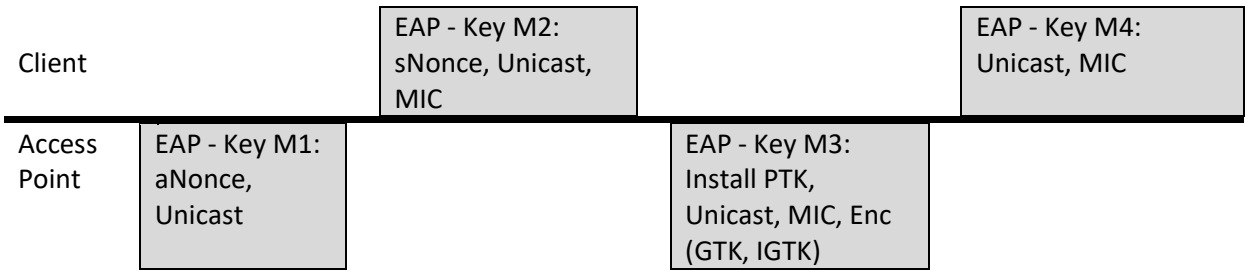


Figure 52: EAP key exchange sequence of frames

6.3.8.1 EAP

EAPOW is used to provide WPA2 protection to the network. EAP itself has several frames but the basic frame format is depicted in Figure 54 and adds some extra data fields but also has overlap with the MAC generic frame (not shown in the figure):

- Ethernet type, as with the generic MAC type this indicates what kind of packet type it is. The value for EAPOW is fixed and thus no data can be derived from this field.
- Version, only one version is standardized thus cannot be used.
- Packet type, there are various types of EAPOL packets. The same reason as with the packet type defined in the generic MAC is used therefore it is required.
- Packet body length, describes the length of the frame body.
- Packet body, depends on the packet type.

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame header	FCS

Figure 53: EAP frame

Bytes	2	1	1	2	#
Field	Ethernet Type	Version	Packet type	Packet body Length	Packet body

Figure 54: EAP frame header

6.3.8.2 EAP – key

The most important frame that is used in WPA2 sessions is the EAP key frame. This frame is used to authenticate the client and to create session keys and transmit the global keys. The frame introduces the following fields (also shown in Figure 56):

Bytes	2	2	6	6	6	2	0-#	4
Field	Frame Control	Duration ID	Destination Address	Source Address	BSSID	Seq Cont	Frame body	FCS

Figure 55: EAP-Key frame

Bytes	1	2	2	8	32	16	8	8	16	2	#
Field	Desc. type	Key info	Key length	Replay counter	Key nonce	Key IV	Key Sequence Start	Key identifier	Key MIC	Key data length	Key data

Figure 56: EAP-Key frame body

- Descriptor type, used to identify which type of WPA variant is used. This is network dependent thus the argument can be made that this makes tracking of network usage possible.
- Key information, a field that is subdivided into five subfields, shown in Figure 57, consisting of the following:
    - Control bits, are set depending on handshake stages, and thus could be used to track the progress of the handshake.
    - Key ID, a two-bit value that is used to determine which key group needs to be used.
    - Key type, used to distinguish between different types of key messages. Can only be used to determine what kind of key messages are transmitted.
    - Key version, indicates which encryption schemes are used, since this might be different per network thus it might indicate network activity.

Total	4				6						2		1	3		
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved				Control bits						Key ID		Key type	Key version		

Figure 57: Key information field

- Key length, determines the length of the resulting key that is used. This might be network specific setting thus it could be used to determine the activity of a network.
- Replay counter, a value that is incremented with each message thus this might be used to track clients.
- Key nonce, a nonce which is used to derive the temporal keys from, should be randomly generated value and thus not usable.
- Key IV, also a random value, thus not usable,
- Key sequence start, a sequence number that is expected in the first frame after the keys are installed. Used to prevent replay attacks but could be used by attackers to links the next packet thus could be used to track clients.
- Key identifier, currently not in use but could be used when the client uses multiple keys to determine which key needs to be used. Though not in use today, if it were used it could be used to track clients.
- Key MIC, message integrity check value, calculated by taking multiple fields from this packet thus not usable.
- Key data length, might again be determined by the network thus could be used to track network activity.
- Key data, data that depending on the message type and handshake stage might contain either encrypted data, thus not usable.

6.3.9 Data transmission

The next sequence encompasses transmission of data. This is the reason for many clients to connect to a network: to send and receive data. The client has some possibilities to transmit data. Like depicted in Figure 58, it can simply transmit data and the receiver could then acknowledge if the message was received correctly or use clear to self or request to send to reserve the transmission medium and then transmit the data.

Figure 58: Data transmission sequence of frames

6.3.9.1 Encryption

Depending on the settings of the network some form of encryption might be used to protect eavesdropping of user data. The three most well-known protocols that provide this kind of protection are: wired equivalent privacy (WEP), temporal key integrity protocol (TKIP) and counter mode cipher block chaining message authentication code protocol (CCMP). WEP was the first but had some flaws which made it possible for attackers to crack the keys and thus the communication. The successor of WEP was TKIP, which essentially uses the same encryption algorithms to stay compatible but would remove the flaws which made cracking WEP possible. Since it was still using a weak encrypting algorithm another successor was developed: CCMP. This uses advanced encryption standard (AES) encryption instead of the older and weaker rivest Cipher 4 (RC4) encryption. Although all methods could be explained in depth only CCMP will be further looked into.

6.3.9.2 CCMP

CCMP encrypts the frame body of a frame. To achieve this some extra data is required and an extra check is implemented to provide tamper proving. The extra data is pre- and appended to the frame body, as shown in Figure 59.

Bytes	8	#	8
Field	CCMP header	Frame body	MIC

Figure 59: CCMP frame format

The frame body and MIC are both encrypted and thus are not usable any more for an attacker.

The header consists of two subfields as shown in Figure 60:

Bits	8	8	5	1	2	8	8	8	8
Field	PN0	PN1	Reserved	Reserved	Key ID	PN2	PN3	PN4	PN5

Figure 60: CCMP header

- Packet number (PN0-5), is a unique identifier of the frame and is incremented every transmission. This is to protect from injection and replay attacks. Since it is an incrementing number it could be used to track clients.
- Key ID, is the index of the key that is used, as this might also change per client the might also be used to track clients. Though the usage of this field is fairly limited due to the field only being two bits in length thus only four different options will be available.

6.3.10 PS-Poll

And the last of the control frames is the PS-Poll frame. As shown in Figure 61 the client sends a PS-Poll frame to the access point to signal it wants to receive data that the access point has buffered. The access point will then respond with a data frame it has stored for the client.

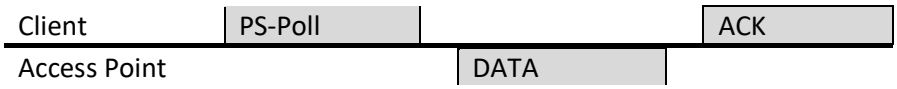


Figure 61: PS-Poll sequence of frames

6.3.10.1 Power-Save Poll (PS-Poll)

The last of the control frames is the PS-poll frame which can be used to allow clients to use power saving features. There is only one different field in this frame and that is the association ID (AID) instead of the duration field (shown in Figure 62). This AID is assigned to a client after it is associated with the base station. The client sends this ID to the station to request the frames the base station has buffered for the client.

Bytes	2	2	6	6	4
Field	Frame Control	AID	BSS ID	Transmitter Address	FCS

Figure 62: Power-save Poll frame

7 Solution

Now that the problems have been determined, a solution needs to be though out and tested. In this case the solution needs to prevent the attackers from reading all the information that is tagged as important in the previous chapter. The choice was made to encrypt these fields in such a way that the attacker could not distil any information from the packet that has been transmitted. This would also include information that is retransmitted, thus this should not look similar to the previous transmission as this would still allow the attackers to possibly track clients. Before going into details about the solution some consideration could already be made, which is done in the next chapter.

7.1 Consideration

The considerations talked about in this chapter could already be made because of some requirements that are already known:

- Adversaries should not be able to read certain fields; therefore, an encryption method is going to be used.
- To encrypt these fields, keys need to be known by both the base station and the client. However, keys used by WPA2 are not available yet, as these keys are required before these keys are generated. Therefore, a separate key exchange is required to provide these new keys.
- The identifier (MAC addresses), that is used to determine the destination, is going to be encrypted and should be nondeterministic for adversaries. Therefore, special care needs to be taken to make sure clients will still be able to determine easily and without much impact in speed and/or battery life that these packets are addressed to them.

7.1.1 Key exchange

The first hurdle in the solution is the fact that messages have to be encrypted before the client has been authenticated, not allowing the use of the WPA2 session keys for encryption. To solve this, a key has to be exchanged between the client and the base station. A key exchange method has to be chosen which is most suitable for this situation. Two different methods were compared: Elliptic curve Diffie-Hellman (ECDH) key exchange and RSA key exchange. First, some criteria are defined that are important for this solution and then a comparison is made between the two using the following criteria:

- Key size, the most important, as the amount of bytes that can be added to messages is limited and has a direct impact on performance: longer keys means less bandwidth available for other data.
- Speed, also important because the protocol also has to work on devices with less computing power.

For the first criterion we look at “Elliptic Curve Cryptography (ECC) Certificates Performance Analysis” (Ajay Kumar et al., 2013), as seen in Figure 63 the difference is that the RSA key is between 6 and 30 times bigger than the ECDH key.

Symmetric Key Size (bits)	RSA, DSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	**3072**	**256**
192	7680	384
256	15360	521

Figure 63: Comparison between key sizes (Ajay Kumar et al. 2013, table 4)

For the performance we will also look at "Performance Evaluation of Public-Key Cryptosystem Operations in WTLS Protocol"(Levi and Savas, 2003) which gives us the graph in Figure 64. This shows that the speed depends mainly on two factors: key size (bits) and which curve is used. What can be noted is that the performance between RSA-1024 and two of the three curves at 160 bit (160P en 163k) are comparable but when the key size is increased RSA becomes slower in comparison to the elliptic curves.

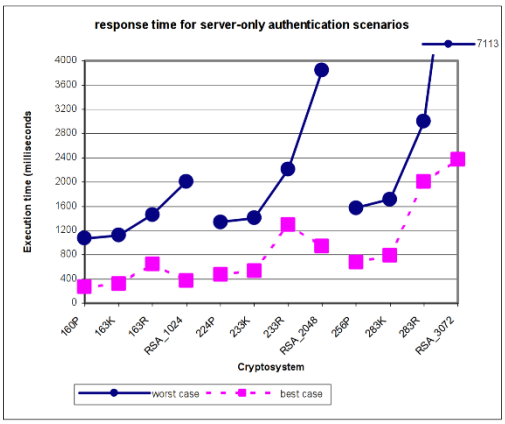


Figure 64: Performance comparison between RSA en ECDH (Levi and Savas 2003, figure A)

Therefore, the choice was made to use ECDH key exchange with the 256P curve. This curve gives better performance than its RSA counterpart (3072 bit RSA) and is also less than a 10^{th} of the size. The choice for 256 bits elliptic curve key was made because it is equivalent in strength to a 128 bits symmetric key which is recommended by "Référentiel Général de Sécurité" (Agence nationale de la sécurité and des systèmes d'information, 2014) and others(example: OWAPS, NIST) for usage after 2020.

7.1.2 Encryption

An encryption standard needs to be chosen which is both safe to use and readily available on the platforms that would want to use this system. The choice was made to make use of encryption methods that are already available to these systems. To find which methods are available to these systems we looked at what kind of encryption methods are currently used to encrypt the content of the data packets. As discussed in the data transmission chapter only two methods of encryption are currently defined in the 802.11 standard: RC4 and AES. And also described in this chapter RC4 should not be used any more thus we are left with only one method: AES.

The next problem lies within how the encryption method is used to encrypt the data, the following encryption modes were considered when making this choice:

- ECB, Electronic CodeBook mode, the data that needs to be encrypted is divided into in to blocks of a certain size and then each block is encrypted separately with the key. In this case this method has two disadvantages. The first is that it can only encrypt a block at a time. The data needs to be made fit into that block size, which is hard because the data cannot be padded in any way or a small enough block size needs to be chosen that would fit the situation which might decrease the effectiveness of the encryption, due to making the block size smaller. The other problem is the fact that the output is linked to the input. Meaning that if a MAC address is encrypted for the second time it will output the identical output data as the first time (deterministic encryption). Thus, an attacker would not know what the MAC address of the client would be but it would still be able to track clients because the MAC addresses would still not change over time.
- CBC, Cipher Block Chaining mode, again the data is divided into blocks of a certain size but instead of encrypting each block separately a chain of encryptions is made. This is done by mixing (XORing) the output of the first block with the input of the second block. To make the output of the encryption unique an Initialization Vector (IV) is used for the first block, instead

of the non-existent previous block output. This system solves the second the problem of ECB mode; deterministic encryption, but does not solve the first; block mode operation.

- CFB, Cipher FeedBack mode, is the first of the streaming cipher modes, this means that the data is encrypted as a stream of bits and thus does not have to be padded to a certain size. This is achieved by encrypting the IV and the key, this creates a block of encrypted data. The input is then combined (XORed) with the encrypted data to create the output. If the input is smaller than a block of encrypted data, only part of the encrypted data will be used (the size of the input) to combine it to make the output. If the input is longer than one block another block will be created by using the encrypted data as IV and then create another block with the new IV and the key. The only disadvantage to this is that when the encryption is used to encrypt longer inputs the encrypted data that is generated is dependent on output of previous blocks.
- OFB, Output FeedBack mode, nearly the same as the previous implementation but instead of using the output as IV for the next block it uses the output of the encryption as the IV for the new block. Thus, the IV is not dependent on the input data any more. However, it is still dependent on the previous block. This might become inefficient when a part of the packet needs to be decrypted that is not in the first block. Because then it would first have to encrypt all previous blocks before it could start with decrypting the actual wanted data.
- CTR, Counter mode, also a variant of the previous two but now the input is completely independent of previous encrypted blocks. This is done by combining the IV with a counter. This counter is then increased for every successive encryption of the block. This then solves the disadvantage of the previous mode because the counter could easily be calculated.

Therefore, AES in CTR mode was chosen as method of encrypting all the required data.

7.1.3 IV generation

AES in CTR mode requires an IV to make the output of encryption non-deterministic. To determine how this IV needs to be generated the requirements of the IV were looked at. To determine these requirements, the RFC for AES-CTR was looked at which provides the following information:

“AES-CTR requires the encryptor to generate a unique per-packet value, and communicate this value to the decryptor. This specification calls this per-packet value an initialization vector (IV). The same IV and key combination MUST NOT be used more than once. The encryptor can generate the IV in any manner that ensures uniqueness.” (Housley, 2004)

As stated in the quote the IV absolutely needs to be a unique value, thus a random value is not good enough as it might randomly repeat itself. Furthermore, it also states that the IV needs to be communicated to the receiver. This is a problem for this system because data cannot simply be added to many of the packets without breaking the protocol. However, if this is interpreted in another way that is more in the spirit of the text, a work around could be found: the decryptor needs to know the IV. This change allows the use of pre shared and locally computed IVs allowing other methods of generating the IV. The devised method would start with a random IV generated by the client, which would then be shared with the AP in a frame that allows this. Then the IV is stored on both client and AP side, thus removing the necessity of communicating it every packet. To make the IV unique the choice was made to hash the IV every time it is used and store it as the new IV.

7.1.4 Hashing

The next step is selecting a hashing function; luckily, in this case the requirements are a bit lower than for a hashing algorithm than that is used for hashing passwords for example. This is because it does not matter that an attacker can determine what the IV is since it was transmitted over in plaintext anyway. There are however two other requirements: The first one is size of the resulting hash. As it is used as an IV for encryption, the output of the hash has to be the same or larger than the IV size. Since the choice was made to use AES, which has the following block sizes: 128, 192 and 256, the output of the hashing algorithm needs to be at least 128 bits long. Furthermore, the selection was limited to hashing algorithms that are already used one way or another as a hashing algorithm in the one of the 802.11 specification and thus are more likely to be supported by the devices that will implement the new protocol. The following common algorithms meet these requirements: MD5 (EAP-MD5), SHA-1 (WPA2 key management), SHA-256 (802.11w key management). To select which algorithm to use, the uniqueness of the output is looked at. To determine these two factors were looked at. The first is how many bits the output is long, more bits of output means less chance of a non-unique output. The second is collision attack rates; this does not give much information about accidental collisions but it could give an insight about possible problems with the algorithm. For the first SHA-256 is the clear winner here as its output is 256 bits long instead of the 160 bits for SHA1 and 128 bits for MD5. As for the second both MD5(“On Collisions for MD5 - M.M.J. Stevens.pdf,” n.d.) and SHA1 (Computer Security Division, 2006) have known collision attacks and thus SHA256 once more comes on top in this case.

Therefore the choice was made to use SHA256 as hashing algorithm.

7.1.5 Packet loss/energy consumption

The idea is to encrypt everything that can identify a client, which in turn will also encrypt the fields that are required to determine the destination of a packet. Therefore, some special considerations had to be made. This is due to many client being mobile devices with limited processing power and battery capacity. A client that would decrypt every message to check if it were addressed to it would use unnecessary encryption cycles that will take processing time and power. Furthermore, there is also a problem with packet loss, whenever a packet is not received there might be a possibility that the sender would hash the IV another time and thus desynchronising the client and the AP. Both these problems were solved by pre-encrypting a certain amount of addresses ahead. This would allow clients to more easily discard message because instead of encrypting messages it would only have to check whether a MAC exists within a pre-made list. Furthermore, creating a list of encrypted MACs would lower the possibility of desynchronization by packet loss because the client would listen for X amount of encrypted MACs ahead and behind. However, the effectiveness of this list will depend on the size of the list but this has to be determined at implementation, as it will also be determined by the resources available on the device.

7.2 Protocol changes

In the following chapter, the changes that were made are shown in flowcharts and are further explained. Because the choice was made to make changes universal for both the client and the access station some colour coding was added. For example for actions taken mainly by a client the colour for these actions were painted blue, others that are mainly actions done by the base station the colour orange was chosen and actions that are used by both are painted grey.

First, the data structures that are used by the new functionality is explained. After which the location where these changes will be added is discussed. Finally, the handling of various packets is explained.

7.2.1 Data structures

As described in the consideration, some data has to be stored for this new protocol to work properly. In total five different structures will be used and how they are shown in the other figures can be seen in Figure 65.

Figure 65: Used data structures

The following data is put into these data structures:

- WiFi config contains whether the new protocol has to be used or not and contains the GTK keys which are required for sending multicast messages.
- Client data contains the current private key which is used for generating the secret keys.
- Network data list, this is a list where the public key of each base station is stored when it is received in a beacon packet.
- Connection data stores the generated secret key, the current random IV and a list of encrypted MAC addresses generated with the secret key and the device MAC.
- MAC list is a list with all the encrypted MAC addresses and corresponding connection data, this list will be searched to check if the packet is addressed to this device.

The management of the encrypted mac list and the associated connections is managed in the two functions below. The first as depicted in Figure 66 is the function that creates the initial list of encrypted MAC for the connection. As can be seen in the figure the MAC is encrypted with the IV and the secret key. The IV and eMAC are then stored in the encrypted MAC list, the eMAC is also stored in the connection for later use. Then a new IV is generated by hashing the previous IV. This is done X amount of times (list size).

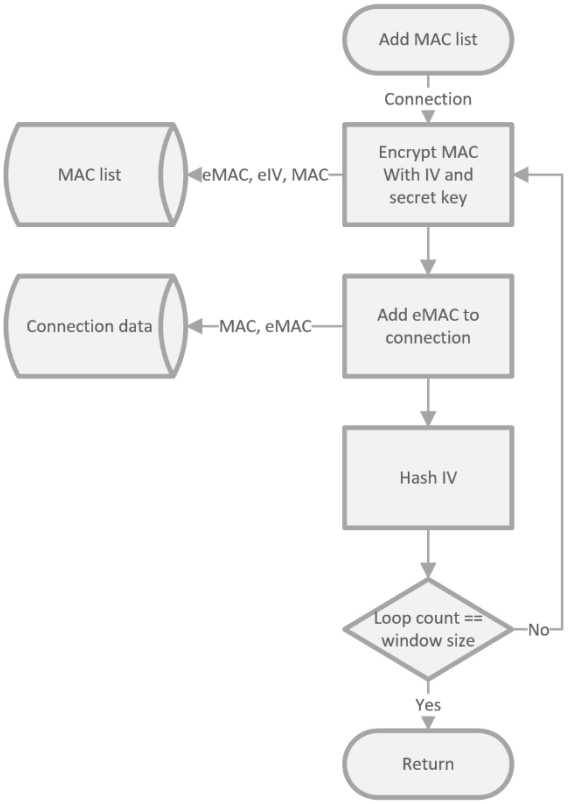


Figure 66: Adding MACs to the encrypted MAC list

The other function is depicted in Figure 67. It shows how this list updated. The last received eMAC sets the new center for the MAC list. Then the elements that fall outside the list are removed from both the connection data MAC list and the global MAC list. Then the same method as the previous function is used to add elements to the end of the list. Note here that elements can only be added to the end of the list due to the use of one-way hash functions that calculate the IV. This should be taken into consideration when determining the list size; because whenever a packet is received, the list is pushed forward removing the possibility of frames laying outside of the list from being received correctly.

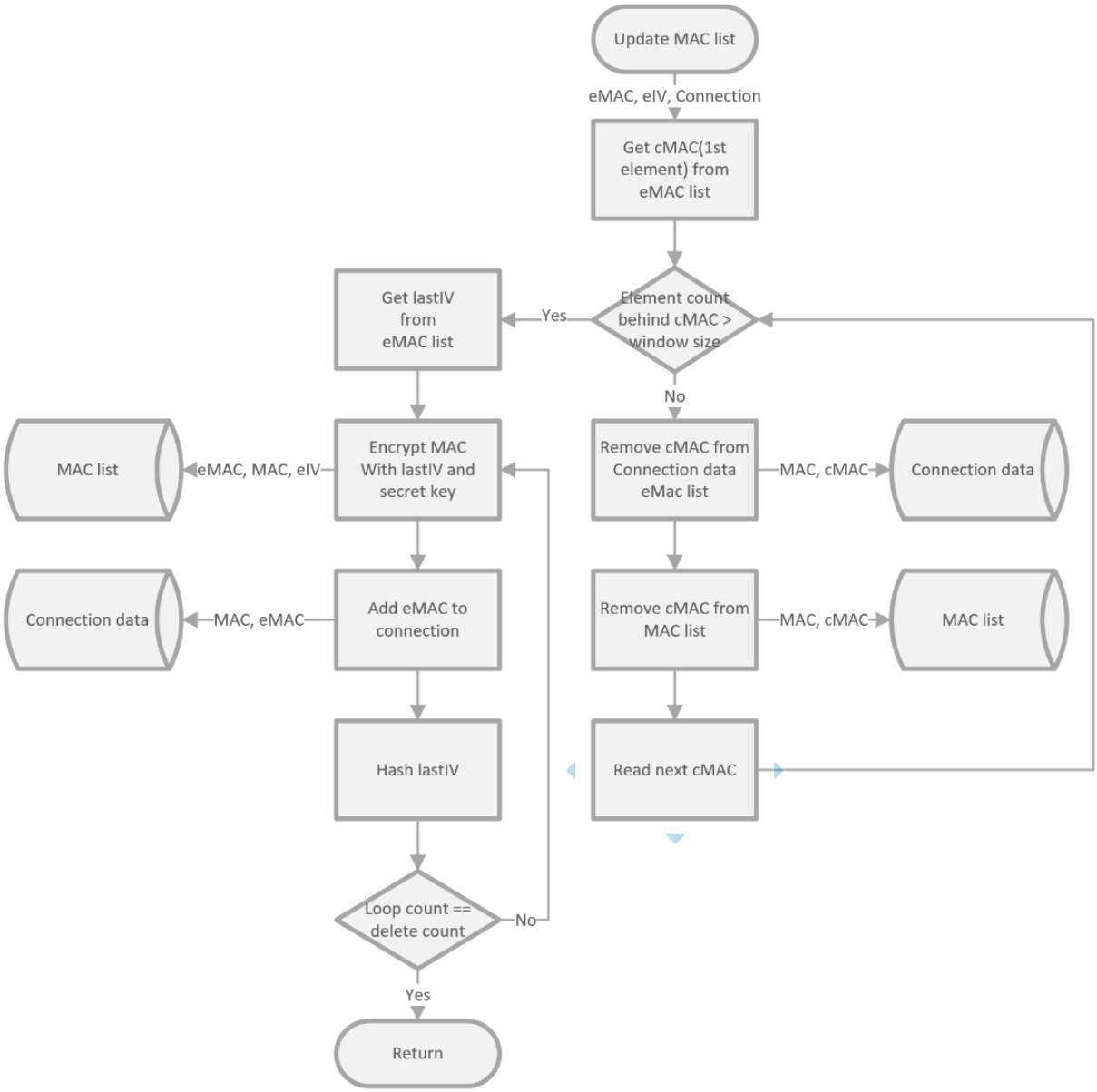


Figure 67: Updating encrypted MAC list

7.2.2 Determine entry point

For the system to work properly the changes have to be made in the correct location inside the chain of processes that are executed. When a packet is transmitted or received, it goes through a certain chain of processing. Putting the new solution in the correct place in this chain is very important as some fields are populated only in certain locations in the chain whilst fields are also verified in certain positions. Starting with the transmission, as seen in Figure 68, the user software determines that a packet has to be transmitted. Then some processing takes place (already in this software layer) and continues to the next step: encryption. The packets will be encrypted if required. The next step will be the new solution as from this point onwards the fields that need to be protected by the new system are fixed and thus the new system will then be able to encrypt these fields. The last step in handling the packets will be the calculation of the FCS. This cannot be done before the new solution because the solution changes the content of the packet and with that the FCS value.

On the receiving side, the packet is received by the hardware. Directly after the packet has been received the FCS is calculated over the received packet and is checked against the value in the packet. Then the new solution will decode all necessary fields into readable fields. The old implementation will then be able to further process and if necessary decrypt the packet.

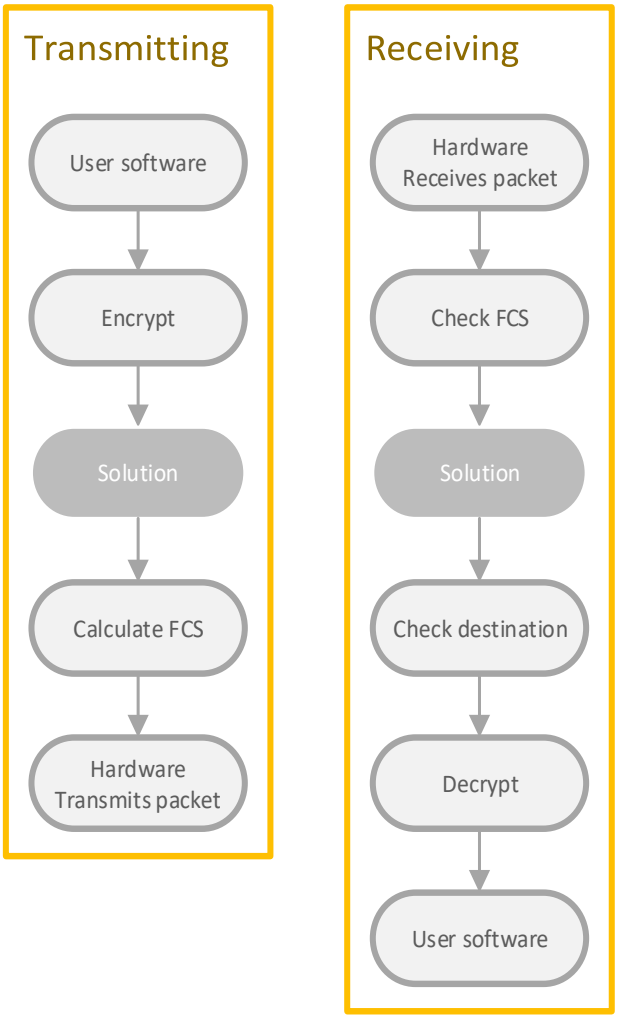


Figure 68: Location of changes

7.2.3 Beacon handling

The beacon packet is the starting point of whenever a client tries to join a network. This is because with the new protocol the client needs to know the public key of network it wants to join. This is done by adding the public key of the base station to the beacon packet as can be seen in Figure 69. This is the first function of the beacon but as described in chapter 6.3.4 there is another function that needed changes: Traffic indication map, this map indicates to clients that there is data available. Because this could leak information, the choice was made to encrypt this field as can be seen in the latter part of the flowchart. Each bit is encrypted using each client secret key and a hash of the current time (included in the beacon) as the initialization vector for encryption. The latter was chosen to minimize the overhead of updating the expected MAC list as a unique value was already available.

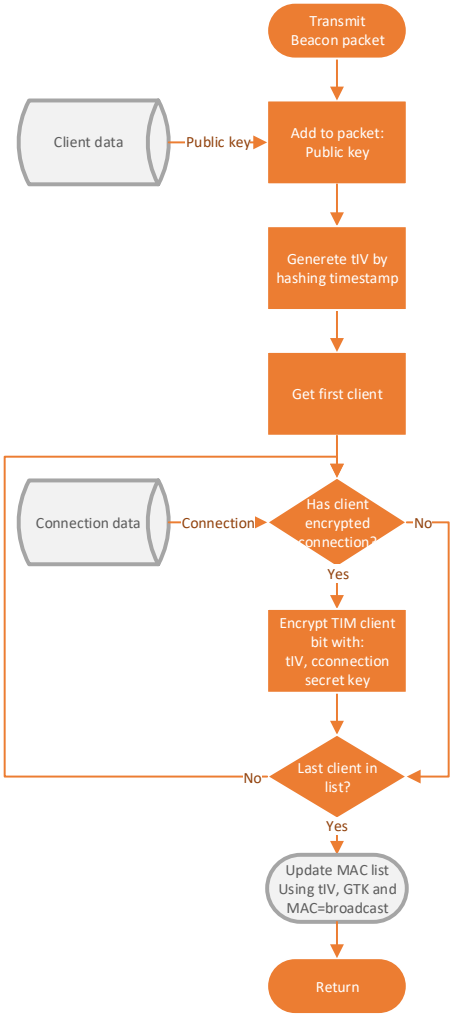


Figure 69: Beacon transmission handling

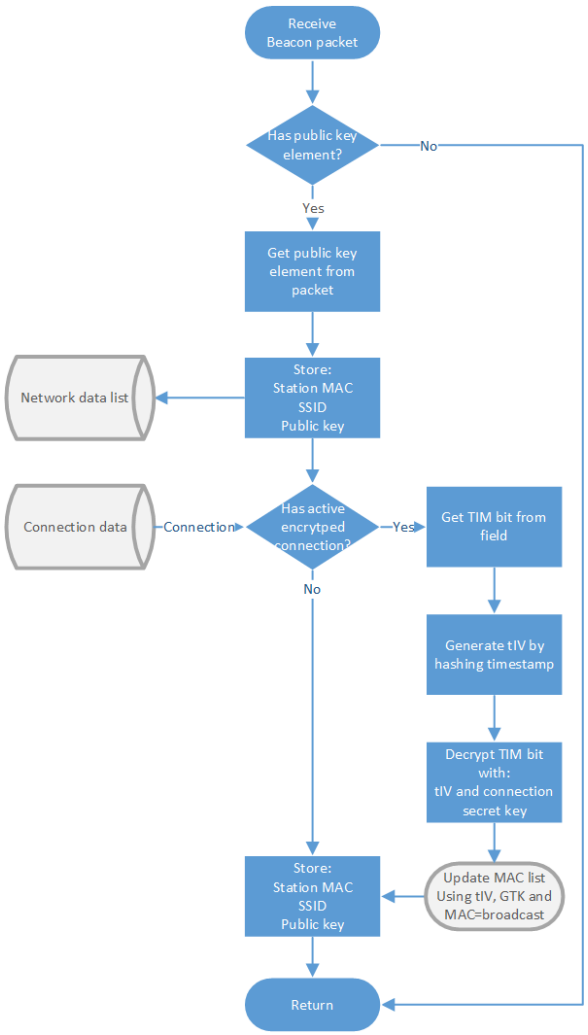


Figure 70: Beacon receive handling

On the receiving side, Figure 70, the client receives this beacon and checks whether the packet contains a public key. If so, it knows that this network uses the new solution and thus needs to store the public key for later use. Furthermore, it then checks if it is already connected with this network by check if connection data for this network is already known. If so, it then reads the TIM field and decrypts its bit in the field using the hashed timestamp as IV and the secret key corresponding to this connection. Lastly, it updates the MAC using the new IV generated by hashing the timestamp and the GTK. In this case, the GTK is used because it needs to update the MAC addresses for multicast message and thus it needs to be used as secret key shared by all clients that are connected: the GTK.

7.2.4 Probe and authentication handling

The probe request and authentication packet are handled identically, this is because either packet might be the first packet that is transmitted to the base station by the client. In both cases the settings are checked after which the network list is used to retrieve the network public key. If this is not available, there are three options left: either stop processing and let the packet be transmitted with the client information out in the open or null out all identifying data knowing that the receiving side will not be able to use any of the data or prevent the transmission of the packet all together. The first option leaks data while the second and the third one delays the making of the connection. The first option is not an option as it will leak information. The second option is less optimal as the third as it will transmit useless information. The third also has a downside: the transmission has to be stopped. The impact of stopping the transmission cannot easily be determined, for example: what kind of signal is sent back to the transmitting layer above? An error? How does the program handle this? All these questions require a lot of research into the control mechanisms above, therefore the second option was chosen. Next, a new public and private key pair is generated, this has to be repeated every probe request/authentication packet otherwise the client might still be tracked using this public key. Then a secret key is created by combining the public key of the network that was stored in the network data list and the private key of the client. An IV was randomly generated by the client. Now the packet can be encrypted using both the secret key and the random IV. Both the public key and the IV are then added to the packet to make sure the receiver is able to decrypt the message. The last step is to create a connection and fill it with the secret key, IV and an empty list of encrypted MAC address that is then filled using the function defined in the chapter 7.2.1.

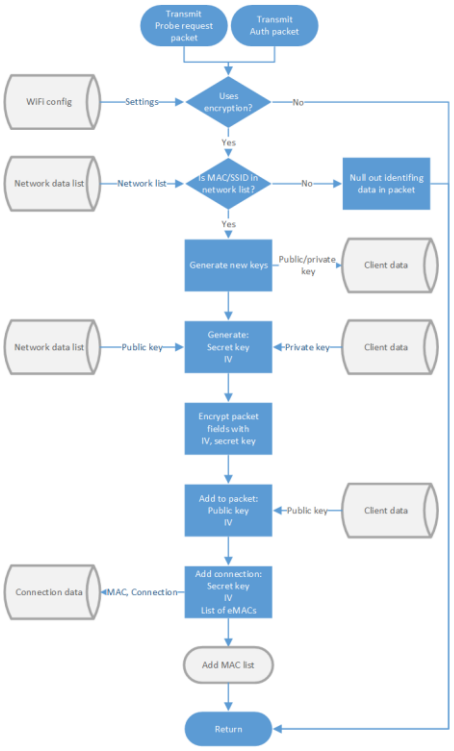


Figure 71: Probe and authentication transmission handling

On the receiving side, the base station starts by checking whether the new solution is enabled or not, if not it will bypass all new changes. Then a check is done to verify that the client also uses the new solution by checking if the client has included the IV and public key in the message, if not it knows that the client does not support the new solution and thus it shall also by pass the changes. The base station now knows its own private key, the client public key and the initial IV used for encryptions. The next step is to create the secret key using both its private key and the client public key. It is now able to decrypt all the fields in message by using the secret key and the IV included in the message. After decrypting the message, a check can be done on whether the message was actually intended for it by verifying the destination address. If so, it then saves this data into a connection data object for later use and adds future expected MAC addresses as described in the data structures chapter.

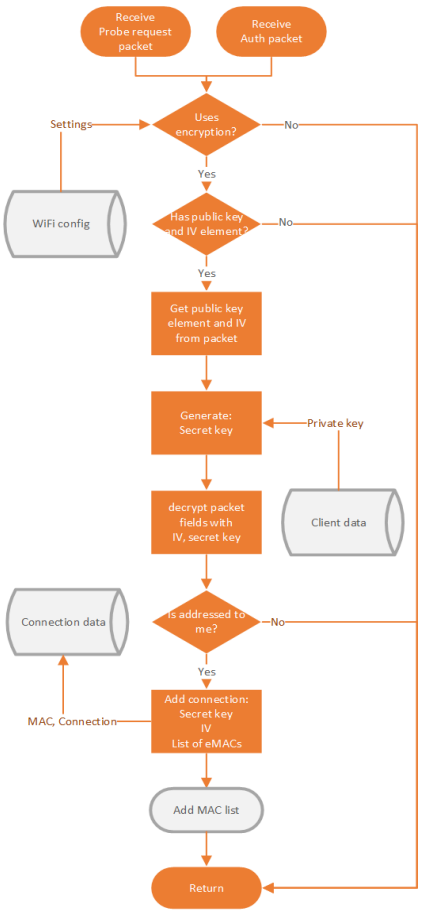


Figure 72:Probe and authentication receive handling

7.2.5 PS-Poll handling and other packets

The last handler is also combined due to their similarity: Power management handling has an addition compared to the normal packet handler.

Whenever a packet is transmitted which is not any of the previous discussed packets this handler, depicted in Figure 73, will be used on both the base station and the client. Whenever either the base station or client wants to transmit data it checks in the connection data list if the destination address is in it. In case of a base station, this list will consist of all its clients that are connected and the multicast addresses. For a client this list will contain the MAC address of the base station it is connected to and the multicast addresses. Whenever the destination is in the list it will then use the information contained in the list to encrypt the packet fields with the secret key and the current IV in use. This IV is then updated to the next IV by hashing the previous IV. Lastly, the MAC list is updated using the function previously discussed.

In case of it being a PS-Poll packet a single step is added before the packet is encrypted (marked green in Figure 74): the PS-Poll packet type is converted to an ordinary data packet and a tag is added to the (encrypted) data field that signals that is a PS-Poll packet.

On the receiving side, the destination is checked against the MAC list, if it is not in this list either the client is desynchronised or it is not addressed to the receiver. Therefore, it will skip further processing and highly like be ignored by the steps that com after this step. If it is in the list it will then retrieve the IV and corresponding (unencrypted) MAC address from this list. This MAC is then used to retrieve the secret key from the connection data list. The IV and secret key can then be used to decrypt the packet. The received IV is then set as the new current IV, the MAC list is then once more updated using the new current IV as centre point of the list.

In case of it being a data packet a step is added after updating the MAC list: it checks whether the tag that was added in case of a PS-Poll packet is in the packet. If so, it will then convert the data packet back into a PS-Poll packet.

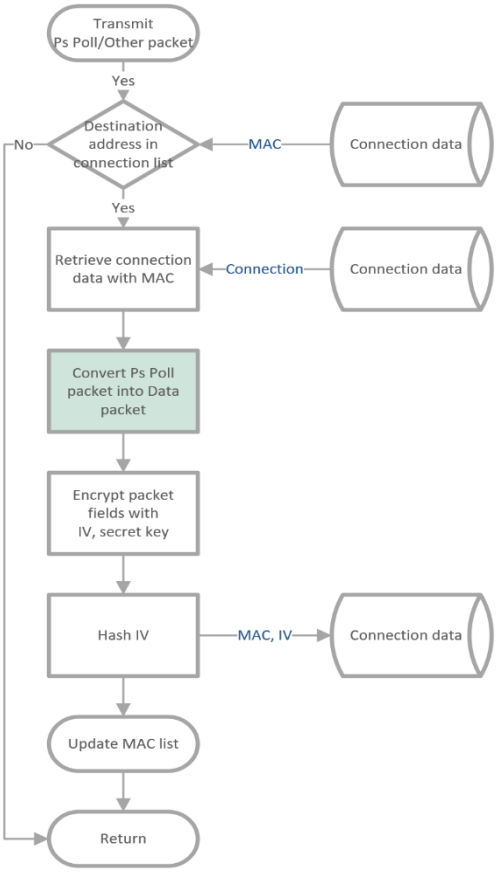


Figure 73: PS-Poll/Other transmission handling

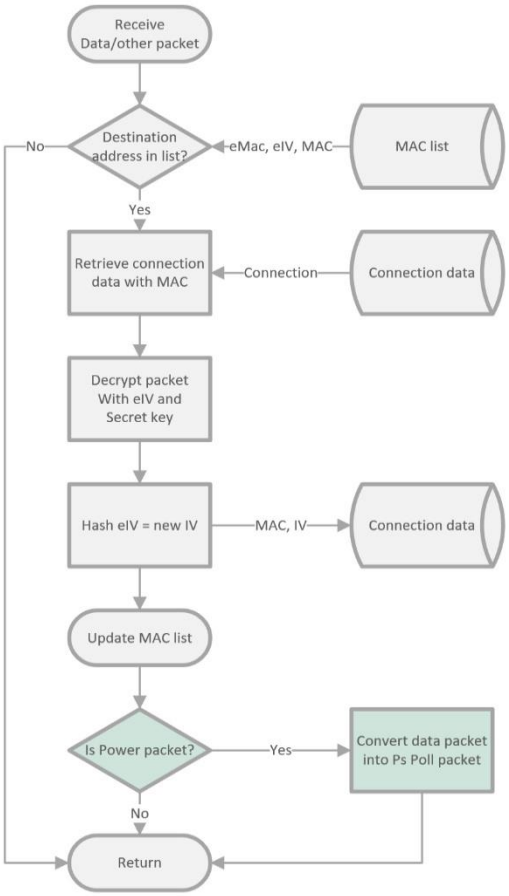


Figure 74: Data/Other packet receive handling

8 Proving solution

8.1 What is Proverif

Proverif is a cryptographic protocol verifier, it is able to analyse the security of these protocols. It is able to prove if the following properties are true for protocols:

- Secrecy
- Strong secrecy
- Authentication
- Equivalences between processes

It provides support for symmetric and asymmetric encryption, hashing and digital signatures. This allows the users to analyse their protocols and provide proof that the properties mentioned above hold for their protocol.

Proverif is not perfect as described in the user manual: “ProVerif is sound, but not complete.

> ProVerif’s ability to reason with reachability, correspondences, and observational equivalence is sound (sometimes called correct); that is, when ProVerif says that a property is satisfied, then the model really does guarantee that property. However, ProVerif is not complete; that is, ProVerif may not be capable of proving a property that holds. Sources of incompleteness are detailed in Section 6.3.4.”(Blanchet et al., 2016)

However, as described in the quote above whenever it does prove certain property is true for a given protocol it really guarantees that.

This guarantee is the most important reason to use Proverif and comes back to the overall research topic: how do you verify that the new protocol actually improves the problems in the old protocol? The problem in this case is the fact that the old protocol sends identifiable data over an open communication medium. In chapter 6, the problem is identified. Then a solution is though out to solve these problems and then the solution needs to be verified and later tested. In this case, we simply have a protocol and a list of fields that require secrecy. Implementing this in Proverif gives us a guarantee that if the secrecy properties hold up for all the fields that were discussed in chapter 6 then it will be a sound solution to the identified problem.

8.2 Implementation choices

As can be seen in the chapter about the solution it is general in certain areas. Thus, to verify whether the solution is working, the choice was made to implement the key sequences described in chapter 6.3 and shortly described below:

- Probes, clients periodically transmit probes to look for networks in their vicinity.
- Authentication and association, in a network, clients have to authenticate and associate before being able to use the network.
- WPA2 key exchange, in this sequence the keys that are used to encrypt the user data are negotiated.
- Data transmission, when a client transmits data.
- Data transmission with multiple receivers (unicast/multicast), a special case of data transmission is between two clients directly. As both do not know the session keys of either one a special mechanism is used to allow this. Therefore a special mechanism needs to be devised to allow this to work using the altered protocol.
- Beacons, beacons announce the presence of a network. A single field compromises the clients. This will have to be addressed and verified.
- Power management, each frame has data bits associated with power management and this leaks information about the topography of the network. This and the implications of this will be addressed and verified here.

8.3 Prove trackability

To prove the trackability of clients in the current protocol implementation the sequences above were implemented before they are changed. As an example, the probe sequence is used to show that the current protocol does leak information.

Converting this into Proverif code gives us the following for the probe request send by the client.

```
1. (* probe req *)
2. new clientMac; (* Mac address *)
3. new probeRates; (* Rates which the client support, need to at least match network *)
4. (* CLI->AP : probe request : da, sa, seqctl, body(SSID, rates) *)
5. out ( net, ( clientMac, apMac, seqctl, ( SSID, probeRates ) ) );
```

To verify that it is indeed leaking information some variables have to be tagged as variables that have to be kept secret. These variables(or fields in the protocol) were determined in the previous chapter. For the probe requests the following variables were tagged: apMac, clientMac, seqctl, SSID and probeRates. This equals to the following code for each variable:

```
1. query attacker : apMac.
```

As also seen in the code example all these variables are transmitted plaintext over the network and thus attackers will be able to see the content of these variables and thus are able to track the client.

Running Proverif results in the following truncated output:

```
The attacker has the message apMac_44_1490.
A trace has been found.
RESULT not attacker:apMac_44[] is false.
```

As seen in the output the attacker found the variable apMac and thus is able to derive whom the receiving end of the probe request is. The output shows that for each of the queries given, Proverif is able to find ways to derive the content of the variable.

8.4 Altering implementation

In the previous chapter the 802.11 protocol was implemented in Proverif to prove that the current protocol leaks important information. In the following chapter this implementation is then altered to solve this issue.

8.4.1 Probes

The first sequence is one where there might not be any connection with any base station yet and the client is looking for networks. In a normal situation as described in chapter 6.3.4, no beacon is present in the probe communication sequence between the AP and the client. However, with the new solution a public key from the network needs to be known before a client is able to transmit a probe requests. This public key can be retrieved either from a beacon transmitted by the AP or from earlier received beacons where it saved this data.

In this case, the assumption is made that no earlier beacons are known and thus a beacon had to be transmitted. As seen in the simplified example code below, the AP will first generate its public/private key pair (1, 2) and then add this to the beacon packet (4). In this case, the beacon is not transmitted over a public channel but over a private channel. This is done because the AP is leaking its own MAC addresses and other data here. As described in chapter 6.3.4, beacon fields, except for the TIM field, cannot be used for attackers. However, whenever a client transmits the MAC address (and other beacon data) of the access point it does leak information thus the data of the AP has to be tagged as secret. Therefore if the beacon was not transmitted over a private channel, Proverif would use the Beacon transmission to prove information was leaked.

```
AP
    1. APPrivKey = genPrivKey()
    2. APPubKey = createPubKey(APPrivKey)
    3. Beacon = new BeaconPacket(APMac, SSID, BeaconData)
    4. Beacon.elements.add(APPubKey)
    5. Send(private network, Beacon)
```

Next, the client receives a beacon (1) as shown in the code below, in the protocol this information would then be stored in a data structure for later use. In this case, this is not required as it is only a simulation of small part of the protocol. Thus, directly after receiving the beacon a probe request is transmitted (2 – 10). The first step in sending a probe requests is to generate its own public and private keys (2, 3) and generate a shared key from its private key and the base station public key (4). Then a random Initialisation Vector is generated (5), which is used with the shared key to encrypt all necessary data fields in the probe request (7). It then adds its own public key (8) and initialization vector (9) unencrypted to the message. The probe is then transmitted over the public network (10).

```
Client
    1. Receive(Beacon)
    2. ClientPrivKey = genPrivKey()
    3. ClientPubKey = genPubKey()
    4. SharedKey = genSharedKey(ClientPrivKey, Beacon.elements.APPubKey)
    5. IV = random()
    6. ProbeReq = new ProbeRequestPacket(ClientMac, APMac, SequenceCtl, SSID, ProbeRates)
    7. eProbeReq = encrypt(ProbeReq, IV, SharedKey)
    8. eProbeReq.elements.add(ClientPubKey)
    9. eProbeReq.elements.add(IV)
    10. Send(public network, eProbeReq)
```

The station then receives a probe request with a key inside (1), it then creates the shared key using its own private key and the public key obtained from the probe request body (2). It will then decrypt the whole message (3). It will then proceed to prepare a probe response transmission. The first step for the response is to hash the previous IV to generate a unique new one (4). Then the previously obtained shared key is used in combination with the new IV to encrypt the response message (6). Lastly, the response is transmitted over the public network to the client. When comparing this flow of actions to the suggested flow described in chapter 7 it can be noted that some actions are missing: storing of data and some checks. Like storing all the public keys transmitted in the beacons and checks on whether the client or AP actually is the receiver of the packet. These checks are not explicitly tested in the actual Proverif program but are instead enforced by telling Proverif that they need to be the same as previously defined. In this case the receiver MAC address is bound the AP Mac address, thus whenever they are different the program fails.

AP

1. Receive(eProbeReq)
2. SharedKey = genSharedKey(APPrivKey, eProbeReq.elements.ClientPubKey)
3. ProbeReq = decrypt(eProbeReq, eProbeReq.elements.IV, SharedKey)
4. IVResp = hash(IV)
5. ProbeResp = new ProbeResponsePacket(ClientMac, APMac, SequenceCtl, SSID, beaconData)
6. eProbeResp = encrypt(ProbeResp, IVResp, SharedKey)
7. Send(public network, eProbeResp)

The client now receives a probe response (1). To check if this message is destined for this client it first needs to create a new IV from the previous IV (2). It then tries to decrypting the whole message (5), as in the previous case, if the message is not address to the client the decryption will fail. Finally, to signal to the access point that it correctly received the message it will then prepare an encrypted acknowledgement packet (7, 8).

Client

1. Receive(eProbeResp)
2. IVResp = hash(IV)
3. ProbeResp = decrypt(eProbeResp, IVResp, SharedKey)
4. IVAck = hash(IVResp)
5. Ack = new AckPacket(APMac, SequenceCtl)
6. eAck = encrypt(Ack, IVAck, SharedKey)
7. Send(public network, eAck)

The last step is for the access point to receive an acknowledgement from the client (1) and decrypt the message (2, 3). As before, the program would fail if it were not able to decrypt the message.

AP

1. Receive(eAck)
2. IVAck = hash(IVResp)
3. ProbeAck = decrypt(eAck, IVAck, SharedKey)

To check whether the changes had the desired effect, some elements were tagged as secret. The elements were determined in the previous chapter but were simplified here. Instead, the choice was made to group the variables together where possible to simplify the program as much as possible. In the whole program, the following items were tagged: ClientMac, APMac, SequenceCtl, SSID, BeaconData, ProbeRates. This list would result in the following code snippet:

```
1. query attacker : clientMac.
2. query attacker : apMac.
3. query attacker : seqctl.
4. query attacker : SSID.
5. query attacker : beaconData.
6. query attacker : probeRates.
```

At first the beacon was also transmitted over the public network therefore, as expected, Proverif was able to get the APMac, SequenceCtl, SSID and BeaconData variables because these were included plaintext in the beacon packet. After removing the beacon packet from the scope of the attacker by sending it over a private network, it was able to continue and find the next problem:

The current key exchange was not enough to solve the problem: After 12 hours of running Proverif, it still was not able to find a definitive answer to whether the implemented protocol was safe or not. After some further analysis it was speculated that man in the middle attack might have been possible within this communication sequence, although no definitive answer could be found because the execution was stopped due to time constraints.

To test if this was the possible problem an extra step was introduced which would eliminate a possible Man-In-The-Middle (MiTM) attack by introducing an extra encryption step: It was chosen to encrypt the shared key with the Pre-Shared Key (PSK), which is usually used to connect to the network. This solved the problem of the possible man in the middle attack because the program would no complete within a reasonable amount of time and would give the following output:

```
RESULT not attacker:probeRates_55[!1 = v_1810] is true.
RESULT not attacker:beaconData_82[!1 = v_4717] is true.
RESULT not attacker:SSID_52[] is true.
RESULT not attacker:seqctl_84[!1 = v_13694] is true.
RESULT not attacker:apMac_83[!1 = v_18184] is true.
RESULT not attacker:clientMac_54[!1 = v_22674] is true.
```

This showed that the solution for the problem was found because the only item that was changed was the key that was used to encrypt all the traffic. Adding the extra encryption step did restrict the usage of this system, therefore the implications of adding this will be discussed in chapter 10.

8.4.2 Authentication & association

There is a lot of overlap with the previous sequence because the handling of probes is identical to the handling of the authentication step. Because of these similarities with the previous implementation, only the variables that are tagged will be discussed: ClientMac, APMac, SequenceCtl, SSID, capRateData, ListenInterval and AssocId. When the client tries to associate with the network it starts by transmitting its capability information, listen interval, SSID and supported rates. The capabilities information and the supported rate fields were joined together, though the listen interval was kept as a separate variable because this would only be transmitted once (from client to AP) whilst the supported rates and capabilities would later be communicated back from the AP to client. In this last exchange of data, the AP would also include the AssocId that is assigned to each client and thus would need to be evaluated separately. Running Proverif gives the following truncated output:

```
RESULT not attacker:assocId_101[!1 = v_1700] is true.
RESULT not attacker:listenInterval_62[!1 = v_4376] is true.
RESULT not attacker:capRateData_61[!1 = v_8492] is true.
RESULT not attacker:SSID_52[] is true.
RESULT not attacker:seqctl_60[!1 = v_16721] is true.
RESULT not attacker:apMac_53[] is true.
RESULT not attacker:clientMac_54[] is true.
```

As with the probe sequence the output shows that the implemented system prevents attackers from finding the contents of these variables and thus prevents attackers from following these clients.

8.4.3 WPA2 key exchange & Data transmission

The WPA2 key exchange and data transmission sequences are also grouped together due to its similarity: both are data transmissions but the content of the key exchange is filled with another packet, the key exchange packet.

The client should now be associated with the network and thus the assumption can be made that they share a shared key to use for further communication. Furthermore, they will also have a shared current IV that can be used to generate a new unique IV .This is important because the packets in these two sequences both use data packets to transmit data. As these packets can have a variable length of content it cannot be assumed that extra data can be added to the packet. This is due to the maximum packet size that is defined in the 802.11 protocol. As described in the previous chapters this is solved by hashing the IV and use this as a new IV, also described is that a list is kept to lower the chance of desynchronisation due to packets. In the test program no packets are lost and therefore there is no need for lists of IVs which simplifies the test program.

The client starts with hashing the IV previously used to create a new unique IV (1), it then creates a packet with all the necessary information (2). The content of the packet in this example is abstracted away because that is the differentiating factor between the WPA2 key exchange and a normal data transmission. The content for these packets is as follows:

- Encrypted data packet, in this case the body of the packet will consist of three fields: a key ID, a packet number and the encrypted data. The first two need to be encrypted as discussed in the previous chapter and thus are tagged as secret in the program.
- WPA2 key exchange, four different packets are exchanged between the access point and the client (key frame 1 to 4) and thus for each of these exchanges a variable is tagged as secret.

The packet is then encrypted with the key, which was generated in the key exchange, and the newly generated IV (3).

Client

1. IVData = hash(IV)
2. DataPacket = new DataPacket(ClientMac, APMac, SequenceCtl, DataContent)
3. eDataPacket = encrypt(DataPacket, IVData, SharedKey)
4. Send(eDataPacket)

Next is the receiver of the packet, in this case the AP, also hashes the IV (2) and then decrypts the packet (3). Note this is completely different from the actual solution as there is no list to check from whether the packet is destined for the AP. It then sends an encrypted response back to the client acknowledging that the packet was received (4-7).

AP

1. Receive(eDataPacket)
2. IVData = hash(IV)
3. DataPacket= decrypt(eDataPacket, IVData, SharedKey)
4. IVAck = hash(IVData)
5. Ack = new AckPacket(ClientMac, SequenceCtl)
6. eAck = encrypt(Ack, IVAck, SharedKey)
7. Send(eAck)

The last step is that the client receives an acknowledgement from the access point, it will then repeat the process of generating a new IV (2) and decrypting the message (3).

Client

1. Receive(eAck)
2. IVAck = hash(IVData)
3. AckPacket= decrypt(eAck, IVAck, SharedKey)

Output of the WPA2 key exchange sequence:

```
RESULT not attacker:M4Data_74[!1 = v_1979] is true.
RESULT not attacker:M3Data_125[!1 = v_8843] is true.
RESULT not attacker:M2Data_73[!1 = v_17386] is true.
RESULT not attacker:M1Data_124[!1 = v_25929] is true.
RESULT not attacker:seqctl_126[!1 = v_34472] is true.
RESULT not attacker:apMac_65[] is true.
RESULT not attacker:clientMac_66[] is true.
```

Output of data sequence:

```
RESULT not attacker:packetNumber_80[!1 = v_1079] is true.
RESULT not attacker:keyId_79[!1 = v_1248] is true.
RESULT not attacker:seqctl_78[!1 = v_2219] is true.
RESULT not attacker:apMac_65[] is true.
RESULT not attacker:clientMac_66[] is true.
```

As seen in the output all variables are safe and thus these sequences are secured against tracking.

8.4.4 Data transmission with multiple receivers (unicast/multicast)

In this case, the shared keys that are used for all other sequences are not usable. This is due to these keys only being known to each individual client and the connected AP. Luckily, WPA2 has had the same problem and thus already a solution for this problem: a group temporal key (GTK). This key is transmitted to each client by the AP in the WPA2 key exchange. To simplify the solution this same key will also be used in the new solution. This leaves one problem: there is no shared IV and an IV cannot be included in the packets due to reason described in chapter 6.3.9. Therefore another variable that is known by each client is used: time. The AP periodically sends out beacons with timestamps in them, the clients are expected to synchronise with this time and thus each client should have an accurate network time available. To allow multiple transmissions within the same timestamp, the same solution will be used as before: the timestamp is hashed after every use. This will use the same mechanism as what is used to track the IVs and encrypted MAC address for normal transmission but instead of updating this list after receiving a transmission it is also updated whenever it receives a new beacon.

As described above this solution is nearly identical to the data transmission solution. The only difference being that the shared key is the GTK and the initial IV is a timestamp received from a beacon. Though two clients are implemented in this solution to verify that it is also save whenever there are more than one receiver and transmitter.

The checks are the same as for the normal data packet but one extra mac address has been added, the second client. Proverif was once more executed which gave the following (truncated) output:

```
RESULT not attacker:packetNumber_78[!1 = v_1149] is true.
RESULT not attacker:keyId_77[!1 = v_1513] is true.
RESULT not attacker:seqctl_124[!1 = v_2710] is true.
RESULT not attacker:apMac_65[] is true.
RESULT not attacker:client2Mac_67[] is true.
RESULT not attacker:client1Mac_66[] is true.
```

As seen above the attacker is not able to retrieve any of the tagged data thus, it is verified that this solution also works.

8.4.5 Beacons

Ignored in the previous chapters was that beacons are leaking data due to one field in the packet: the TIM field. This field is a list of bits that signals to clients that data is available to them by setting the bit that corresponds to the client to one, this could be used to track the activity of said users. This is solved by encrypted these fields. The AP starts with hashing the current timestamp to generate the IV that will be used to encrypt the bit (2). Then to build the TIM field, it will loop through all connected clients (4) and checks whether the client is a new client or a legacy client (5). It will encrypt the bit using the key that is shared with each client and the generated IV if it uses the new solution (6) or simply populate the field with the unencrypted data (8).

```
AP
    1. Beacon = new BeaconPacket()
    2. BeaconIV = hash(Beacon.timestamp)
    3. Beacon.TIM = new Array()
    4. For Client : Clients
    5. IF Client.hasPrivacy then
    6.     Beacon.TIM[Client.id] = encrypt(hasData(Client), Client.sharedKey, beaconIV)
    7. Else
    8.     Beacon.TIM[Client.id] = hasData(Client)
    9. End
```

The client creates the IV by also hashing the timestamp (2). It then looks up its hasData bit in the TIM bitmap and decrypts this bit using the shared key and the IV (3).

```
Client A
    1. Receive(Beacon)
    2. BeaconIV = hash(Beacon.timestamp)
    3. HasData = decrypt(Beacon.TIM[clientId], sharedKey, BeaconIV)
```

The Proverif script also includes a legacy client, ClientB, to make sure this does not affect the new client, ClientA.

```
RESULT not attacker:ClientBitB_86[!1 = v_594] is false.
RESULT not attacker:ClientBitA_85[!1 = v_913] is true.
```

As seen from the output it shows that the new client is safe and not affected by the older client, but the older client is still vulnerable, as expected.

8.4.6 Power management

If a client uses the power saving features available in the protocol the following sequence will be used by the client. In this sequence, a client receives a beacon in which the TIM bit was set for this particular client. This signals to the client that data is available at the access point, the client then has to request the data by sending a PS-Poll packet to the access point. Unfortunately, this leaks data about the topology of the network as this says that there are devices that use the power saving features of the network. The only solution for this problem is to stop using the PS-Poll type packets thus removing the power saving feature. Instead of removing it completely another route was chosen: the packet will be encapsulated in a normal data packet like the WPA2 authentication sequence. Since this would essentially result in the same sequence of data transmission as with the data and WPA2 authentication sequence no further implementation details are described here. The following variables were tested: Moredata flag, Power management flag, TIM clientbit, Assoc ID, SSID, BeaconData, SequenceCtl, APMac and ClientMac. Running Proverif gives the following output:

```
RESULT not attacker:moredata_80[!1 = v_1736] is true.
RESULT not attacker:pwrmgmt_79[!1 = v_4944] is true.
RESULT not attacker:clientBit_135[!1 = v_9536] is true.
RESULT not attacker:assocId_71[] is true.
RESULT not attacker:beaconData_130[!1 = v_18717] is true.
RESULT not attacker:SSID_64[] is true.
RESULT not attacker:seqctl_131[!1 = v_27898] is true.
RESULT not attacker:apMac_65[] is true.
RESULT not attacker:clientMac_66[] is true.
```

Proving that this solution solves the problem.

9 Implementing solution

9.1 Place for changes

Whenever a packet is received or transmitted a certain path of execution is walked through. As described in the chapter 7.2.2, the location where the solution is implemented has some restrictions. Furthermore, adding changes on multiple points on the execution path is also less desirable. Therefore some research into the execution path had to be done to select the correct location. In this case the assumption is made that some kind of *unix system is used due to availability of code and ease of use. The full overview of the path can be seen in Figure 75 and looking at this figure one thing immediately stands out: there are two paths between the software and the Wi-Fi hardware. Which path is taken depends on hardware, as seen there are two types: the first is the Soft-MAC side, on this side the MAC layer is implemented in software that lives in the kernel. The Full-MAC side implements many of the MAC features into the hardware. Both versions have their advantages and disadvantages. In this case, some changes needed to be made the Mac software layer, which makes the Full-MAC driver unusable. Now that the path between user and hardware is known, a location for the solution could be selected. The following three candidates remain: Full/Soft MAC driver, mac80211 and cfg80211. The first candidate is cfg80211, the highest level of the drivers, this one is chosen because the higher level the entry point will be the more devices will be supported. The cfg80211 layer is essentially a configuration layer; this layer will hold user settings about certain networks. A look at the entry points for this module show that this layer does not receive packets, it only receives statistical messages and requests for network information. The next layer is the mac80211 layers, a quick look gives positive results: there are functions that process both incoming and outgoing packets and has functions to disable hardware encryption. Disabling hardware encryption is important because processing of the packets needs to be done before they are decrypted. Furthermore, it also has specific functions that allow drivers to call these functions within an interrupt. This is especially important because it should mean that these functions would be called directly by the hardware whenever a new message is received. The next step was to add some rudimental hooks into key functions and check what kind of data packets were available in these functions. After some testing, it showed that this layer processes both incoming and outgoing packets and thus is useable for this solution. Lastly, the driver was looked at and some investigation showed that the driver simply pushes all messages received from the mac layer above to the hardware below and vice versa. Thus, the mac80211 was chosen as a place where the solution would have to be implemented.

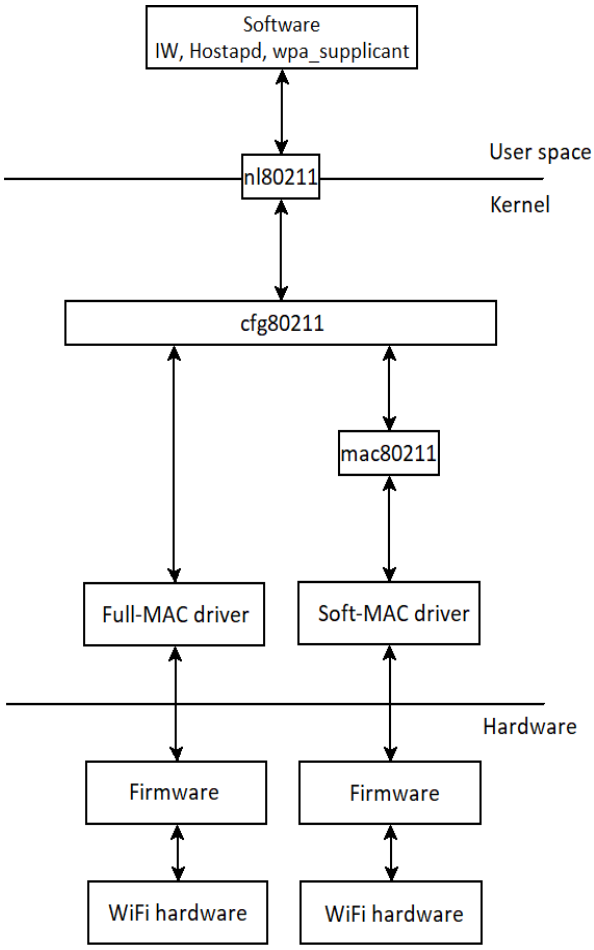


Figure 75: Overview of execution path

9.2 Implementation setup

The next problem lies with the fact that the module identified for modification has no unit tests or any other method of testing the correct working of the original or modified module. Therefore, a setup had to be created to test the new solution. The solutions would consist out of three different systems:

- An access point
- A client
- A sniffer

Figure 76: Overview of the setup

All these systems would then be connected using ethernet to keep easy access and the ability to push module updates on the fly to the systems. Both the AP & client would load the modified kernel module after which the AP creates a WPA2 home network. The client would then connect to the created network. Meanwhile, the sniffer would have its wireless adapter in monitoring mode to listen to all traffic generated by the other systems. The sniffer is used to verify the changes to the module are affecting the traffic that is generated.

9.3 Implementation

Before the solution could be implemented some structures need to be further defined, the solution stays vague in certain aspects. One of these aspects is the actual data structure and the implementations of the maps that have to be kept. For the latter, the maps, it was important that fast searching in the list was possible. To provide this a hash map was chosen as the implementation for the map because of its performance: on average O (1) for searching, removing and inserting into the map. A hash map works by hashing the key, this hash is then used as index to which the value is inserted. In this implementation, three different hash maps were used to hold the network data list, connection data list and MAC list (depicted in Figure 65). The leaves the Wi-Fi config and client data, the Wi-Fi config is already provided by the mac80211 layer itself and the client data only contains the public and private key which are as such.

The first map is used to store data about each access point, as key the MAC address of the base station was used and as value a BeaconData structure was created (depicted in Figure 77). This structure would hold four variables: the public key of base station, a Boolean which would tell whether the base station even published its key, a timeout variable which would be used to determine which data was stale (to manage the size of the data list) and the time which is included in the beacon frame. The timeout variable is never described in the solution but is required in this case because otherwise the list of networks would grow over time. If for example some other map or list implementation was chosen this would not be required, an example would be a ring buffer: a list of fixed size where the last element would simply be overwritten by the newest element if the list was full.

BeaconData
Has key
Public key
Time out
Time

Figure 77: Beacon data structure

The second map would contain the connection data, as key the MAC address of the client was used and as value a structure called ConnectionData was generated (Figure 78). This structure contained five variables: the shared key that was generated in the key exchange, all the encrypted MAC addresses which would point to this client, all the IVs used to encrypt these MAC addresses and the MAC address of the other client. Both these lists are "windowsize" long as this is predefined to how many encrypted MAC addresses the implementation wants to be behind/ahead. Again the window is dependent on how much memory is available and thus dependent on hardware.

ConnectionData
MAClist[windowsize]
IVlist[windowsize]
Shared key
MAC address

Figure 78: Connection data structure

The third list would contain the same connection data structure but in this map the key would be the encrypted MAC addresses instead of the plaintext MAC addresses. This is useful to verify if a packet was addressed to the client in an efficient and fast way. In this map the connection data structure is actually only pointing to a connection data structure in the previous map, as the structure is already in the other list so some memory could be saved by not copying it but by referencing it.

9.4 Problems

9.4.1 Elliptic Curve Diffie-Hellman exchange

The kernel includes a module that implements all the algorithms used to make the elliptic curve Diffie-Hellman key exchange work but there is almost no documentation available about any of these functions. For example, when googling the function name that generates the shared key from a public key and a private key only 18 results are returned of which many are either the source code itself, indexers of the Linux kernel source and message boards discussing improvements to the code. Thus, not much information could be found which resulted in the following problem:

9.4.1.1 Public key size

The public key size problem became apparent when the protocol was implemented and the keys were exchanged. Both sides should then be able to generate a shared secret key using the others public key and their private key. Unfortunately, the shared key, which should be the same on both sides, would not match. Which in term meant that the new implementation would not work. As if you symmetrically encrypt something the same key is required to decrypt it. After some more research and analysis of the shared key generation function, it was found that the public key had to be double in size of the private key. After fixing this, the key exchange was successful in generating a shared secret. After more research into what the public and private key actually represent it became clear. Unlike in for example RSA, the public key of an elliptic curve does not represent an integer but instead a point(X, Y) on a curve (hence the name elliptic curve), the private key in elliptic curve encryption is an integer and therefore halve the size of public key.

9.4.2 Encryption

Recently changes were made to the cryptography API that the Linux kernel provides. This meant that a lot of the documentation and examples were still written for the old API. Furthermore, nearly all documentation and examples use asynchronous encryption/decryption which could not be used in this situation.

9.4.2.1 Initialization

Unrelated to the problems provided by the documentation another problem arose with one of the clients used in the test system. The client would throw random kernel panics whenever the client tried to connect to the access point. The kernel panics would usually include message about lock violations. After debugging this problem, it showed that whenever the encryption, which was used to encrypt the message, was initialized in certain execution paths a kernel panic would occur. As described in chapter 7.2.2, there are multiple entry points to the module to which changes were made. All these entry points could be called by either the user programs or the Wi-Fi driver. It turned out that in some cases these calls would be surrounded by locks and other measures to protect hardware from interfering whilst that call would execute. These locks/protections were in turn conflicting with other locks/protections that were used to initialize the cryptographic engine used to encrypt all messages. To solve this problem the initialization was moved outside these critical functions and instead was only run whenever the module was initialized. This meant that module would have to reuse the initialized engine repeatedly. For now, this was not giving any further errors but special care has to be taken whenever doing this because some calls come from interrupts that could interrupt a process using the same resource that could lead to undefined behaviour.

9.4.3 No ACK control

The final problem is that the acknowledgement packets transmission is not controlled by any software in the kernel (in Figure 79 everything above the hardware line), not in the driver or the mac80211 layer above it. This means whenever an encrypted packet is received which needs an ACK that that ACK will not be generated by the firmware or driver because the receiver address will not match that of the current device. The next question was could there be a way to circumvent this problem in any way. Two possible solutions were devised:

- Generate the ACK in software
- Change the MAC address to the encrypted MAC address that is expected

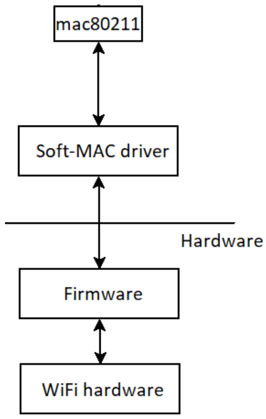


Figure 79: Kernel Wi-Fi stack

To check if the first solution would have any merit a closer look was required to the current situation. For this Wireshark was used to look at what kind of packets were transmitted by each client.

This resulted in the Wireshark output as shown in Figure 80. What can be seen here is that the client (ChengHon_21:bc:22) retransmits the packet to the base station (ChengHon_21:bc:52) several times, note here that the actual destination(5c:65:91:28:eb:12) is encrypted in these packets. The packet is then transmitted in plaintext, which changes the destination address to ChengHon_21:bc:22, after which the access point responds directly with a plaintext message finishing the authentication part.

Source	Destination	Length	Info
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=.........
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bd:52	Broadcast	174	Beacon frame, SN=1, FN=0, Flags=........., BI=100, SSID=RemieTestWifi
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	5c:65:91:28:eb:12	115	Authentication, SN=12, FN=0, Flags=....R....
ChengHon_21:bc:22	ChengHon_21:bd:52	49	Authentication, SN=13, FN=0, Flags=.........
	ChengHon_21:bc:22	29	Acknowledgement, Flags=.........
ChengHon_21:bd:52	ChengHon_21:bc:22	49	Authentication, SN=774, FN=0, Flags=.........
	ChengHon_21:bd:52	29	Acknowledgement, Flags=.........
ChengHon_21:bc:22	ChengHon_21:bd:52	88	Association Request, SN=14, FN=0, Flags=........., SSID=RemieTestWifi
	ChengHon_21:bc:22	29	Acknowledgement, Flags=.........
ChengHon_21:bd:52	ChengHon_21:bc:22	80	Association Response, SN=775, FN=0, Flags=.........

Figure 80: Wireshark trace of authentication

Looking at the logs generated at the access points we get the following:

```
AP
[ +1.235640] encrypted auth
[ +0.000009] RX: Auth request: DA: 00:19:86:21:bd:52, SA: 00:19:86:21:bc:22, BSSID: 00:19:86:21:bd:52, mem: ffff8ebc79d55000
[ +0.001147] TX: Auth request: DA: 00:19:86:21:bc:22, SA: 00:19:86:21:bd:52, BSSID: 00:19:86:21:bd:52, mem: ffff8ebc76226d00
…
[ +0.000006] RX: Auth request: DA: 00:19:86:21:bd:52, SA: 00:19:86:21:bc:22, BSSID: 00:19:86:21:bd:52, mem: ffff8ebc79a42300
```

As seen in the first line the access point detects that a client tries to connect using the new implementation, the line after that shows that the packets has been successfully decrypted to its original packets. The access point then responds by sending an authentication request to the client.

The first two lines are then repeated several times but the access point does not respond any more. Then another authentication request is received this time without any key inside, shown in Figure 80 surrounded with red and thus the access point responds like it would respond to a legacy client.

The following logs were generated by the client:

```
Client
[  +0.002818] Auth: DA: 5c:65:91:28:eb:12, SA: 00:19:86:21:bc:22, BSSID: 00:19:86:21:bd:52, mem: ffff8a92983c6500
[  +0.000003] TX: Auth request: DA: 5c:65:91:28:eb:12, SA: 00:19:86:21:bc:22, BSSID: 00:19:86:21:bd:52, mem: ffff8a92983c6500
[  +0.002493] RX: Auth request: DA: 00:19:86:21:bc:22, SA: 00:19:86:21:bd:52, BSSID: 00:19:86:21:bd:52, mem: ffff8a92983d1500
```

These show that the retransmissions are not handled by the software because it only logs that it has transmitted a single transmission and received an unencrypted reply. The software does not have any influence on the retransmission of the packets. Luckily, analysis of the packets showed that these packets are essentially the same as the previous packets but with another flag set. The flag, which is set, has not been changed by the implementation and thus does not have any impact on the working of this solution. What is a problem is the fact that after several retries an unencrypted packet is transmitted which then leaks all the data that should have been hidden. This scenario does not happen every time because sometimes the access point is fast enough in its encrypted response, the client receives this response before it transmits the unencrypted packet and stops transmitting. But the fact that it can happen means that this solution will not be a reliable solution. Furthermore, the fact that an unencrypted retransmission is transmission without any interference from the software means that this could happen for any packet that requires an acknowledgement.

After some more research and another look into the data a possibility for the problem of unencrypted retransmission could be found: the firmware or hardware keeps a list of earlier connected networks and then responds to beacons. In this case a beacon was received two frames before the problematic behaviour.

The next solution was to change the MAC address to the next address that is expected. The first problem with this solution is that the access point does not know what the first address is because it needs to generate that when receiving the first message. Secondly, the access point will never be able to communicate with more than one client because it would otherwise not know which client transmits the next packet.

Therefore no solution could be found for this problem in the current implementation, as both solutions will not be feasible. The only other solution for this problem would be to implement the changes in a different location earlier in the receive path. This would mean that either the implementation would have to be implemented in the firmware of the device or in the hardware, the implications of this will be discussed in chapter 12.1.

10 Limitations

In this chapter the limitations of the new solution are discussed.

10.1 Statistical analysis

While the solution will not leak any information about the clients any more, data is still transmitted. Since the amount of data that is transmitted and the timing of this transmission is not changed some information about the client still might leak. For example, a static client like a printer might periodically transmit small keep alive message packets to some server. This would result in a very different pattern of packets than for example a client that is watching a video, this would result in big packets for the duration of the video.

10.2 Analogue information

In this research only the digital side, the software, was considered. Unfortunately, the bits that are transmitted are not merely one or zero but carry more information. For example, the strength of the signal is an important parameter that has not been considered. The strength could be used to determine the distance between the sender and receiver. An attacker could use multiple devices place strategically to determine the location of a device. In case of the problem: determining the occupancy of a house, this could then be used to determine the overall activity of the monitored house. Strength of the signal is not the only information that might be retrieved from it, small timing differences between transmitter implementations might also be considered to determine which client is transmitting the data.

11 Conclusion

This research consisted of two parts: the shared research trying to prove that tracking of household occupancy was possible by eavesdropping Wi-Fi traffic and the personal part trying to solve this by altering the current Wi-Fi protocol whilst keeping interoperability with current systems.

The shared part was already researched before for a course but was not statistically relevant due to used participants not being random. The new research would use students living on campus as participants, this changed resulted in an extra research question. The two research questions were defined as follows:

- Is it possible to determine which Wi-Fi devices belong to a certain household in a shared network with only passively detectable parameters?
- *Is it possible to reliably track occupancy in a household with passive eavesdropping on its Wi-Fi traffic?*

The first question showed that with the used filters we were not able to successfully separate the devices from households. Altering the filters to make them stricter would often result in the removal of devices which were considered correct. Furthermore, looking to communication between devices did not work either as devices were also communicating a lot outside of their own household.

Due to failure to answer the first question it was impossible to give a definite answer to the second question but looking at the previous experiment good results were expected. Manually going over the data and manually filtering a lot of the devices showed that the data was there but was reliant on selecting the correct devices. Thus, this definitely needs further research preferably using homes without shared networks, as shown by the previous this could give very good results.

Next was the individual part of the research, each of researchers would research a solution into the problem, this resulted in the following research question:

- Is it possible to find a solution that prevents tracking of clients whilst keeping interoperability with current implementations?

To answer this question the research was split into multiple parts:

- Research and identify the data in the current Wi-Fi protocol that makes users trackable.
- Verify with Proverif that this information is indeed leaked by the current protocol.
- Alter the protocol in such a way that it would not leak information but keep interoperability with the current protocol and implement and prove this solution in Proverif.
- Implement the new protocol on an access point and a client to prove that it works and is still interoperable with older clients.

The identification of the problem and verifying this in Proverif proofed to not give any problems. The third step, altering the protocol, had some problems at first but could eventually be finished. The final step however did provide a problem which could not be overcome: there was no control over acknowledgement frames which made implementing the new solution impossible. The solution did show promise as the hard part, implementing the key exchange and the encryption of the packet, was already implemented and tested. Thus, if it were also possible to alter the acknowledgement frame then the final part of the research would also be successful.

Going back to the original research question some further remarks had to be made. The solution was proven by Proverif but it did not account for other possible parameters as described in chapter 10. Keeping this limitations in mind it can only be concluded that it is impossible to create a solution that completely prevents tracking of clients. Though, if this solution could be implemented successfully in for example the firmware of a wireless chip it will increase the difficulty for attackers a lot. Furthermore, when looking at the compatibility of the new solution it is a definite improvement upon all the other implementations. As proven in chapter 7, it is possible to encrypt the messages whilst keeping the format the same and thus it can be used even in implementation that use a MAC hardware implementation. Furthermore, the current implementation has been setup in such a way that only a few extra function calls had to be added to the existing driver, albeit in the correct location, to implement the new solution thus allowing easy adoption to the new solution.

12 Discussion

12.1 ACK control

The implementation of the solution failed due to not having control over the ACK messages. Resulting in that either the implementation would have to be implemented in the firmware of the device or in the hardware. The expectation is that if it was possible to change the firmware on the device then ACK messages could be controlled and thus would make the solution feasible. Unfortunately, this cannot be verified as only the manufactures of these devices know the proprietary code running on these devices and know how these devices are implemented. But if this is possible it is expected that this is an easy change because on many of the devices the firmware is transferred to the device when it is initialized. This firmware (usually called binary blob) is included in the driver package, thus a driver update would also allow a firmware update. Thus, it would lower the speed and ease of the roll-out of the new solution. Instead of having an open source driver already implemented with the changes manufactures will have to implement the changes in their firmware, though implementing the solution should not be a big problem. As the solution was though out in such a way that only in a single location in the execution path code has to be added.

12.2 Usage of PSK key

To solve the man in the middle attack that is possible in a Diffie Hellman key exchange an extra encryption step had to be added to obtain the shared key. This extra encryption step uses the PSK key that is originally used in the WPA2 key exchange to generate the WPA2 keys. This makes the new solution dependent on an already known shared key. As the solution already requires a form encryption, as unencrypted traffic would leak information about the client in the data packets, this is not a big problem, but a more interesting question arises from this dependency: does the solution really need to be resistant to a man in the middle attack? One could argue not, because in the scope of the problem this would mean an attacker would start to transmit data to and from the client and the access point. This could then easily be detected either by the client or by the access point. Therefore in the scope of the problem, this would not be a real problem.

References

Agence nationale de la sécurité, des systèmes d'information, 2014. RGS_v-2-0_B1.pdf [WWW Document]. URL https://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf (accessed 3.17.18).

Ajay Kumar, Antony Jerome, Gaurav Khanna, Hari Veladanda, Hoa Ly, Ning Chai, Rick Andrews, 2013. Elliptic Curve Cryptography (ECC) Certificates Performance Analysis [WWW Document]. URL https://www.websecurity.symantec.com/content/dam/websitesecurity/digitalassets/desktop/pdfs/whitepaper/Elliptic_Curve_Cryptography_ECC_WP_en_us.pdf (accessed 3.17.18).

Blanchet, B., Smyth, B., Cheval, V., 2016. ProVerif 1.94 pl1: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial.

Computer Security Division, I.T.L., 2006. NIST Comments on Cryptanalytic Attacks on SHA-1 [WWW Document]. URL https://csrc.nist.gov/news/2006/nist-comments-on-cryptanalytic-attacks-on-sha-1 (accessed 3.17.18).

Dalal, H.N., Soni, N.V., Razaque, A., 2016. Header encryption of IEEE802. 15.4, in: Long Island Systems, Applications and Technology Conference (LISAT), 2016 IEEE. IEEE, pp. 1–6.

Freudiger, J., 2015. How talkative is your mobile device?: an experimental study of Wi-Fi probe requests, in: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. ACM, p. 8.

Greenstein, B., McCoy, D., Pang, J., Kohno, T., Seshan, S., Wetherall, D., 2008. Improving wireless privacy with an identifier-free link layer protocol, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services. ACM, pp. 40–53.

Gruteser, M., Grunwald, D., 2005. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. Mob. Netw. Appl. 10, 315–325.

Housley, R., 2004. Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP) [WWW Document]. URL https://tools.ietf.org/html/rfc3686 (accessed 3.18.18).

Könings, B., Schaub, F., Weber, M., 2013. PriFi beacons: piggybacking privacy implications on wifi beacons, in: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. ACM, pp. 83–86.

Levi, A., Savas, E., 2003. Performance evaluation of public-key cryptosystem operations in WTLS protocol, in: Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003. Presented at the Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003, pp. 1245–1250 vol.2. https://doi.org/10.1109/ISCC.2003.1214285

Lindqvist, J., Aura, T., Danezis, G., Koponen, T., Myllyniemi, A., Mäki, J., Roe, M., 2009. Privacy-preserving 802.11 access-point discovery, in: Proceedings of the Second ACM Conference on Wireless Network Security. ACM, pp. 123–130.

Lisa Sullivan, PhD, n.d. Issues in Estimating Sample Size for Confidence Intervals Estimates [WWW Document]. Power Sample Size Determ. URL http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Power/BS704_Power2.html (accessed 6.10.18).

Mobile marketing statistics 2018 [WWW Document], 2018. . Smart Insights. URL http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/ (accessed 4.9.18).

On Collisions for MD5 - M.M.J. Stevens.pdf, n.d.

Vanhoef, M., Matte, C., Cunche, M., Cardoso, L.S., Piessens, F., 2016. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms, in: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM, pp. 413–424.

Verbree, E., Zlatanova, S., van Winden, K.B.A., van der Laan, E.B., Makri, A., Taizhou, L., Haojun, A., 2013. To localise or to be localised with WiFi in the Hubei museum? ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. XL-4/W4, 31–35. https://doi.org/10.5194/isprsarchives-XL-4-W4-31-2013

Xing, X., Shakshuki, E., Benoit, D., Sheltami, T., 2008. Security analysis and authentication improvement for ieee 802.11 i specification, in: Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE. IEEE, pp. 1–5.

Appendix

Appendix 1: Small scale research

Digital stakeout analysis

Ruben Lubben
University of Twente
Drienerlolaan 5
7522NB Enschede

Tim Kers
University of Twente
Drienerlolaan 5
7522NB Enschede

Remie Löwik
University of Twente
Drienerlolaan 5
7522NB Enschede

ABSTRACT

Modern times brings modern crime. The presence of Wi-Fi enabled devices make it possible to extract life patterns of members of a household, pinpointing the times at which the house is empty. Potentially a great tool for burglars. By analyzing 12 households we can determine whether a person is home or not correctly for an average of 86.7% of time. Considering that this is done with limited hardware makes that this technique could be described as a potential threat. The social effect on the participants is limited but they behave as expected. The amount of users that feel unsafe sometime at home rose from 50 to 58.3%. The neighborhood safety rating declined from 7.5 to 7.3. The deemed chance of a burglary occurring in the next 12 months rose from 2.5 to 2.66 (both on a scale of 10). A potential reduction on the usability of this type of data extraction would be to switch over to the 5 GHz Wi-Fi standard.

Keywords

Digital stakeout, presence detection, modern burglary, burglary techniques, sense of safety based on exposure

1. INTRODUCTION

More than 4 billion Wi-Fi-enabled devices are currently in use [1] and the predictions are that there will be over 7 billion in 2017. Although users are more privacy aware nowadays, Wi-Fi-enabled devices do give some information away. This information could be used for criminal activities.

For example, Burglary, because this is still a big concern in our modern society [2][3]. When more and more people are away from home for longer periods of the day [4], the opportunities for burglary rise. The biggest concern with this type of crime isn't the material damage but the loss of faith in the safety of our own environment [5][6]. Over time we've played a cat and mouse game with burglars with better protections to keep them out and systems to deceive them. Think of all the time switches, used when people are on holiday, to make them think we're still there. However, anyone could potentially track occupants by tracking their Wi-Fi enabled devices. Although there aren't any reports that burglars do this, we think that with little effort, life patterns and current presence of people could be detected with very cheap hardware. The rise of the Wi-Fi enabled devices has created a risk that has not yet been explored by burglars. Although there aren't reported crimes with this phenomenon, we do already see big companies use this kind technique to track costumers in department stores and on large festival areas [7].

2. Method

The objectives for this research can defined as three research question. These questions are:

"Can we measure the presence of a person in a household based on Wi-Fi-enabled devices?"

"Can we extract life patters from a household based on Wi-Fi-enabled devices?"

"Does the knowledge of the ability to measure their presents effect the level of safety a persons experienced."

The goal of this research is to test if we can measure the presence of a household. When this research question is confirmed, we like to extend the question if we can reliably extract life patterns. The idea behind this is, that if we are able to establish this, someone with a malicious intent would also be able as well. This will make the job for a burglar much easier as he can reliably determine and/or predict if someone is away from home.

Secondly, this research aims at the psychological aspects of burglary in general. How safe does the consumer feel in and around his or her house and is this sense of safety influenced when people learn about the possibilities of Wi-Fi enabled life pattern extraction?

2.1 Approach

To answer our research questions, we need to collect all the Wi-Fi data transmitted by Wi-Fi-enabled devices from multiple households in different settings and measure the sense of safety in house before and after the experiment. The first part is achieved by placing another Wi-Fi-enabled device in these households which has the sole purpose of listening to all Wi-Fi-traffic in the direct vicinity. The content of the streams will not be stored but the mere statistics of each device and/or stream will be kept and stored in a database. This data holds no further information than the presence of device X on time Y. The second part is achieved by surveying the subjects before and after the experiment.

These listening devices are left to track these statistics in the selected households for a period of time after which they are retrieved. With the datasets created at each household we try to create a timeline indicating which recorded device was present at what time. After this stage, we go back to the households and confront the residents with the schedules we derived from the data. At this point we match devices with the actual owner by a time sheet they kept during the experiment and verify our extracted schedule with theirs. This is to verify if we are able to accurately trace someone's schedule by tracking their Wi-Fi enabled devices. After we confront the residents we ask them to fill in the survey again and we compare the results of the survey with the one we asked them to fill in before the test started.

If we can accurately trace someone's presence, we can also extract returning events. For example, a resident leaving every Wednesday evening for a couple of hours. The actual activity of this resident is not important for our research, we only want to know when the house is unoccupied. For a potential burglar, knowing the house is empty every week at that day and that time is a great piece of information

So this research will first do a survey to test the feeling of safety on forehand. Meanwhile we measure the presence of the household by capturing all the Wi-Fi-signals. During the measurement period the persons in the households will keep a diary of their presence. After a period of 2 weeks we take the data and extract the presence based on the Wi-Fi-data. After processing we will confront the participants with the findings and show how correct the data is. To track the changes of the feeling of safety, we will do a new survey to reveal the impact of this findings. The surveys are based on the annual "nationale veiligheidsmonitor" (national safety monitor) of the Dutch Office of National Statistics (CBS) commissioned by the Ministry of Security and Justice. This should give us a survey with questions that should not bias the participant.

The approach of this research is carefully discussed with and approved by the ethical committee of the University of Twente with the remark that the privacy of the participants of this research is quite well ensured.

2.2 Technique

To be able to trace the occupants, their Wi-Fi enabled devices need to transmit identifiable signals during the day. Fortunately, the 802.11 standard defines multiple frame types that are used for managing a wireless link. In a wired network, a device is easily recognized and connected by plugging in the cable. When it's disconnected again the network will notice this immediately. With wireless networks, this is obviously different. A wireless device first needs to find a compatible network in its reachable area and then be authenticated by this network. Only after these steps the device is associated with this network and able the backbone.

To setup and sustain such a network, the following set of management frames is used [8]:

Beacon, Probe Request, Probe Response, IBSS announcement traffic indication map (ATIM), Disassociation, Association Request, Reassociation Request, Association Response and Reassociation Response, Authentication, Deauthentication

Most of these frames are used in the connecting and authenticating process of a device to a wireless network. As we want to continuously monitor the devices that are present in the area, most of the frames are useless for monitoring continuously in a passive way. After careful inspections, the Beacon frame, Probe request and the Probe response turned out to be useful for this research.

Probe requests are send by Wi-Fi enabled devices to test if a certain device, such as the access point, is (still) in its vicinity. Beacon frames are send by the access point to indicate its own presence. Because all these messages contain a header with a sender and receiver mac address we can track these devices.

Another important aspect of Wi-Fi is that it uses two frequency spectrums: 2.4GHz and 5GHz. In this research we focus on 2.4GHz which is divided in 16 channels each 5 MHz wide as defined in the 802.11 standard. The arrangement of these channels is shown in figure 1. Because wireless signals interfere with each other we want to have as much space between the channels as possible to prevent noise in the transmission. That is the reasons why the channels 1, 6 and 11 are blue. These are called the super channels because their frequency range is completely separated from each other. Therefore, these channels are mostly used [9].

This frequency division in channels is an important factor in the rest of the experiment as a Wi-Fi receiver can only listen on 1 channel at the same time.

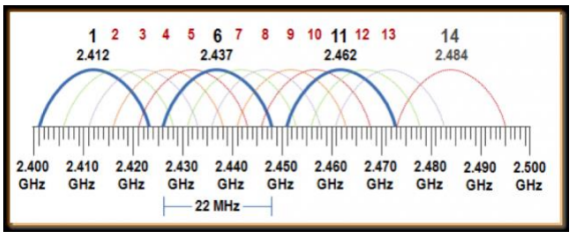


Figure 1:2.4GHz frequency division

2.3 Test setup

To monitor the wireless network in our test cases, we used a Wi-Fi enabled device of our own. A normal laptop computer with a version of Ubuntu Linux installed. On this laptop, we used two programs to capture the data and we created a program that analyzed, anonymize and compressed the output of these programs.

TCPdump is used to capture all data transmitted on the frequency channel currently in use. With this software we capture the probe requests, beacon frames and normal data frames. From the probe requests and data frames we extract the identification numbers of the device sending and receiving this packet. These are the so called MAC addresses. These unique numbers are used to identify the different devices during the tests. As long as we receive packets send by or to a certain MAC address, that device is still active.

We also created a program that runs as a service on the device we placed. This program starts the TCPdump program, analyzes the output of the program, creates the statistics and stores them to the local hard drive.

As the airwaves get increasingly busy with different Wi-Fi networks, sometimes a wireless access point decides to switch over to a different frequency channel with less interference. As our system can only monitor one channel at a time, this would mean we would lose our eavesdropping capability at this moment. To litigate this problem, we also capture the Beacon frame. This is used by the access point to indicate its presence on its channel.

When the program stops receiving Beacon frames, we lost communication with the access point. When this occurs, our program automatically calls the second program called Airmon. Airmon is used to quickly hop to a different channel. In the meantime TCPdump is still running, so every time Airmon locks onto a new frequency, we start to receive the data transmitted on that channel. If our access point is on this channel, we will receive beacon frames with the right mac address again. If it isn't, we skip to the next channel. When found, Airmon is stopped as we no longer want to switch frequencies.

2.4 Data storage

After capturing, the usable data needs to be stored for later processing. This poses some extra challenges. The simplest form of storing all data we capture poses some problems. It creates a very large dataset and more importantly, it contains a lot of privacy sensitive information. Therefore the dataset is slimmed down and anonymized.

From all wireless packets we look at the beacon frames, data frames and probe requests. We keep track of all beacon frames we capture. As more than one access point can be present on one frequency channel, we keep a list. In this list we store which device (by MAC address) sends a beacon frame at what time (per minute).

A similar approach is used for probe requests. Only in his case it concerns the MAC address of the Wi-Fi enabled device sending the request and the time at which the request is send.

For the data frames we store a little bit more information, 5 items in fact. First we store three MAC addresses, that of the network router, the sender of the packet and the receiver of the packet. Additionally the time at which the packet is captured and the amount of data packets our device captured in the last minute. This last field can potentially be interesting for more detailed profiling of someone's schedule.

In addition to these packets, the system also receives and stores the acknowledgments usually transmitted by the router after correctly receiving a packet. Only the receiving MAC address of this packet is stored (the device that send the acknowledged packet).

As the data is potentially privacy sensitive all data entries are then hashed with a SHA-256 type hash. This non-reversible hash makes the data not linkable to specific devices, households or persons. In the case that a potential burglar would get a hold on this data and be able to process it correctly, they still don't know which households to apply their found schedules on. With more than 7.5 million households in the Netherlands [8] those schedules are useless.

2.5 Test scenario

To answer our research questions, we placed 22 laptops during a period of two weeks in 22 different households. By the limitations caused by the ethical committee of the University of Twente we only choose households where no children are present. The households were chosen between relatives of the 3 researchers. The laptops were placed in a central place in the house where the chance of picking up Wi-Fi signal is the biggest.

To verify our findings, the test subjects were also asked to keep track of their presence during these two weeks. The subjects were informed about this experiment and asked to not behave differently during this experiment.

To test the feeling of safety in a household before and after the intervention of this test, we perform a survey based on the "veiligheidsmonitor 2016". This survey should give a good indication of the level of safety, experienced by a member of the household.

Concerns about how participants behave differently due to participation of the research is not addresses in this situation. We assume the participants do not behave differently.

3. Results

After collecting all the devices in the different households, all the data is extracted to a central server. All the data is then filtered, processed and analyzed.

3.1 Filtering

To get a useful reading of the raw data we need to exclude some devices to make the data useful. Because Wi-Fi signals are not bound to a specific household, especially in urban areas, where signals from neighbors can easily be recorded, the solution is to fix the reading towards a dedicated access point. The idea behind this is that a specific access point belongs to a specific household. By checking the headers, we can passively extract if the Wi-Fi enabled device is connected to that specific access point and discard all the other data. This filtering is done by looking at the mac headers and see towards which device a packages is addressed. If these packages are addresses to the mac address of the router that is based in that household, we keep that data.

After extracting all the data that corresponds with the access point, the real filtering can begin.

Step 1: normalizer activities per 15 minutes
Due to energy saving techniques in modern devices, devices will tend to sleep periods of time when not in use. During these periods, the amount of packets send by the device is severely limited. Therefore we need to determine a cooldown period for a device's presence. This prevents sleep cycles to be recognized as leaving and returning and therefore smooths the data. By analyzing initial test data we see that almost all devices send signals within 15 minutes. For that reason, we chose to set the cooldown period of the data on 15 minutes.

Step 2: filter out overactive and inactive devices
In the data we saw that people had a lot of Wi-Fi enabled devices at home. Some of these devices are always present and therefore not relevant in determining presence. To filter these out we introduces two thresholds. The first threshold: a device should be offline for at least two consecutive hours in the whole test period. If not, we assume this device is permanently at home. The second threshold: a device should be active for at least 2% of the total measuring period. This is to exclude visitors and other strange devices. After testing with the initial test data, the 2% proved to be the most effective filter.

Step 3: merging devices
To make the data much more manageable and reduce the weight factors we merge the devices that occurred in the full time span of each other. For example if you only watch TV when you at home and you always have your phone with you at home. The activity of the TV will always fall in the time span of the presence of the phone. In that case we merge the TV with the mobile phone. Note: If there is an occurrence where the TV is active and the phone isn't, there cannot be a merger.

Step 4: manual filtering
Although the previous steps filter out a lot of unwanted data, we still need to process manually. Certain patterns of "rogue" devices are difficult to filter out by rules but are very easy to spot when the graphs are plotted. For example people went on a holiday and then switched off their normally permanent devices.

3.2 Processing

By processing the results, we want to create a table that indicates the correctness of our technique compared with schedule that the households have filled in. To create this table, we process the data in a couple of stages:

Stage 1: Determine a measuring point where the house is absolute empty. This is done by counting all moments a device is active in an hour and divide that by the number of devices that ever occurred in the measurement after the filtering. An example can be found in table 1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1-5-2016																					1	1	1	1
2-5-2016	1	1	1	1	1	1	1	0,8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,9	1	1	1	0,5	0,5
3-5-2016	0,5	0,5	0,5	0,5	0,5	0,6	1	0,8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	1	0,8	1	0,6	0,5
4-5-2016	0,5	0,5	0,5	0,5	0,5	0,6	1	0,8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,8	1	1	0,8	1	0,6
5-5-2016	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,8	0,9	0,9	0,9	1	1	0,9	0,8	1	1	1	1	1	1	1	1	0,6
6-5-2016	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,8	1	1	1	1	1	1	1	1	0,3	0	0	0	0	0	0	0
7-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,5	0,5	1	1	0,6	0,5
9-5-2016	0,5	0,5	0,5	0,5	0,5	0,5	1	0,8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,9	1	1	1	0,6	0,5
10-5-2016	0,5	0,5	0,5	0,5	0,5	0,6	1	0,8	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,9	1	0,8	1	1	1
11-5-2016	1	1	1	1	1	1	1	1	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,9
12-5-2016	1	1	1	1	1	1	1	1	0,6	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,9	1	1	1		

Table 1: Raw occupancy output

Stage 2: Now we choose the lowest value in the table of stage 1 as the "being absent" marker and make a binary table differentiating between "absent" or "present". An example table can be seen at table 2. These are the moments where this approach think the house is empty.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1-5-2016																					1	0	0	0
2-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
7-5-2016	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8-5-2016	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
9-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 2: Binary occupancy output

Stages 3: In this stage we need to compare our results with the dairy that the test subjects have kept during the measuring period. In the table below we see the majority of the fields marked with a zero. This represents a correctly predicted timeframe (occupied and vacant). Cells marked with the number 1 represent time slots where people say they weren't home but our analysis stated differently (from now on referred to as "false occupied"). Lastly the timeslots with the number 2 are moments where the data stated the homes are empty where they are not. This is later referred to as "false vacant".

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1-5-2016																					2	0	0	0
2-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
3-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
4-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
11-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12-5-2016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 3: Compared output

3.3 Capture data

During the experiment, only 12 of the initial 22 test runs returned usable data due to hardware and software issues. These datasets were processed and the average results are shown in Table 4.

	Average occurrence
Correctly predicted	86.69%
False occupied	10.49%
False vacant	2.82%

Table 4: average predication rates over 12 households

Here we can see that our system is able to correctly predict the occupation of the house for an average of 86.69 percent of the time. This is already a fairly good score. When we take a realistic approach, we can regard the false occupied values as semi-good results. The false occupied parts are the moments were the system stated the house is not empty where it was. Although the system isn't correct here. In the case of a potential burglar, these faults are not extremely problematic. Besides, a large portion of the false occupied occurrences were only short absence recordings. These can be caused by actual short absences of subjects (for example a short trip to neighbor or a close by shop) that aren't recorded on the timesheets. As a potential burglar would be most interested in longer absences, we asked to focus on the longer absences on the timekeeping sheets.

The actual risk of this system lies in the false vacant faults where a burglar is at risk of being caught in the act by the residents deemed to be away. This fault only occurred in 2.82% of the time. For a first experiment, this can be considered fairly good.

The difference in fault rates across the households is clearly noticeable as shown in figure 2.

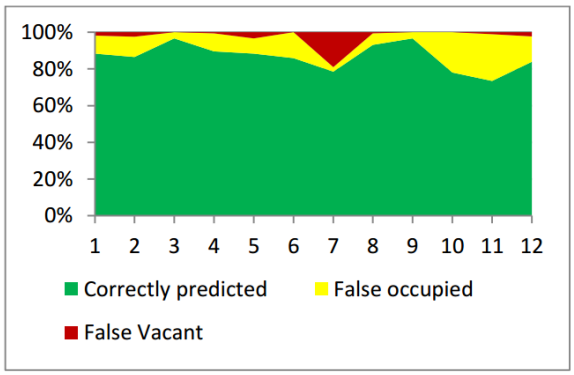


Figure 2: Fault rates across households

Also, a lot of the faults (false vacant and false occupied) only occur for short periods of time. A would be burglar using this kind of system would be most interested in predictable long absences like work hours. This reduces the impact of a part faults. This kind of long absence is clearly visible in figure 3, where the red plot shows a device being vacant from 06:40 till 16:40. A weekly occurrence of such a schedule would be a great piece of information.

Figure 3: Daily occupancy of a household

The data also reveals other interesting information. Because we capture all the devices we found evidence that somebody has likely changed phones in one of the household because as you can see in figure 6 (blue and green lines) somebody left the house and never came back while another device entered the same household hours later and continued the trend the previous device created in the previous days.

A problem we encountered in the experiment was the limitation of Wi-Fi reception of our test devices. This influenced the accuracy of the readings as well. Figure 4 shows the results of a home where the false vacant readings would skyrocket at night:

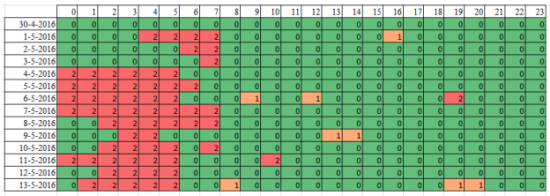


Figure 4: Range limitation example

The false vacant rate of this households turned out to be 19.05%. This was very likely to be a range issue. Later, when the results were presented to the residents, they confirmed that they were having issues with Wi-Fi range on the first floor where the bedrooms were. As our measurement device placed on the ground floor, just as the wireless access point, our range would probably also be an issue. On top of that, some of the residents turned off their Wi-Fi when they went to bed to resort to the mobile networks instead. At that point we lose their connection too.

As we expected this issue to play in more households, a test was executed where measurements between 0:00 and 7:00 where ignored. In other words, only day- and evening occupancy was checked. This improved the scores, especially the false vacant rate was drastically reduced as seen in Table 5:

	Average occurrence
Correctly predicted	89.33% (+2.65%)
False occupied	9.98% (-0.52%)
False vacant	0.69% (-2.13%)

Table 5: average predication rates over 12 households with 7:00 to 23:59 schedule

This difference is also shown in the graphs with data separated per household in figure 5.

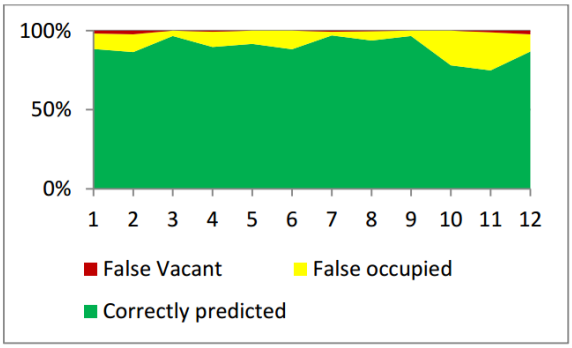


Figure 5 - presence representation

This difference does make it clear that, to become a usable system, the range of the listening device needs to improve drastically. Remember that the would-be burglar would probably place the device somewhere around the house instead of in the living room.

3.4 Survey analysis

For the research we have conducted 2 surveys. The first survey was to measure the feeling of safety before the research and the second survey was to measure the change of the feeling of safety. Although we used a standard survey, the participants could be a little bit biased because the experiment was explained beforehand.

From the survey results, we focused on three aspects. The occurrence of feeling unsafe, the rating of safety in the neighborhood and the deemed likeliness of falling victim to a burglar in the next 12 months. These ratings before and after the experiment are presented in figure 6.

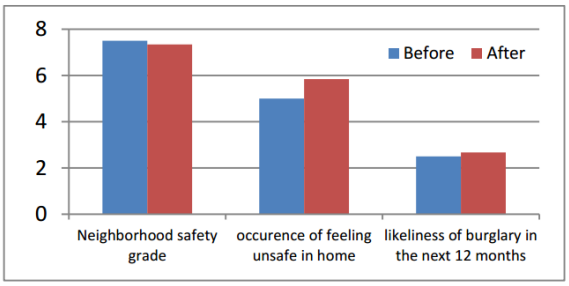


Figure 6: Results of the social intervention

As can be seen in figure 6, the neighborhood safety grade changed from 7.5 to a 7.33. Meanwhile the amount of people feeling unsafe sometimes rose from 50% to 58.33%. The likelihood of becoming victim of burglary also rose. The grade went from 2.5 to 2.66.

Unfortunately, because of the reduction in usable household datasets together with a number of people not filling in (one of the) surveys, the final dataset resulted only in 18 completed interventions. Although the results look promising, this makes the statistical relevance low.

After the survey we asked some extra questions to our test subjects. As it turned out, two thirds of our test subjects deemed data extraction somewhat or completely possible a as shown in figure 7. Most important, nobody thought it was impossible.

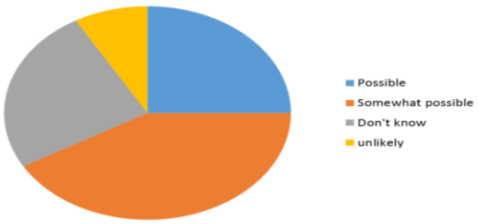


Figure 7: Deemed likeliness of presence extraction being possible

Also, a third of the test subject claimed the results of the research had influence on their sense of safety in their home. Largely the same people said it influenced their view on existing burglary prevention techniques. People recognized that the availability of information has made them more vulnerable. One subject even came with the idea of running a laptop to deceive the measuring device.

4. Discussion

This chapter contains the future research and the conclusions of this research. The main goals of this research were:

"Can we measure the presence of a person in a household based on Wi-Fi-enabled devices?"

"Can we extract life patters from a household based on Wi-Fi-enabled devices?"

"Does the knowledge of the ability to measure their presents effect the level of safety a persons experienced."

The tests showed a correct prediction rate of 86.69% with a 2.82% "fatal" failure rate. We regard this as a fairly good result. After removing the hours between 0:00 and 7:00, which gave a lot of the faults due to wireless range issues, these rates changed to 89.33% and 0.69%.

Due to time limitations, measuring windows were limited to 2 weeks. This made recurring pattern recognition problematic. Some households seemed to have certain patterns, but with such short measuring period, no credible conclusion could be drawn from the data. Therefore this research question remains unanswered. But with the said accuracy rating on the presence detection, it seems likely that pattern detection is possible.

The intervention showed an increase in people feeling unsafe in their home and a decrease in their "neighborhood safety" score. The expected chance of becoming victim to a burglary in the next 12 months was also raised slightly. The outcome of the experiment therefore seems to have an effect. The question, however, is if this difference is only a temporary effect.

Unfortunately the statistical relevance can be disputed due to the small amount of test subjects.

The study shows that this type of data extraction is a potential thread for the future, and with the (psychological) problems that occur with burglary, this can become a big concern. For the residents themselves, the potential problem doesn't seem to have a very large influence. But as wireless technology is all around us, this can also be caused by the lack of understanding about where the vulnerability lies and what could be done to reduce the risks.

The easiest solution to this problem would be to stop using wireless networks. However, this option won't be feasible in practice. A potential solution could be to phase out the 2.4GHz Wi-Fi networks and switch to the 5GHz version. This type of network suffers from greater signal strength loss through walls and other objects. In practice this would mean that a household needs to increase the number of access points in the house to get full coverage, but "leakage" of the network to the outside would be severely reduced. This makes potentially eavesdropping on the network increasingly complicated. Also, the 5GHz standard has (currently) 25 different non-overlapping channels (19 allowed in The Netherlands). This complicates the tracking even more.

4.1 Limitations and future research

As measurement windows were only two weeks long, we were unable to address the goal of extracting life patterns. To make this research more accurate we would recommend to extend the measuring period. This increases the visibility of recurring patterns in the resident's presence. These recurring patterns are what a potential burglar is probably searching for.

Another point for improvement would be to improve the reliability of the schedule maintained by the residents. In the current experiment, these schedules are kept in a table on paper, but this is prone to error and it relies completely on the precision of the residents filling in the forms. The schedules were also relatively coarse increasing the likely

Wi-Fi range also posed an issue. In multiple households, we saw problems with reception. In some households, residents claimed they often turn off their Wi-Fi and revert to mobile networks when they were upstairs because their network reception was poor. Additionally, the capturing range of the laptops was limited, which showed in some devices going in and out of range.

In some households, network strength issues were solved by placing additional wireless access points. Our test setup could only track one access point potentially missing information.

Apart from technical improvements, we would also advice a larger and more diverse set of test subjects. Ideally a randomly selected group of households to maximize the statistical usability of the results.

5. REFERENCES

[1] McGlaun, S. (2014, February 27). 4 Billion Wi-Fi enabled devices in use around the world today. Retrieved March 05, 2016, from http://www.tweaktown.com/news/35816/4-billion-wi-fi- enabled-devices-in-use-around-the-world-today/index.html [not peer reviewed]

[2] Burglary. (2011). Retrieved June 07, 2016, from https://www.fbi.gov/about-us/cjis/ucr/crime-in-the-u.s/2010/crime-in-the-u.s.-2010/property-crime/burglarymain [not peer reviewed]

[3] Beaton, A., Cook, M., Kavanagh, M., & Herrington, C. (2000). The psychological impact of burglary. Psychology, Crime & Law, 6(1), 33-43. doi:10.1080/10683160008410830

[4] Meer tweeverdieners met een voltijdbaan én een grote deeltijdbaan. (2015, January 29). Retrieved June 07, 2016, from https://www.cbs.nl/nl-nl/nieuws/2015/05/meer-tweeverdieners-met-een-voltijdbaan-en-een-grote-deeltijdbaan [not peer reviewed]

[5] Spelman, W., & Brown, D. K. (1981). Calling the police: Citizen reporting of serious crime. Washington, DC: Police Executive Research Forum.

[6] Mawby, R. I., & Walklate, S. (1997). The impact of burglary: a tale of two cities. International Review of Victimology, 4(4), 267-295.

[7] Verbree, E., Zlatanova, S., Van Winden, K., Van der Laan, E. B., Makri, A., Taizhou, L., & Haojun, A. (2013, December). To localise or to be localised with Wifi in the Hubei museum? In Acquisition and Modelling of Indoor and Enclosed Environments 2013, Cape Town, South Africa, 11-13 December 2013, ISPRS Archives Volume XL-4/W4, 2013. ISPRS.

[8] Geler, J. (2002, August 15). Understanding 802.11 Frame Types. Retrieved June 07, 2016, from http://www.wi-fiplanet.com/tutorials/article.php/1447501/Understanding-80211-Frame-Types.htm

[9] [11] Won, C., Youn, J., Ali, H., Sharif, H., & Deogun, J. (2005, September). Adaptive radio channel allocation for supporting coexistence of 802.15. 4 and 802.11 b. In *IEEE Vehicular Technology Conference* (Vol. 62, No. 4, p. 2522). IEEE; 1999.

[10] CBS StatLine - Huishoudens; samenstelling, grootte, regio, 1 januari. (2016, June 8). Retrieved June 07, 2016, from http://statline.cbs.nl/StatWeb/publication/?VW=T&DM=SLNL&PA=71486NED&D1=0-2,23-26&D2=0&D3=0,5-16&D4=(l-1)-l&HD=090402-0910&HDR=T,G3&STB=G1,G2 [not peer reviewed]

Appendix 2: Probe Proverif implementation

```
(* Network *)
free net.
private free beaconNet.
(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.
(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.
(* Hashing algorithm *)
fun hash/1.
(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).
(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.
(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.
(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.
(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.
(* security property: the attacker does not learn the client-SSID *)
        query attacker : clientMac.
        query attacker : apMac.
        query attacker : seqctl.
        query attacker : SSID.
        query attacker : beaconData.
        query attacker : probeRates.
let Client =
        new clientMac; (* Mac address *)
        new probeRates; (* Rates which the client support, need to atleast match network *)
        (* AP beacon, new element it included: the public key of the AP for the dh exchange *)
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        in ( beaconNet, ( apMac, seqctl, ( dhPubAP, =SSID, beaconData ) ));
        (* Randomly create a base Iv which we hash each time to get an unique IV everytime, furthermore create the enc key with DH exchange *)
        new baseIv;
        new dhPrivClient;
        let dhPubClient         = g( dhPrivClient ) in
        let dhKey                       = f( dhPrivClient, dhPubAP ) in
        let hashedDhKey         = hash(dhKey) in
        let encKey                      = sencrypt(hashedDhKey, PMK, baseIv) in
        (* probe req *)
        let eProbeMacReq        = sencrypt( clientMac, encKey, baseIv ) in
        let eApProbeMacReq      = sencrypt( apMac, encKey, baseIv ) in
        let eSeqctlReq                  = sencrypt( seqctl, encKey, baseIv ) in
        let eSSID                       = sencrypt( SSID, encKey, baseIv ) in
        let eProbeRates         = sencrypt( probeRates, encKey, baseIv ) in
        (* CLI->AP : probe request : da, sa, seqctl, body(SSID, rates) *)
        (* Include baseIV needed as IV for every transaction, furthermore share public DH client key *)
        out ( net, ( eProbeMacReq, eApProbeMacReq, eSeqctlReq, ( baseIv, dhPubClient, eSSID, eProbeRates ) ) );
        (* AP->CLI : probe response        : da, sa, bssid, seqctl, body(ssid, beaconData(Timestamp, Beacon interval, Cap info, SSID, FH, DS, CF, IBSS)) *)
        in ( net, ( eProbeMacResp, eApProbeMacResp, eSqctlResp, ( eSSIDResp, eBeaconDataResp )));
        let probeIvResp         = hash(baseIv) in
        let =apMac                      = sdecrypt( eApProbeMacResp, encKey, probeIvResp ) in
        let =clientMac                  = sdecrypt( eProbeMacResp, encKey, probeIvResp ) in
        let =seqctl             = sdecrypt( eSqctlResp, encKey, probeIvResp ) in
        let =SSID                       = sdecrypt( eSSIDResp, encKey, probeIvResp ) in
        let =beaconData         = sdecrypt( eBeaconDataResp, encKey, probeIvResp ) in
        (* CLI->AP : ack : da, seqctl *)
        let probeIvAck                          = hash(probeIvResp) in
        let eApProbeMacAck              = sencrypt( apMac, encKey, probeIvAck ) in
        let seqctlAck                           = sencrypt( seqctl, encKey, probeIvAck ) in
        out ( net, (eApProbeMacAck, seqctlAck) );
        new end.
```

```
let AccessPoint =
        new gtk;
        new beaconData;
        new apMac;
        new seqctl;
        new dhPrivAP;
        let dhPubAP = g( dhPrivAP ) in
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        out ( beaconNet, ( apMac, seqctl, ( dhPubAP, SSID, beaconData )));
        (* CLI->AP : probe request : da, sa, seqctl, body(SSID, rates) *)
        in ( net, ( eProbeMacReq, eApProbeMacReq, eSeqctlReq, ( baseIv, dhPubClient, eSSID, eProbeRates ) ));
        (* Construct key and check if this message is for us by decrypting apProbeMac & probeMac *)
        let dhKey                       = f(dhPrivAP, dhPubClient ) in
        let hashedDhKey         = hash(dhKey) in
        let encKey                      = sencrypt( hashedDhKey, PMK, baseIv) in
        let =apMac                      = sdecrypt( eApProbeMacReq, encKey, baseIv ) in
        let clientMac                   = sdecrypt( eProbeMacReq, encKey, baseIv ) in
        let =seqctl                     = sdecrypt( eSeqctlReq, encKey, baseIv ) in
        let =SSID                       = sdecrypt( eSSID, encKey, baseIv ) in
        let probeRates                  = sdecrypt( eProbeRates, encKey, baseIv ) in
        (* AP->CLI : probe response         : da, sa, bssid, seqctl, body(ssid, beaconData(Timestamp, Beacon interval, Cap info, SSID, FH, DS, CF, IBSS)) *)
        let probeIvResp         = hash(baseIv) in
        let eProbeMacResp       = sencrypt( clientMac, encKey, probeIvResp ) in
        let eApProbeMacResp = sencrypt( apMac, encKey, probeIvResp ) in
        let eSqctlResp                  = sencrypt( seqctl, encKey, probeIvResp ) in
        let eSSIDResp                   = sencrypt( SSID, encKey, probeIvResp ) in
        let eBeaconDataResp     = sencrypt( beaconData, encKey, probeIvResp ) in
        out ( net, ( eProbeMacResp, eApProbeMacResp, eSqctlResp, ( eSSIDResp, eBeaconDataResp )));
        (* CLI->AP : ack : da, seqctl *)
        in ( net, (eApProbeMacAck, eSeqctlAck) );
        let probeIvAck                          = hash( probeIvResp) in
        let =apMac                              = sencrypt( eApProbeMacAck, encKey, probeIvAck ) in
        let =seqctl                             = sdecrypt( eSeqctlAck, encKey, probeIvAck ) in
        new end.
process
        new key;
        new SSID;
        let PMK = pmkGen( SSID, key ) in
        ( !Client | !AccessPoint )
```

Appendix 3: Authentication & association Proverif encryption

```
(* Network *)
free net.
private free beaconNet.
(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.
(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.
(* Hashing algorithm *)
fun hash/1.
(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).
(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.
(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.
(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.
(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.
(* security property: the attacker does not learn the client-SSID *)
        query attacker : clientMac.
        query attacker : apMac.
        query attacker : seqctl.
        query attacker : SSID.
        query attacker : capRateData.
        query attacker : listenInterval.
        query attacker : assocId.
let Client =
        new seqctl;
        new capRateData;
        new listenInterval;
        (* Auth *)
        let authReqIv                   = hash(baseIv) in
        let eAuthReqMac         = sencrypt( clientMac, encKey, authReqIv ) in
        let eAuthReqApMac       = sencrypt( apMac, encKey, authReqIv ) in
        let eAuthReqSeqctl      = sencrypt( seqctl, encKey, authReqIv ) in
        new authParams;
        (* CLI->AP : Auth : (da=bssid, sa, seqctl, body(params)) *)
        out ( net, ( eAuthReqApMac, eAuthReqMac, eAuthReqSeqctl, authParams ));
        (* AP->CLI : Ack : (da, seqctl) *)
        in ( net, (eAuthReqAckMac, eAuthReqAckSeqctl) );
        let authReqAckIv        = hash(authReqIv) in
        let =clientMac                  = sdecrypt( eAuthReqAckMac, encKey, authReqAckIv ) in
        let =seqctl             = sdecrypt( eAuthReqAckSeqctl, encKey, authReqAckIv ) in
        (* AP->CLI : Auth : (da, sa=bssid, seqctl, body(params)) *)
        in ( net, ( eAuthRespMac, eAuthRespApMac, eAuthRespSeqctl, authApParams ));
        let authRespIv                  = hash(authReqAckIv) in
        let =apMac                      = sdecrypt( eAuthRespApMac, encKey, authRespIv ) in
        let =clientMac                  = sdecrypt( eAuthRespMac, encKey, authRespIv ) in
        let =seqctl             = sdecrypt( eAuthRespSeqctl, encKey, authRespIv ) in
        let authRespAckIv               = hash(authRespIv) in
        let eAuthRespAckApMac           = sencrypt( apMac, encKey, authRespAckIv ) in
        let eAuthRespAckSeqctl  = sencrypt( seqctl, encKey, authRespAckIv ) in
        (* CLI->AP : Ack : (da, seqctl) *)
        out ( net, ( eAuthRespAckApMac, eAuthRespAckSeqctl ) );
        (* assoc *)
        let assocReqIv                                          = hash(authRespAckIv) in
        let eAssocReqMac                        = sencrypt( clientMac, encKey, assocReqIv ) in
        let eAssocReqApMac                      = sencrypt( apMac, encKey, assocReqIv ) in
        let eAssocReqSeqctl                     = sencrypt( seqctl, encKey, assocReqIv ) in
        let eAssocReqCapRateData        = sencrypt( capRateData, encKey, assocReqIv ) in
        let eAssocReqListenInterval     = sencrypt( listenInterval, encKey, assocReqIv ) in
        let eAssocReqSSID                       = sencrypt( SSID, encKey, assocReqIv ) in
        (* CLI->AP : Assoc request : (da=bssid, sa, seqctl, body(cap info, listen interval, ssid, rates)) *)
        out ( net, ( eAssocReqApMac, eAssocReqMac, eAssocReqSeqctl, ( eAssocReqCapRateData, eAssocReqListenInterval, eAssocReqSSID ) ));
        (* AP->CLI : Ack : (da, seqctl) *)
        in ( net, (eAssocReqAckMac, eAssocReqAckSeqctl));
        let assocReqAckIv       = hash(assocReqIv) in
        let =clientMac                  = sdecrypt( eAssocReqAckMac, encKey, assocReqAckIv ) in
        let =seqctl             = sdecrypt( eAssocReqAckSeqctl, encKey, assocReqAckIv ) in
```

```
        (* AP->CLI : Assoc response : (da, sa=bssid, seqctl, body(cap info, status code, assoc id, supported rates)) *)
        in ( net, ( eAssocRespMac, eAssocRespApMac, eAssocRespSeqctl, ( eAssocRespCapRateData, eAssocRespAssocId, statusCode ) ));
        let assocResplv                 = hash(assocReqAcklv) in
        let =apMac                              = sdecrypt( eAssocRespApMac, encKey, assocResplv ) in
        let =clientMac                          = sdecrypt( eAssocRespMac, encKey, assocResplv ) in
        let =seqctl                     = sdecrypt( eAssocRespSeqctl, encKey, assocResplv ) in
        let =capRateData                = sdecrypt( eAssocRespCapRateData, encKey, assocResplv ) in
        let assocId                     = sdecrypt( eAssocRespAssocId, encKey, assocResplv ) in
        let assocRespAcklv              = hash(assocResplv) in
        let eAssocRespAckApMac          = sencrypt( apMac, encKey, assocRespAcklv ) in
        let eAssocRespAckSeqctl = sencrypt( seqctl, encKey, assocRespAcklv ) in
        (* CLI->AP : Ack : (da, seqctl) *)
        out ( net, ( eAssocRespAckApMac, eAssocRespAckSeqctl ) );
        new end.
let AccessPoint =
        new assocId;
        new statusCode;
        (* CLI->AP : Auth : (da=bssid, sa, seqctl, body(params)) *)
        in ( net, ( eAuthReqApMac, eAuthReqMac, eSeqctlAuthReq, authParams ));
        let authReqlv                   = hash(baselv) in
        let =apMac                      = sdecrypt( eAuthReqApMac, encKey, authReqlv ) in
        let =clientMac                  = sdecrypt( eAuthReqMac, encKey, authReqlv ) in
        let seqctl                      = sdecrypt( eSeqctlAuthReq, encKey, authReqlv ) in
        let authReqAcklv                = hash(authReqlv) in
        let eAuthReqAckMac              = sencrypt( clientMac, encKey, authReqAcklv ) in
        let eSeqctlAuthReqAck   = sencrypt( seqctl, encKey, authReqAcklv ) in
        (* AP->CLI : Ack : (da, seqctl) *)
        out ( net, (eAuthReqAckMac, eSeqctlAuthReqAck) );
        new authApParams;
        let authResplv                  = hash(authReqAcklv) in
        let eAuthRespMac        = sencrypt( clientMac, encKey, authResplv ) in
        let eAuthRespApMac      = sencrypt( apMac, encKey, authResplv ) in
        let eSeqctlAuthResp     = sencrypt( seqctl, encKey, authResplv ) in
        (* AP->CLI : Auth : (da, sa=bssid, seqctl, body(params)) *)
        out ( net, ( eAuthRespMac, eAuthRespApMac, eSeqctlAuthResp, authApParams ));
        (* CLI->AP : Ack : (da, seqctl) *)
        in ( net, ( eAuthRespAckApMac, eSeqctlAuthRespAck ) );
        let authRespAcklv       = hash(authResplv) in
        let =apMac                      = sdecrypt( eAuthRespAckApMac, encKey, authRespAcklv ) in
        let =seqctl             = sdecrypt( eSeqctlAuthRespAck, encKey, authRespAcklv ) in
        (* assoc *)
        (* CLI->AP : Assoc request : (da=bssid, sa, seqctl, body(cap info, listen interval, ssid, rates)) *)
        in ( net, ( eAssocReqApMac, eAssocReqMac, eAssocReqSeqctl, ( eAssocReqCapRateData, eAssocReqListenInterval, eAssocReqSSID ) ) );
        let assocReqlv                  = hash(authRespAcklv) in
        let =apMac                      = sdecrypt( eAssocReqApMac, encKey, assocReqlv ) in
        let =clientMac                  = sdecrypt( eAssocReqMac, encKey, assocReqlv ) in
        let =seqctl             = sdecrypt( eAssocReqSeqctl, encKey, assocReqlv ) in
        let capRateData         = sdecrypt( eAssocReqCapRateData, encKey, assocReqlv ) in
        let listenInterval      = sdecrypt( eAssocReqListenInterval, encKey, assocReqlv ) in
        let =SSID                       = sdecrypt( eAssocReqSSID, encKey, assocReqlv ) in
        let assocReqAcklv               = hash(assocReqlv) in
        let eAssocReqAckMac     = sencrypt( clientMac, encKey, assocReqAcklv ) in
        let eAssocReqAckSeqctl  = sencrypt( seqctl, encKey, assocReqAcklv ) in
        (* AP->CLI : Ack : (da, seqctl) *)
        out ( net, (eAssocReqAckMac, eAssocReqAckSeqctl));
        let assocResplv                 = hash(assocReqAcklv) in
        let eAssocRespMac                       = sencrypt( clientMac, encKey, assocResplv ) in
        let eAssocRespApMac             = sencrypt( apMac, encKey, assocResplv ) in
        let eAssocRespSeqctl            = sencrypt( seqctl, encKey, assocResplv ) in
        let eAssocRespCapRateData       = sencrypt( capRateData, encKey, assocResplv ) in
        let eAssocRespAssocId           = sencrypt( assocId, encKey, assocResplv ) in
        (* AP->CLI : Assoc response : (da, sa=bssid, seqctl, body(cap info, status code, assoc id, supported rates)) *)
        out ( net, ( eAssocRespMac, eAssocRespApMac, eAssocRespSeqctl, ( eAssocRespCapRateData, eAssocRespAssocId, statusCode ) ));
        (* CLI->AP : Ack : (da, seqctl) *)
        in ( net, ( eAssocRespAckApMac, eAssocRespAckSeqctl ) );
        let assocRespAcklv              = hash(assocResplv) in
        let =apMac                              = sdecrypt( eAssocRespAckApMac, encKey, assocRespAcklv ) in
        let =seqctl                     = sdecrypt( eAssocRespAckSeqctl, encKey, assocRespAcklv ) in
        new end.
process
        new key;
        new SSID;
        new apMac;
        new clientMac;
        new dhKey;
        new baselv;
        let PMK                         = pmkGen( SSID, key ) in
        let hashedDhKey         = hash(dhKey) in
        let encKey                      = sencrypt(hashedDhKey, PMK, baselv) in
        ( !Client | !AccessPoint )
```

Appendix 4: WPA key exchange Proverif implementation

```
(* Network *)
free net.
private free beaconNet.
(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.
(* Symmetric key cryptography *)
fun senc/2.
reduc sdec(senc(x,y),y) = x.
(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.
(* Hashing algorithm *)
fun hash/1.
(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).
(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.
(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.
(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.
(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.
(* security property: the attacker does not learn the client-SSID *)
        query attacker : clientMac.
        query attacker : apMac.
        query attacker : seqctl.
        query attacker : M1Data.
        query attacker : M2Data.
        query attacker : M3Data.
        query attacker : M4Data.
(* In this example i have abstracted all fields in the key frame away into a KeyData variable, *)
(* this is done because if one variable could be send without problems more can be send using the same method of encryption. *)
(* all data that is required to calculate keys is send in seperate variables *)
let Client =
        new sNonce;
        new M2Data;
        new M4Data;
        (* key exchange *)
        (* AP->CLI : M1 : (da, sa=bssid, seqctl, body(data, anonce)       *)
        in ( net, ( eM1ApMac, eM1Mac, eM1Sqctl, ( eDataM1, aNonce ) ));
        let M1Iv                        = hash(baseIv) in
        let =apMac                      = sdecrypt( eM1ApMac, encKey, M1Iv ) in
        let =clientMac                  = sdecrypt( eM1Mac, encKey, M1Iv ) in
        let seqctl                      = sdecrypt( eM1Sqctl, encKey, M1Iv ) in
        let M1Data                      = sdecrypt( eDataM1, encKey, M1Iv ) in
        let M1AckIv             = hash(M1Iv) in
        let eM1AckApMac                 = sencrypt( apMac, encKey, M1AckIv ) in
        let eM1AckSeqctl        = sencrypt( seqctl, encKey, M1AckIv ) in
        (* CLI->AP : Ack : (da, seqctl) *)
        out ( net, (eM1AckApMac, eM1AckSeqctl) );
        (* Calculate all keys *)
        let PTK = ptkGen( PMK, aNonce, sNonce, clientMac, apMac ) in
        let kck = kckGen( PTK ) in
        let kek = kekGen( PTK ) in
        let tk = tkGen(PTK) in
        let M2Iv                        = hash(M1AckIv) in
        let eM2ApMac                    = sencrypt( apMac, encKey, M2Iv ) in
        let eM2Mac                      = sencrypt( clientMac, encKey, M2Iv ) in
        let eM2Seqctl                   = sencrypt( seqctl, encKey, M2Iv ) in
        let eM2Data                     = sencrypt( M2Data, encKey, M2Iv ) in
        let mM2data                     = ( eM2Data, sNonce ) in
        let M2mic                       = sign( mM2data, kck ) in
        (* CLI->AP : M2 : (da=bssid, sa, seqctl, body(data, snonce, mic)) *)
        out ( net, ( eM2ApMac, eM2Mac, eM2Seqctl, ( mM2data, M2mic ) ));
        (* AP->CLI : Ack : (da, seqctl) *)
        in ( net, ( eM2AckMac, eM2AckSqctl) );
        let M2AckIv                     = hash(M2Iv) in
        let =clientMac                  = sdecrypt( eM2AckMac, encKey, M2AckIv ) in
        let =seqctl                     = sdecrypt( eM2AckSqctl, encKey, M2AckIv ) in
```

```
		(* AP->CLI : M3 : (da, sa=bssid, seqctl, body(data, anonce, mic, gtk)) *)
		in ( net, ( eM3Mac, eM3ApMac, eM3Seqctl, ( mM3data, M3Mic )));
		let M3Iv 			= hash(M2AckIv) in
		let =apMac 			= sdecrypt( eM3ApMac, encKey, M3Iv ) in
		let =clientMac 			= sdecrypt( eM3Mac, encKey, M3Iv ) in
		let =seqctl 			= sdecrypt( eM3Seqctl, encKey, M3Iv ) in
		let =M3Mic 			= sign( mM3data, kck ) in
		let ( eM3Data, =aNonce, encGtk ) = mM3data in
		let M3Data 			= sdecrypt( eM3Data, encKey, M3Iv ) in
		let gtk 			= sdec( encGtk, kek ) in
		let M3AckIv 			= hash(M3Iv) in
		let eM3AckApMac 		= sencrypt( apMac, encKey, M3AckIv ) in
		let eM3AckSeqctl 		= sencrypt( seqctl, encKey, M3AckIv ) in
		(* CLI->AP : Ack : (da, seqctl) *)
		out ( net, (eM3AckApMac, eM3AckSeqctl) );
		let M4Iv 			= hash(M3AckIv) in
		let eM4ApMac 			= sencrypt( apMac, encKey, M4Iv ) in
		let eM4Mac 			= sencrypt( clientMac, encKey, M4Iv ) in
		let eM4Seqctl 			= sencrypt( seqctl, encKey, M4Iv ) in
		let eM4Data 			= sencrypt( M4Data, encKey, M4Iv ) in
		let M4mic 			= sign( eM4Data, kck ) in
		(* CLI->AP : M4 		: (da=bssid, sa, seqctl, body(data, mic)) *)
		out ( net, ( eM4ApMac, eM4Mac, eM4Seqctl, ( eM4Data, M4mic )));
		(* AP->CLI : Ack : (da, seqctl) *)
		in ( net, ( eM4AckMac, eM4AckSqctl) );
		let M4AckIv 			= hash(M4Iv) in
		let =clientMac 			= sdecrypt( eM4AckMac, encKey, M4AckIv ) in
		let =seqctl 			= sdecrypt( eM4AckSqctl, encKey, M4AckIv ) in
		new end.
let AccessPoint =
		new gtk;
		new M1Data;
		new M3Data;
		new seqctl;
		new aNonce;
		(* key exchange *)
		let M1Iv 			= hash(baseIv) in
		let eM1ApMac 			= sencrypt( apMac, encKey, M1Iv ) in
		let eM1Mac 				= sencrypt( clientMac, encKey, M1Iv ) in
		let eM1Sqctl 			= sencrypt( seqctl, encKey, M1Iv ) in
		let KeyDataM1 			= sencrypt( M1Data, encKey, M1Iv ) in
		(* AP->CLI : M1 : (da, sa=bssid, seqctl, body(data, anonce)	*)
		out ( net, ( eM1ApMac, eM1Mac, eM1Sqctl, ( KeyDataM1, aNonce )));
		(* CLI->AP : Ack : (da, seqctl) *)
		in ( net, (eM1AckApMac, eM1AckSeqctl) );
		let M1AckIv 			= hash(M1Iv) in
		let =apMac 			= sdecrypt( eM1AckApMac, encKey, M1AckIv ) in
		let =seqctl 			= sdecrypt( eM1AckSeqctl, encKey, M1AckIv ) in
		(* CLI->AP : M2 : (da=bssid, sa, seqctl, body(data, snonce, mic)) *)
		in ( net, ( eM2ApMac, eM2Mac, eM2Seqctl, ( mM2data, M2mic )));
		let M2Iv 				= hash(M1AckIv) in
		let =apMac 				= sdecrypt( eM2ApMac, encKey, M2Iv ) in
		let =clientMac 				= sdecrypt( eM2Mac, encKey, M2Iv ) in
		let =seqctl 				= sdecrypt( eM2Seqctl, encKey, M2Iv ) in
		let ( eM2Data, sNonce ) = mM2data in
		let M2Data 				= sdecrypt( eM2Data, encKey, M2Iv ) in
		(* Calculate all keys *)
		let PTK = ptkGen( PMK, aNonce, sNonce, clientMac, apMac ) in
		let kck = kckGen( PTK ) in
		let kek = kekGen( PTK ) in
		let tk = tkGen(PTK) in
		(* Check m2 mic *)
		let =M2mic 				= sign( mM2data, kck ) in
		let M2AckIv 		= hash(M2Iv) in
		let eM2AckMac 			= sencrypt( clientMac, encKey, M2AckIv ) in
		let eM2AckSqctl 	= sencrypt( seqctl, encKey, M2AckIv ) in
		(* AP->CLI : Ack : (da, seqctl) *)
		out ( net, ( eM2AckMac, eM2AckSqctl) );
		let M3Iv 			= hash(M2AckIv) in
		let eM3ApMac 			= sencrypt( apMac, encKey, M3Iv ) in
		let eM3Mac 				= sencrypt( clientMac, encKey, M3Iv ) in
		let eM3Seqctl 			= sencrypt( seqctl, encKey, M3Iv ) in
		let eM3Data 			= sencrypt( M3Data, encKey, M3Iv ) in
		let encGtk 			= senc( gtk , kek ) in
		let mM3data 			= ( eM3Data, aNonce, encGtk ) in
		let M3Mic 			= sign( mM3data, kck ) in
		(* AP->CLI : M3 : (da, sa=bssid, seqctl, body(data, anonce, mic, gtk)) *)
		out ( net, ( eM3Mac, eM3ApMac, eM3Seqctl, ( mM3data, M3Mic )));
		(* CLI->AP : Ack : (da, seqctl) *)
		in ( net, (eM3AckApMac, eM3AckSeqctl) );
		let M3AckIv 		= hash(M3Iv) in
		let =apMac 			= sdecrypt( eM3AckApMac, encKey, M3AckIv ) in
```

```
        let =seqctl                 = sdecrypt( eM3AckSeqctl, encKey, M3AckIv ) in
        (* CLI->AP : M4             : (da=bssid, sa, seqctl, body(data, mic)) *)
        in ( net, ( eM4ApMac, eM4Mac, eM4Seqctl, ( eM4Data, M4mic )));
        let M4Iv                            = hash(M3AckIv) in
        let =apMac                          = sdecrypt( eM4ApMac, encKey, M4Iv ) in
        let =clientMac                      = sdecrypt( eM4Mac, encKey, M4Iv ) in
        let =seqctl                 = sdecrypt( eM4Seqctl, encKey, M4Iv ) in
        let M4Data                          = sdecrypt( eM4Data, encKey, M4Iv ) in
        let =M4mic                          = sign( eM4Data, kck ) in
        (* AP->CLI : Ack : (da, seqctl) *)
        let M4AckIv                 = hash(M4Iv) in
        let eM4AckMac                       = sencrypt( clientMac, encKey, M4AckIv ) in
        let eM4AckSqctl             = sencrypt( seqctl, encKey, M4AckIv ) in
        out ( net, ( eM4AckMac, eM4AckSqctl) );
        new end.
process
        new key;
        new SSID;
        new apMac;
        new clientMac;
        new dhKey;
        new baseIv;
        let PMK                             = pmkGen( SSID, key ) in
        let hashedDhKey             = hash(dhKey) in
        let encKey                          = sencrypt(hashedDhKey, PMK, baseIv) in
        ( !Client | !AccessPoint )
```

Appendix 5: Multicast data transmission Proverif implementation

```
(* Network *)
free net.
private free beaconNet.

(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.

(* Symmetric key cryptography *)
fun senc/2.
reduc sdec(senc(x,y),y) = x.

(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.

(* Hashing algorithm *)
fun hash/1.

(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).

(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.

(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.

(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.

(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.

(* security property: the attacker does not learn the client-SSID *)
        query attacker : client1Mac.
        query attacker : client2Mac.
        query attacker : apMac.
        query attacker : seqctl.
        query attacker : keyId.
        query attacker : packetNumber.

let Client1 =
        (* Content of CCMP header *)
        new keyId;
        new packetNumber;

        (* AP beacon, new element it included: the public key of the AP for the dh exchange *)
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        in ( beaconNet, ( =apMac, seqctl, ( timestamp, dhPubAP, =SSID, beaconData )));

        (* CLI->AP : Data
        (
                framecontrol, sa, bssid, seqControl,
                body(
                        CCMPheader(KeyId, PacketNumber),
                        encrypted((data, mic), TK)
                )
        )       *)
        let dataIv              = hash(timestamp) in
        let eData1Mac           = sencrypt( client1Mac, gtk, dataIv ) in
        let eData2Mac           = sencrypt( client2Mac, gtk, dataIv ) in
        let eDataSeqctl         = sencrypt( seqctl, gtk, dataIv ) in

        new framecontrol;
        new dataPacket;
        let nonce = ( packetNumber, client1Mac ) in
        let aad = ( framecontrol, client2Mac, client1Mac, seqctl ) in
        let mic = sign( ( tk, nonce, aad, dataPacket ), kck ) in
        let CCMPHeader = sencrypt( ( keyId, packetNumber ), gtk, dataIv ) in
        let body = (CCMPHeader, senc(( dataPacket, mic ), tk )) in
        out ( net, ( framecontrol, eData1Mac, eData2Mac, eDataSeqctl, body ) );
```

```
		in ( net, ( eClient1Mac, eDataAckSeqctl ) );
		let dataAckIv 			= hash(dataIv) in
		let =client1Mac 		= sdecrypt( eClient1Mac, gtk, dataAckIv ) in
		let =seqctl 		= sdecrypt( eDataAckSeqctl, gtk, dataAckIv ) in
		new end.
let Client2 =
		(* AP beacon, new element it included: the public key of the AP for the dh exchange *)
		(* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
		in ( beaconNet, ( =apMac, seqctl, ( timestamp, dhPubAP, =SSID, beaconData )));
		in ( net, ( framecontrol, eData1Mac, eData2Mac, eDataSeqctl, body ) );
		let dataIv 		= hash(timestamp) in
		let =client1Mac 		= sdecrypt( eData1Mac, gtk, dataIv ) in
		let =client2Mac 	= sdecrypt( eData2Mac, gtk, dataIv ) in
		let =seqctl 		= sdecrypt( eDataSeqctl, gtk, dataIv ) in
		let (CCMPHeader, rData) = body in
		let ( keyId, packetNumber ) = sdecrypt( CCMPHeader, gtk, dataIv ) in
		let ( dataPacket, mic ) = sdec(rData, tk) in
		let nonce = ( packetNumber, client1Mac ) in
		let aad = ( framecontrol, client2Mac, client1Mac, seqctl ) in
		let micCheck = sign( ( tk, nonce, aad, dataPacket ), kck ) in
		if mic = micCheck then
		(* CLI->AP : Data (framecontrol, sa, bssid, seqControl, body( CCMPheader(KeyId, PacketNumber), encrypted((data, mic), TK) )) *)
		let ackIv 			= hash(timestamp) in
		let eClient1Mac 		= sencrypt( eData1Mac, gtk, dataIv ) in
		let eDataAckSeqctl 	= sencrypt( seqctl, gtk, dataIv ) in
		out ( net, ( eClient1Mac, eDataAckSeqctl ) );
		new end.
let AccessPoint =
		new seqctl;
		new beaconData;
		new timestamp;
		new dhPrivAP;
		let dhPubAP = g( dhPrivAP ) in
		(* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
		out ( beaconNet, ( apMac, seqctl, ( timestamp, dhPubAP, SSID, beaconData )));
		new end.
process
		new key;
		new SSID;
		new apMac;
		new client1Mac;
		new client2Mac;
		new dhKey;
		new aNonce;
		new sNonce;
		new gtk;
		let PMK 			= pmkGen( SSID, key ) in
		let PTK 			= ptkGen( PMK, aNonce, sNonce, client1Mac, apMac ) in
		let kck 			= kckGen( PTK ) in
		let kek 			= kekGen( PTK ) in
		let tk 				= tkGen(PTK) in
		( !Client1 | !Client2 | !AccessPoint )
```

Appendix 6: Beacon Proverif implementation

```
(* Network *)
free net.
(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.
(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.
(* Hashing algorithm *)
fun hash/1.
(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).
(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.
(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.
(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.
(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.
(* security property: the attacker does not learn the client-SSID *)
        query attacker : ClientBitA.
        query attacker : ClientBitB.
let ClientA =
        (* AP beacon, new element it included: the public key of the AP for the dh exchange *)
        (* Furthermore is the TIM partly encrypted with each clients key depending if they support it or not *)
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        in ( net, ( =apMac, seqctl, ( timestamp, dhPubAP, =SSID, beaconData, TIM )));
        let (eClientBitA, ClientBitB) = TIM in
        let beaconIv                    = hash(timestamp) in
        let ClientBitA                  = sdecrypt( eClientBitA, encKeyA, beaconIv ) in
        new end.
let ClientB =
        (* AP beacon, new element it included: the public key of the AP for the dh exchange *)
        (* Furthermore is the TIM partly encrypted with each clients key depending if they support it or not *)
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        in ( net, ( =apMac, seqctl, ( timestamp, dhPubAP, =SSID, beaconData, TIM )));
        let (eClientBitA, ClientBitB) = TIM in
        new end.
let AccessPoint =
        new gtk;
        new beaconData;
        new seqctl;
        new dhPrivAP;
        let dhPubAP = g( dhPrivAP ) in
        new timestamp;
        (* Bits that indicate if a client has a message or not *)
        new ClientBitA;
        new ClientBitB;
        let beaconIv                    = hash(timestamp) in
        let eClientBitA         = sencrypt( ClientBitA, encKeyA, beaconIv ) in
        let TIM = (eClientBitA, ClientBitB) in
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        out ( net, ( apMac, seqctl, ( timestamp, dhPubAP, SSID, beaconData, TIM )));
        new end.
process
        new key; new SSID; new apMac; new clientMac; new dhKeyA;
        new dhKeyB; new baseIv; new aNonce; new sNonce;
        let PMK                         = pmkGen( SSID, key ) in
        let hashedDhKeyA        = hash(dhKeyA) in
        let encKeyA             = sencrypt(hashedDhKeyA, PMK, baseIv) in
        let hashedDhKeyB        = hash(dhKeyB) in
        let encKeyB             = sencrypt(hashedDhKeyB, PMK, baseIv) in
        let PTK                         = ptkGen( PMK, aNonce, sNonce, clientMac, apMac ) in
        let kck                         = kckGen( PTK ) in
        let kek                         = kekGen( PTK ) in
        let tk                                  = tkGen(PTK) in
        ( !ClientA | !AccessPoint )
```

Appendix 7: Power management Proverif implementation

```
(* Network *)
free net.
private free beaconNet.
(* Symmetric key cryptography with IV *)
fun sencrypt/3.
reduc sdecrypt(sencrypt(x,y,z),y,z) = x.
(* Symmetric key cryptography *)
fun senc/2.
reduc sdec(senc(x,y),y) = x.
(* A-Symmetric key cryptography *)
fun pk/1.
fun aencrypt/2.
reduc adecrypt(aencrypt(x,pk(y)),y) = x.
(* Hashing algorithm *)
fun hash/1.
(* Diffie-Hellman *)
fun f/2.
fun g/1.
equation f(x,g(y)) = f(y,g(x)).
(* Signing *)
fun sign/2.
reduc checksign( sign(m, sk), pk(sk) ) = m.
(* Generate PMK(256bits) from ssid & key *)
fun pmkGen/2.
(* Generate PTK from PMK, ANonce, SNonce, AAid(ap ssid), SPAid(client ssid) *)
fun ptkGen/5.
(* Derive the other keys from the 512bits PTK: 4x128bits: EAPOL-KCK, EAPOL-KEK, TKIP-TK, TKIP-MIC key *)
(* Derive the other keys from the 384bits PTK: 3x128bits: EAPOL-KCK, EAPOL-KEK, CCMP-TK *)
fun kckGen/1.
fun kekGen/1.
fun micGen/1.
fun tkGen/1.
(* security property: the attacker does not learn the client-SSID *)
        query attacker : clientMac.
        query attacker : apMac.
        query attacker : seqctl.
        query attacker : SSID.
        query attacker : beaconData.
        query attacker : assocId.
        query attacker : clientBit.
        query attacker : pwrmgmt.
        query attacker : moredata.
let Client =
        new pwrmgmt;
        new moredata;
        (* AP beacon, new element it included: the public key of the AP for the dh exchange *)
        (* Furthermore is the TIM partly encrypted with each clients key depending if they support it or not *)
        (* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
        in ( beaconNet, ( =apMac, seqctl, ( timestamp, dhPubAP, =SSID, beaconData, TIM )));
        let (eClientBit) = TIM in
        let beaconIv                    = hash(timestamp) in
        let clientBit           = sdecrypt( eClientBit, encKey, beaconIv ) in
        (* Request buffered frame from ap with PS-Poll frame encapsulated in data frame *)
        (* CLI->AP : Data : body(AID) *)
        (* Content of CCMP header *)
        new keyId;
        new packetNumber;
        (* CLI->AP : Data (framecontrol, da, sa, bssid, seqControl, body(CCMPheader(KeyId, PacketNumber), encrypted((data, mic), TK))) *)
        let dataReqIv                   = hash(baseIv) in
        let eDataMac            = sencrypt( clientMac, encKey, dataReqIv ) in
        let eDataApMac          = sencrypt( apMac, encKey, dataReqIv ) in
        let eDataSeqctl         = sencrypt( seqctl, encKey, dataReqIv ) in
        new framecontrolData;
        let ePwrmgmt            = sencrypt( pwrmgmt, encKey, dataReqIv ) in
        let eMoredata           = sencrypt( moredata, encKey, dataReqIv ) in
        let framecontrol = (framecontrolData, ePwrmgmt, eMoredata) in
        let dataReqPacket = assocId in
        let nonce = ( packetNumber, clientMac ) in
        let aad = ( framecontrol, apMac, clientMac, seqctl ) in
        let mic = sign( ( tk, nonce, aad, dataReqPacket ), kck ) in
        let CCMPHeader = sencrypt( ( keyId, packetNumber ), encKey, dataReqIv ) in
        let body = (CCMPHeader, senc(( dataReqPacket, mic ), tk )) in
        out ( net, ( framecontrol, eDataApMac, eDataMac, eDataSeqctl, body ) );
        in ( net, ( framecontrolResp, eDataRespApMac, eDataRespMac, eDataRespSeqctl, bodyResp ) );
        let dataRespIv                  = hash(dataReqIv) in
        let =apMac                      = sdecrypt( eDataRespApMac, encKey, dataRespIv ) in
        let =clientMac                  = sdecrypt( eDataRespMac, encKey, dataRespIv ) in
        let =seqctl             = sdecrypt( eDataRespSeqctl, encKey, dataRespIv ) in
        let ( =framecontrolData, ePwrmgmtResp, eMoredataResp) = framecontrol in
        let =pwrmgmt                    = sdecrypt( ePwrmgmtResp, encKey, dataRespIv ) in
```

```
		let =moredata			= sdecrypt( eMoredataResp, encKey, dataResplv ) in
		let (CCMPHeaderResp, rDataResp) = bodyResp in
		let ( keyIdResp, packetNumberResp ) = sdecrypt( CCMPHeader, encKey, dataResplv ) in
		let ( dataRespPacket, micResp ) = sdec(rDataResp, tk) in
		let nonceResp = ( packetNumberResp, clientMac ) in
		let aadResp = ( framecontrolResp, apMac, clientMac, seqctl ) in
		let micCheckResp = sign( ( tk, nonceResp, aadResp, dataRespPacket ), kck ) in
		if micResp = micCheckResp then
		let dataAcklv			= hash(dataResplv) in
		let dataAckMac			= sencrypt( clientMac, encKey, dataResplv ) in
		let dataAckSeqctl	= sencrypt( seqctl, encKey, dataResplv ) in
		(* AP->CLI : Ack : (da, seqctl) *)
		out ( net, (dataAckMac, dataAckSeqctl));
		new end.
let AccessPoint =
		new gtk;
		new beaconData;
		new seqctl;
		new dhPrivAP;
		let dhPubAP = g( dhPrivAP ) in
		new timestamp;
		(* Bits that indicate if a client has a message or not *)
		new clientBit;
		let beaconlv			= hash(timestamp) in
		let eClientBit			= sencrypt( clientBit, encKey, beaconlv ) in
		let TIM = (eClientBit) in
		(* AP->CLI : beacon : sa=bssid, seqctl, body(ssid + beaconData(Timestamp, Beacon interval, cap info, SSID, FH, DS, CF, IBSS, TIM)) *)
		out ( beaconNet, ( apMac, seqctl, ( timestamp, dhPubAP, SSID, beaconData, TIM )));
		in ( net, ( framecontrol, eDataApMac, eDataMac, eDataSeqctl, body ) );
		let dataReqlv			= hash(baselv) in
		let =apMac			= sdecrypt( eDataApMac, encKey, dataReqlv ) in
		let =clientMac			= sdecrypt( eDataMac, encKey, dataReqlv ) in
		let =seqctl		= sdecrypt( eDataSeqctl, encKey, dataReqlv ) in
		let ( framecontrolData, ePwrmgmt, eMoredata) = framecontrol in
		let pwrmgmt			= sdecrypt( ePwrmgmt, encKey, dataReqlv ) in
		let moredata			= sdecrypt( eMoredata, encKey, dataReqlv ) in
		let (CCMPHeader, rData) = body in
		let ( keyId, packetNumber ) = sdecrypt( CCMPHeader, encKey, dataReqlv ) in
		let ( dataReqPacket, mic ) = sdec(rData, tk) in
		let nonce = ( packetNumber, clientMac ) in
		let aad = ( framecontrol, apMac, clientMac, seqctl ) in
		let micCheck = sign( ( tk, nonce, aad, dataReqPacket ), kck ) in
		if mic = micCheck then
		if dataReqPacket = assocId then
		(* Content of CCMP header *)
		new keyIdResp;
		new packetNumberResp;
		(* CLI->AP : Data ( framecontrol, da, sa, bssid, seqControl,  body( CCMPheader(KeyId, PacketNumber),  encrypted((data, mic), TK))) *)
		let dataResplv		= hash(dataReqlv) in
		let eDataRespMac		= sencrypt( clientMac, encKey, dataResplv ) in
		let eDataRespApMac	= sencrypt( apMac, encKey, dataResplv ) in
		let eDataRespSeqctl	= sencrypt( seqctl, encKey, dataResplv ) in
		let ePwrmgmtResp		= sencrypt( pwrmgmt, encKey, dataResplv ) in
		let eMoredataResp		= sencrypt( moredata, encKey, dataResplv ) in
		let framecontrol = (framecontrolData, ePwrmgmtResp, eMoredataResp) in
		new dataPacketResp;
		let nonceResp = ( packetNumberResp, clientMac ) in
		let aadResp = ( framecontrolResp, apMac, clientMac, seqctl ) in
		let micResp = sign( ( tk, nonceResp, aadResp, dataPacketResp ), kck ) in
		let CCMPHeaderResp = sencrypt( ( keyId, packetNumberResp ), encKey, dataResplv ) in
		let bodyResp = (CCMPHeader, senc(( dataPacketResp, mic ), tk )) in
		out ( net, ( framecontrolResp, eDataRespApMac, eDataRespMac, eDataRespSeqctl, bodyResp ) );
		(* CLI->AP : Ack : (da, seqctl) *)
		in ( net, ( eDataAckApMac, eDataAckSqlctl ) );
		let dataAcklv		= hash(dataResplv) in
		let =apMac			= sdecrypt( eDataAckApMac, encKey, dataAcklv ) in
		let =seqctl		= sdecrypt( eDataAckSqlctl, encKey, dataAcklv ) in
		new end.
process
		new key; new SSID; new apMac; new clientMac; new dhKey;
		new baselv; new aNonce; new sNonce; new assocId;
		let PMK				= pmkGen( SSID, key ) in
		let hashedDhKey		= hash(dhKey) in
		let encKey			= sencrypt(hashedDhKey, PMK, baselv) in
		let PTK				= ptkGen( PMK, aNonce, sNonce, clientMac, apMac ) in
		let kck				= kckGen( PTK ) in
		let kek				= kekGen( PTK ) in
		let tk					= tkGen(PTK) in
		( !Client | !AccessPoint )
```