NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF INFORMATION SECURITY AND COMMUNICATION TECHNOLOGY

Lab. Report - Group 5

TTM4137 - Wireless Network Security

Submitted by

Alice Giraud - alicejg@stud.ntnu.no
Simon Martin Ott - simonmo@stud.ntnu.no
Renaldas Skarnulis - renaldas@stud.ntnu.no
Vittorio Triassi - vittorit@stud.ntnu.no

October 27, 2019

Introduction

This report describes procedures, results and theoretical background of the Wireless Network Security Lab TTM4137. The lab is divided in two parts, WiFi security and mobile network security. In the WiFi security part, we will analyze and break into a WEP and a WPA-PSK network in order to gain an understanding of the general weaknesses of WEP and the weaknesses of WPA under certain circumstances such as weak passwords. Afterwards, we will setup a state-of-the-art and secure WPA2 Enterprise (IEEE 802.11 RSN) network with EAP-TLS authentication. In the mobile network part, we will setup and analyze a GSM network and discover the vulnerabilities of this network type.

This work is organized as follows: For each of the four parts, WEP, WPA, WPA2 and GSM, the procedure carried out in the lab is described. Afterwards, the results are presented and the questions of the lab assignment are answered.

1 WEP Penetration and Cryptanalysis

1.1 Experimental Procedure

The objective of this task is to register to a WEP network as a legitimate client without knowing the password. The network is provided by a Cisco AP with 104/128-bit WEP encryption and a hidden Service Set ID (SSID). As the SSID is hidden, the first task is to find the SSID. For this, the wireless network detector program `Kismet` is used in conjunction with the NIC in monitor mode. `Kismet` is able to automatically determine the hidden SSID as soon as it captures a probe request/response or an association/reassociation request, as these frames always contain the SSID. As the manufacturer (`Net Manuf: Cisco-Li`) is also shown in Kismet, we could immediately retrieve the SSID, `WirelessLab2019Week2`. If the SSID is not shown directly, because no packet that contains the SSID was captured yet, legitimate clients can be forced to reassociate through sending a deauthentication frame with the `aireplay-ng` tool. All other network details were also retrieved via `Kismet` and are listed in the answer to question 1. Important information is especially channel and BSSID which are required for capturing the traffic with `airodump-ng` in the next step. We now start capturing the traffic with the `airodump-ng` tool and in parallel to crack the password via the `aircrack-ng` tool via the commands:

```
airodump-ng -write wep_traffic -channel 6 -bssid 98:FC:11:B2:D2:67 wlp58s0
aricrack-ng -bssid 98:FC:11:B2:D2:67 wep_traffic.cap
```

The tool `aircrack-ng` automatically tries to crack the password with the so far captured traffic until it succeeds. It uses the PTW-attack and as a fallback in case this does not success the FMS/KoreK attack. In our case, the tool succeeded with its default method, the PTW-Attack after acquiring about 50.000 keys, as can be seen in Figure 1. The key is also listed in the answer to question 1. Having retrieved the key, we are now able to sign in to the network as if we would be a legitimate client. Also, we are now able to decrypt all the acquired traffic. In order to do so, we use wireshark. We set the retrieved encryption key in wireshark and then inspect the traffic.

1.2 Results

We were able to retrieve the key within minutes after about 50.000 initialization vectors were captured. This can be considered quite fast, as the PTW attack needs between 40.000 and 85.000 keys to crack the password. Another reason we could acquire the key that fast was that there was much traffic. With less traffic, ARP packet injections might be necessary to acquire enough data. The retrieved AP and WEP encryption data can be found in the answer to question Q1. The cracked WEP key can additionally be seen in Figure 1 and the associated clients, acquired with `Kismet` and `airodump-ng` can be seen in Figure 2 and Figure 3. We can see that there is a client browsing the webpage `www.aircrack-ng.org`

1.3 Questions

Q1. *Describe the parameters of the AP under analysis, such as the SSID, BSSID, channel number (and optionally the frequency), encryption mechanism, associated clients, the WEP key. How many packets did the PTW attack require? Which web page was the legitimate client browsing?*

- AP BSSID: 98:FC:11:B2:D2:67
- AP SSID: WirelessLab2019Week2
- mon0: channel 6 frequency: 2.437 GHz
- encryption WEP
- WEP key Hex: 45:2D:6A:35:40:33:34:26:45:6E:30:39:3F
- WEP key ASCII: E-j5@34En09?
- associated clients: see Figure 3
- number of packets: 50774
- `www.aircrack-ng.org`

Q2. *Is it possible to run a completely passive PTW attack on 104-bit WEP? Why and under which circumstances?*
In many cases, it is [1]. As we can not inject packets the traffic must be high enough so we can capture enough packets. The packets that are not ARP have to be IPv4 because we know that the first encrypted bits correspond to the LLC/SNAP header. As we know that the LLC header is always the same, and 4 out of the first 7 bytes of the IP header are also fixed. One of the 3 bytes can only have two values and so we just have to find the 2 others bytes that can be found by iterating over all the possibilities. This gives us enough plain text - cipher text pair bytes to carry out the attack.

Q3. *Why is it possible to send an arbitrary amount of ARP-requests to the AP without knowing the WEP key?*
This is possible because there is no replay protection in WEP and it is possible to do it arbitrarily often as there are usually no packet filters or rate limiting rules for link layer packets [1].

Q4. *What can you do to strengthen your attack when there are some clients, but hardly any traffic at all?*
We can send a de-authenticate message to one of the clients to tell it that it lost the connection. It will usually try to reconnect and send a new ARP packet. We can capture this packet and send it arbitrarily to the AP.

Q5. *How can you obtain packets for injection if no clients are associated with the AP?*
If the AP uses Open System authentication we just have to send a request to connect. If the AP

uses Shared Key authentication we can do a fake authentication with aireplay-ng. Afterwards we can do a chop chop or fragmentation attack to obtain a PRGA (pseudo random generation algorithm) file. This file can be used for the generation of new packets. Finally, we can use packetforge-ng to create an ARP packet for injection [2].

Q6. *Which weaknesses of WEP are we taking advantage of in our attack, and why?*
The aircrack-ng tool uses the PTW-Attack as its default method and the older FMS attack as a fallback. The main weakness utilized in the FMS attack is that the RC4 pseudo random number generator algorithm violates the cryptography principle of diffusion for the first bytes of the keystream in case certain keys, so-called weak keys are used. Diffusion means that changing a single bit in the plaintext/key should result in a change of statistically 50% of the bits in the ciphertext and vice versa. The general use of weak keys cannot be avoided, as the changing IV will sooner or later result in a weak key. The first bytes of the plaintext are known (802.11 LLC and SNAP header plus the ARP header in case of ARP packets or fixed bytes of the IP header) and due to this weakness there is a correlation between ciphtertext, plaintext and secret key bytes. Therefore only certain values are possible, for the keybytes which can be narrowed down to the correct key bytes if enough traffic was captured. Another weakness is that in WEP, the IV is prepended to the Key ID (resulting in the RC4 encryption key) and sent in clear. Therefore the first three bytes of the RC4 encryption key (=the IV) are always known, so the attacker can wait for a potentially weak key. The PTW attack is a newer attack that does not rely on weak IVs. It is based on the Klein Attack which uses new correlations between plaintext, ciphertext, keystream and secret key bytes to retrieve the key. The PTW attack extends the Klein attack and also uses enhanced FMS techniques. Therefore the main weakness exploited in this attack is also the RC4 encryption algorithm [1; 3; 4; 5].

Q7. *Under what circumstances would you consider WEP to be sufficiently secure?*
If the network has an additional intrusion detection, which prevents attackers from injecting arbitrarily many ARP packets AND the network traffic is very low, it could be considered sufficient. Still, WEP should not be used at all if in any case avoidable and if not, at least additional security measures on the upper layer protocols should be taken, such as the use of a VPN or tools that enforce the use of encrypted protocols such as HTTPS.

Q8. *Would these attacks be effective against a network deploying WPA with TKIP? If the RC4 cipher used in WEP is considered to be insecure, why is it reused in WPA?*
These attacks would not be effective against a network deploying WPA with TKIP, as TKIP compensates the weaknesses from RC4 that are used in these attacks. Specifically, the use of weak keys is avoided, the IV is not send in clear anymore and there are per-packet keys that are processed through a key mixing scheme. WPA was designed to be compatible with existing WEP network hardware as its intended use was to quickly upgrade potentially unsecure network hardware using WEP. The encryption in network hardware is done utilizing hardware accelerators. Thus, deploying a different encryption mechanism, such as AES, was not possible. Therefore RC4 (with additional measures to counteract its weaknesses) had to be chosen.

2 Password Dictionary Attack

2.1 Experimental Procedure

The objective of this task was to create a network using the WPA-PSK protocol and then hack into the network using a dictionary attack. The first step to do that was to create the access point on one of the computers using hostapd. To do so we configured the parameters of the AP in the file `/etc/hostapd/hostapd.conf` as shown below.

```
interface=wlp58s0
bridge=br0
driver=nl80211
ssid=Group5WPA
hw_mode=g
wpa_key_mgmt=WPA-PSK
wpa=1
wpa_passphrase=Rainbow1
rsn_pairwise=TKIP
```

Once this is done we could start the AP using the command line `hostapd /etc/hostapd/hostapd.conf` (Figure 4 in Appendix B) and connect to the network with our personal laptop using the PSK. The second part of the task was to hack into the network performing a dictionary attack. The first step was to build the dictionary. As suggested we downloaded a dictionary from the internet and then we created an extended list of strings using john. `john -wordlist=words.lst -stdout -rules > extendedwords.lst`. From a list of 233.1 Ko we obtain an extended list of 12.8 Mo. This list contains words with uppercase letters, numbers and special characters (Figure 5). Thanks to this file we can now run this dictionary attack with aircrack-ng.

2.2 Results

As you can see in Figure 6 in Appendix C, our password was in the extended list used to run the attack. It took only 47 sec to the program to find the password.

2.3 Questions

Q9. *Why is TKIP a more secure encryption mechanism than the one-key WEP alternative?* While still relying on RC4 as primary encryption mechanism, TKIP implements several measures to counteract the weaknesses of WEP. Instead of using a single key, it uses a key hierarchy with derived temporal keys that are different between each pairwise communication, i.e. between each STA and AP as well as different sets of keys for encryption and authentication. Additionally, several other measures have been taken to make TKIP more secure:

- Add Message Integrity Protocol (MIC)
- Increase the size of the IV, change the rules of how IV values are selected and avoid use of weak IV values, reuse IV as replay counter
- Change encryption key for every frame and perform key mixing
- Add mechanism to distribute and change the keys

Q10. *Which information is used to compute the PMK and the PTK in the password-based PSK scenario? Why is the offline password brute-force attack possible?*
The password, SSID and SSID length are used to compute the PMK like the following:

$$PMK = PBKDF2(HMAC - SHA1, Password, SSID, 4096, 256) \tag{1}$$

The PMK, MAC addresses of STA and AP and Nonces of STA and AP are used to compute the PTK like the following:

$$PTK = f(PMK, MAC_{AP}, MAC_{STA}, Nonce_{AP}, Nonce_{STA}) \tag{2}$$

The offline attack is possible because of the way the 4-way handshake is carried out: the nonces and MAC addresses are sent in clear in the first two messages, thus we have all the information except the PMK, which we start guessing. The PMK can be checked versus the MIC. As soon as the PMK is found, the dictionary attack against the PBKDF2 (Password Based Key Derivation 2) can be started. As the SSID is known, we also have every necessary information to start the attack. We know that we found the correct password through comparing the output of the PBKDF2 against the already retrieved PMK.

Q11. *Does the compromise of password imply the disclosure of traffic from previous sessions? Justify your answer.*
Yes, it does, as WPA and WPA2 do not provide forward secrecy. The requirement is that the full traffic including the 4-way handshake is captured. We then know all the relevant information (password, nonces, MAC addresses, SSID) to re-perform first the PBKDF2 to acquire the PMK from the password, and then we can re-compute the PTK, that was used for the session and decrypt the traffic.

Q12. *Does the use of AES in WPA2-PSK give an advantage against a password dictionary attack, as compared to RC4 in WPA-PSK?*
The main difference between WPA and WPA2 is the algorithm that we use to encrypt the data. This encryption happens after the 4 way handshake. The dictionary attack uses a weakness in the 4 way handshake so the use of AES gives no advantage against the dictionary attack.

Q13. *What would be advantages and disadvantages of using a precomputed database of PMKs in a PSK password dictionary attack?*
The general advantage of precomputed databases (or rainbow tables) is a speed up in retrieving the password, while the disadvantage is the required memory for storing these databases, which can be large. However, as it is stated in the lab report [6]: "Note that it is not possible to pre-compute a table of PSK/PMKs (rainbow-table) because the SSID is also an argument to PBKDF2". This means, that a database could only be computed for one specific SSID, which would not make much sense.

Q14. *Suppose an attacker employs 30 days of processing time on the new supercomputer at NTNU (find out its processing power at [24] and recall the password enumeration speed in the lab). Which minimum requirements would you put on a WPA password to ensure that the probability of successful attack does not exceed 2e-20?*
During the lab we recorded a speed of 1604.96 keys per seconds. With the same computer in 30 days we could test $4.16 * 10^9$ keys. If we don't restrict the use of the 95 characters available to create the password then we would have 95^n possibilities of passwords with n equal to the number of characters used to create the password. To find the minimum number of characters

to use so the chance of an attack would be under 2e-20, we need to solve the following equation:

$$\begin{aligned} p = \frac{passwordcheck}{passwordpossibilities} = \frac{4.16 \times 10^9}{95^n} &< 2^{-20} \\ \Rightarrow 95^n &> \frac{4.16 \times 10^9}{2^{-10}} \\ \Rightarrow n &> \frac{\ln \frac{4.16 \times 10^9}{2^{-20}}}{\ln 95} = 7.9 \end{aligned} \tag{3}$$

Therefore we would need at least 8 characters.
If the hacker used a supercomputer which processing power is 275 teraflops [7], then we would need even more characters. The processing power is multiplied by 86 so we could test $86 * 4.16 * 10^9 = 3.58 * 10^{11}$ keys.
Solving the equation we find:

$$n > \frac{\ln \frac{3.58 \times 10^{11}}{2^{20}}}{\ln 95} = 8.88 \tag{4}$$

In this case we would need at least 9 characters.

3 Setting up RSN-EAP Wireless Access Point

3.1 Experimental Procedure

The goal of this task was to set up a WPA2-Enterprise network (RSN-EAP configuration of the IEEE802.11 standard). WPA2-Enterprise networks require 3 entities: A supplicant (e.g. a laptop), an authentication server (AS) and an authenticator (usually the AP). We implement the AS using the software `FreeRADIUS` and the AP using `hostapd` on one PC and use the other PC as the supplicant. First we use OpenSSL to create a basic Public Key Infrastructure (PKI) according to the lab description: We create a self-signed certificate for our Certification Authority (CA) which is then used to sign the Certificate Signing Requests (CSRs) of the supplicant and the AS. Then we configure the radius server:

/etc/freeradius/eap.conf

```
default_eap_type=tls
certdir=${confdir}/certs
cadir=${confdir}/certs
private_key_file=${certdir}/private/askey.key
certificate_file=${certdir}/newcerts/ascert.pem
CA_file=${cadir}/cacert.pem
dh_file=${certdir}/dh
random_file=/dev/urandom
```

For the eap-configuration we set the eap-type is set to `TLS`. Additionally, for `TLS` we have to specifiy the certificate and private key files, the dh file we created with OpenSSL and `/dev/urandom` as our random file. FreeRADIUS also needs a client configured in the `/etc/freeradius/client.conf`. We set the IP-address to localhost, as AS and Authenticator are running on the same machine and specify a secret which we will also set in the `hostapd` configuration, as shown as follows.

/etc/freeradius/client.conf

```
client localhost {
 ipaddr = 127.0.0.1
 secret = testing123
 shortname = localhost
 nastype = other
}
```

For the Authenticator, we need to modify the `hostapd.conf` file as specified below:

/etc/hostapd/hostapd.conf

```
interface=wlp58s0
bridge=br0
driver=nl80211
ssid=Group5WPA2
hw_mode=g
eapol_version=2
dh_file=/home/ttm4137/task3/dh
auth_server_addr=127.0.0.1
auth_server_port=1812
auth_server_shared_secret=testing123
radius_server_auth_port=1812
wpa_key_mgmt=WPA-EAP
wpa=2
wpa_passphrase=Rainbow1
rsn_pairwise=CCMP
```

The most significant changes are the switch to WPA2-EAP with AES-CCMP and the configuration of the `RADIUS` AS with localhost and the same secret as set in the `FreeRADIUS` clients configuration. After setting up and starting AS and AP, we connect our second PC through the network with the `NetworkManager` as can be seen in Figure 9 in Appendix F. We also create a configuration file for the `wpa-supplicant` as can be seen in Figure 10 and connect via this program. You can find a scheme detailing the protocol from the first message of the supplicant who asked to connect to the access point to the message from the radius server to the access point authorizing the association. The whole authentication process can be seen in Figure 7 in Appendix D.

3.2 Results

The log of Part 3 and 4 of the 4-way handshake and the successful authentication can be seen in Figure 8 in Appendix E.

3.3 Questions

Q15. *When would it be appropriate to use WPA2-PSK instead of WPA2-EAP (as a key management scheme)?*

In small office and home installations (SoHo), where only a limited number of trusted persons has access to the network. Additionally, a strong PSK must be chosen.

Q16. *EAP-TLS deploys mutual authentication of two communicating parties. What kind of attack is possible against authentication protocols lacking such authentication?*
It facilitates attacks with a malicious AP, as the STA cannot check if the AP is malicious. The AP can e.g. send challenges and receives responses. It can also send a success message during the handshake without knowing the key. This is possible in WEP. (Answers apply to AP authenticating station but not vice versa, which is the usual case). An advanced attack would be a man in the middle attack.

Q17. *Give an overview of the EAP authentication protocols which can be used in WPA2-Enterprise WLANs.*
The most common protocols are [8]:

- **13 – TLS over EAP** (uses RADIUS, more complex to implement, uses SSL certificates for authentication)
- **17 – Cisco Light EAP (LEAP)**
- **18– EAP-SIM** for cell phones to connect to wifi interface via EAP.
- **21– TTLS over EAP** (less overhead, no client side certificates necessary)
- **25– PEAP** (easy to implement, authentication via username and password)
- **43 – EAP-FAST**

Q18. *List the security-related protocols used in your RSN-EAP-TLS setup and explain their purpose.*
The security related protocols according to [9] are:

- **EAP (RFC2284)** EAP is designed to support different types of authentication methods (generic authentication framework), encapsulates the specific authentication protocols RADIUS - used to communicate EAP messages between AP and Authentication Server
- **802.1X EAPOL (Between STA and AP)** EAP Over LAN, include additional message types to support using EAP over a LAN network
- **TLS (RFC2246)** Provides the concrete authentication mechanism, based on certificates
- **TLS over EAP (RFC2716)** Defines how TLS is used encapsulated in EAP
- **802.11 (Between STA and AP)** the WiFi protocol stack
- **RADIUS (RFC28659)** Protocol between AP and Authentication Server for Authentication of clients
- **EAP over RADIUS (RFC2869) (Between AP and Authentication Server)** RADIUS was first designed for PPP (Point-to-Point Protocol) with its authentication methods PAP and CHAP. For RADIUS to support EAP, some minor modifications were necessary (e.g. access-challenge method is not a challenge but used to send EAP requests and responses)

4 Mobile Network Security

4.1 Experimental Procedure

In the last part we set up a GSM one-cell test network with one of our PCs as a software-based GSM access point in conjunction with a software-defined radio device (USRP) as the physical interface. As the software is already installed and ready to use, we can perform checks to see that the USRP is working properly and then switch to the OpenBTS Command Line Interface (CLI). We configure OpenBTS for our use case as shown in Appendix H. Afterwards, it is possible to find the networks on the test smartphones by doing a manual network operator search. However, the authentication fails. We note down the IMSI, TMSI and IMEI of the phone that tried to connect.

Wireshark SIP filter

Open Authentication First we examine Open Registration Mode. Any IMSI that matches a regular expression is allowed to access the network in this mode, therefore we set `config Control.LUR.OpenRegistration 901700000012925` (the IMSI of our first phone). After re-registering it is now possible to connect to the network. The output of the TMSIS command can be seen in Figure 11. The TMSIS Command now shows that we are authenticated with an AUTH value of 2 and that a welcome message was sent. We inspect the captured Wireshark Traffic, filter for `SIP` messages and find a single registration message. We note that while being registered to the networks, services such as calls or SMS are not possible with Open Registration.

Cached Authentication Next we examine Cached Authentication. We disable Open Registration again and run the `nmcli` python script to create subscribers with the retrieved IMSIs and the MSISDNs 1 and 2 for the two phones. The output can be seen in Figure 12. After having trouble with single digit MSISDNs we change them to the values 1112 and 1113. We inspect the wireshark traffic and the output of the TMSIS command in OpenBTS as shown in Figure 13 and cannot see any differences between those two authentication methods. However, it is now possible to make phone calls and send SMS between the two phones.

Full Authentication The final part is Full Authentication. We run the nmcli script again, this time additionally with the `Kis` 8BE76D863C1B9D834C46FD37765E8D35 and 64A5D236684F3C3510E908FF529D1E19 for our phones. We then enable encrpytion by setting `config GSM.Cipher.Encrypt 1`. Calls and SMS are still possible, but we now notice differences in the output of this TMSIS command: The `AUTH` Column now changed to 1 and the `IMEI` is not displayed anymore (see Figure 14).

4.2 Results

Single Registration Messages: weak security (4-way handshake) Cached Authentication allows network functions although no better security.

4.3 Questions

Q19. *Make a technical specification of the equipment you used and the mobile network you set up.*

- Software Defined Radio (USRP): Ettus B200mini (Serial 30B005B, Band 1900 MHz, ARFCN: 554)

- Range Networks OpenBTS Software
- Standard Android Smartphones
- Intel NUC PC with Linux Ubuntu
- Overall setup: One-cell OpenBTS network (software based, lightweight network with direct interconnectivity between USRP and internet telephony protocols via OpenBTS [10]).

Q20. *Note similarities and differences between the OpenBTS implementation you experimented with and the ideal textbook/lecture description with respect to the usage of the subscriber's temporary identity (TMSI).*
The TMSI is a temporary identity that is used to enable a certain degree of user anonymity. The TMSI is assigned to the MS when it authenticates to the network and the VLR keeps the IMSI-TMSI relationships in the standard implementation. If the TMSI is not present in the BSS, the BSS resolves a new TMSI through an L3 MM Identity Request Message [11]. In OpenBTS, the TMSI-IMSI relationships are kept internally in a database. No other entities are invoked in the process. The TMSI behaviour can be controlled in the OpenBTS configuration [11].

Q21. *Explain the technical differences between 'Open Registration' and 'Cached Authentication' based on the data in your access database ('TMSI-table') and the authentication messages that you captured. Exemplify.*
There is no difference between 'Open Registration' and 'Cached Authentication' regarding the TMSI table and the captured authentication messages. However, Open Registration offers only limited functionality, thus phone calls or SMS service is not available compared to Cached Authentication. In fact, Open Registration is a feature that is available in OpenBTS only in order to make assigning to a network easier [10].

Q22. *Compare OpenBTS 'Cached Authentication' to the 'Full Authentication' and the textbook/lecture/standard description of the GSM authentication. What is required to perform the complete GSM authentication protocol and how is this achieved in the lab?*
The standard GSM authentication is as follows: the MS presents IMSI/TMSI to the BSS. The BSS resolves the TMSI to an IMSI through a database or Identity Request message and sends it to the HLR/AuC. Then a challenge-response mechanism based on random numbers, `Ki` and the algorithms A3 and A8 is carried out. This exchange, if successful, creates also `Kc` for data encryption. The standard only defines the characteristics of the A3/A8 algorithm, the specific algorithms are secretly exchanged between SIM manufacturers and carriers [11]. In `OpenBTS` Full Authentication, the BSS is replaced by `OpenBTS` itself and the HLR/AuC is replaced by the subscriber registry. COMP128v1 is used as A3. `OpenBTS` cached authentication invokes the same protocol steps with a different behaviour on the registry and without invoking the A3 algorithm. This is a weaker authentication that can be used if `Ki` is not known [11]. Despite the basic hardware setup, `Ki` must be known to perform the complete authentication. This key is never transmitted over any interface, i.e. is only present in the SIM and the AuC. In the lab, this key is given and handed over to `OpenBTS` via the nmcli script.

Q23. *Compare the OpenBTS encryption behaviour to the GSM encryption standard. Describe the technical differences that you can find when OpenBTS encryption is disabled and when it is enabled, based on data in your 'TMSI-table', the key K i , and the protocol messages that you captured. Refer to both scenarios: using your own SIM card, respectively using the programmable SIM cards. Exemplify.*
The GSM standard only defines the characteristics of the algorithms, the specific algorithms are not disclosed. OpenBTS uses the A5/1 library that is available from `www.scard.org/gsm` as A5 encryption algorithm [11]. In the TMSIs table, AUTH changes to 1 with encryption enabled (2

without encryption) and the IMEI cannot be seen anymore with encryption enabled. We did not use our own SIM cards as it was not possible to detect the 1900 MHz band network with our phones.

Q24. *Which security modes of operation are signalled to the user by the phone's display? In particular, consider various types of registration, authentication, and encryption modes.*
The user is not aware of the security modes of operation. The user is only notified whether the registration is successful or not. Additionally, even a downgrade from a network with better security such as 4G to the insecure 2G is done automatically without any prominent notification (can only be seen by the small connection symbol on Android/iOS phones).

Q25. *How can this lab project be improved? (Answering is optional and does not influence your grade.)*
One possible improvement might be splitting the lab on several weeks doing only one task a week. In this way it would be more interesting because it can get challenging to focus when only theory is discussed during the course. It would also be easier to follow because the student would try to understand one chapter a week rather than trying to understand everything in one time for the final Lab.

5 Conclusion

The first task allowed us to gain an insight into WEP and how security should not be implemented. WEP is a protocol with many security flaws and should not be used at all anymore. With the knowledge of this course and publicly available tools it was possible to crack the WEP key within a very short amount of time. Even without the theoretical background knowledge, people should be able to break into WEP just by using a few tools and commands.

In the second and third task we got a deeper understanding about the security of more recent WPA/WPA2 networks. It became clear that the network administrator has to be aware of a lot of configuration parameters in order to create a secure network and that it is absolutely mandatory to choose strong passwords, because the design of the 4-way handshake makes offline brutefore/dictionary attacks possible.

The last task gave an insight into the old and insecure GSM mobile network. Just as with WEP, it was a demonstration why it is a bad idea to develop protocols and encryption mechanism in secrecy (violating Kerkhoff's principle) instead of making them public to be analysed by security experts. Important to see was also, that the user is not aware about the current security status of his phone. Therefore it might be a good idea to forbid the use of 2G networks in the phone configuration in general if possible.

Appendix A Results for Challenge 1

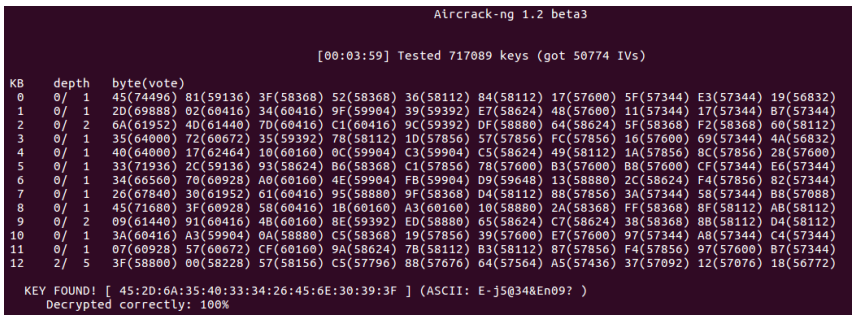

Figure 1: Cracked WEP key

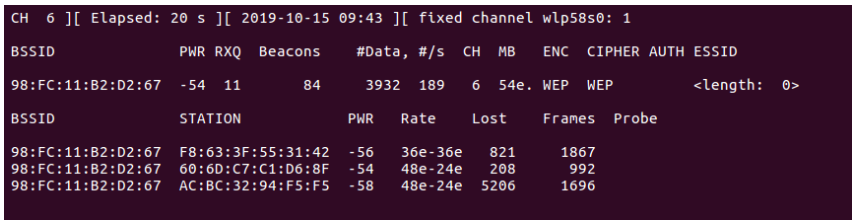

Figure 2: Traffic of associated clients captured with `airodump-ng`

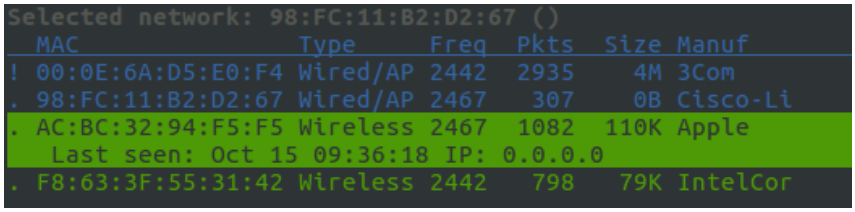

Figure 3: Clients associated to the access point

Appendix B Experimental procedure in Challenge 2

```
ttm4137@ttm4137-desktop:~/group5/task2$ sudo hostapd /etc/hostapd/hostapd.conf
[sudo] password for ttm4137:
Configuration file: /etc/hostapd/hostapd.conf
Using interface wlp58s0 with hwaddr f8:63:3f:27:86:11 and ssid "Group5WPA"
wlp58s0: interface state UNINITIALIZED->ENABLED
wlp58s0: AP-ENABLED
wlp58s0: STA 18:1d:ea:c4:40:dc IEEE 802.11: authenticated
wlp58s0: STA 18:1d:ea:c4:40:dc IEEE 802.11: associated (aid 1)
wlp58s0: AP-STA-CONNECTED 18:1d:ea:c4:40:dc
wlp58s0: STA 18:1d:ea:c4:40:dc RADIUS: starting accounting session 5DA5AA5A-00000000
wlp58s0: STA 18:1d:ea:c4:40:dc WPA: pairwise key handshake completed (WPA)
wlp58s0: STA 18:1d:ea:c4:40:dc WPA: group key handshake completed (WPA)
wlp58s0: STA d0:65:ca:5e:d5:a2 IEEE 802.11: authenticated
wlp58s0: STA d0:65:ca:5e:d5:a2 IEEE 802.11: associated (aid 2)
wlp58s0: AP-STA-CONNECTED d0:65:ca:5e:d5:a2
wlp58s0: STA d0:65:ca:5e:d5:a2 RADIUS: starting accounting session 5DA5AA5A-00000001
wlp58s0: STA d0:65:ca:5e:d5:a2 WPA: pairwise key handshake completed (WPA)
wlp58s0: STA d0:65:ca:5e:d5:a2 WPA: group key handshake completed (WPA)
```

Figure 4: Working Access Point

```
Ragweed1
Raid1
Rail1
Railbird1
Railhead1
Railing1
Raillery1
Railroad1
Railway1
Raiment1
Rain1
Rainbow1
```

Figure 5: Extract of the extended list

Appendix C Results in Challenge 2

```
                        Aircrack-ng 1.2 beta3

                [00:00:47] 74784 keys tested (1604.96 k/s)

                        KEY FOUND! [ Rainbow1 ]


    Master Key     : D2 FE E9 1E E2 4B E0 94 EE 59 82 C9 33 2E A0 E4
                     0A 7E 38 D9 0E 9D 63 B4 C5 55 4F 7D CD AE 44 97

    Transient Key  : 7A E1 10 7C 3D 01 3C CF AE 15 F4 85 01 A7 AD DC
                     30 AA 3C 5F AE 27 8F B0 BD 51 51 C5 6F 12 77 BD
                     FC E8 64 63 C7 19 CB 7E D2 BD 14 C9 98 07 77 24
                     C2 D5 9C 29 A3 47 50 E4 4F C9 9E 1F 0E 3B 51 20

    EAPOL HMAC     : E1 14 DB BC 5D F3 65 2C AB DF 27 60 1F E3 DE F8
```

Figure 6: Result of the dictionary attack

Appendix D Supplicant, authenticator and authentication server in Challenge 3

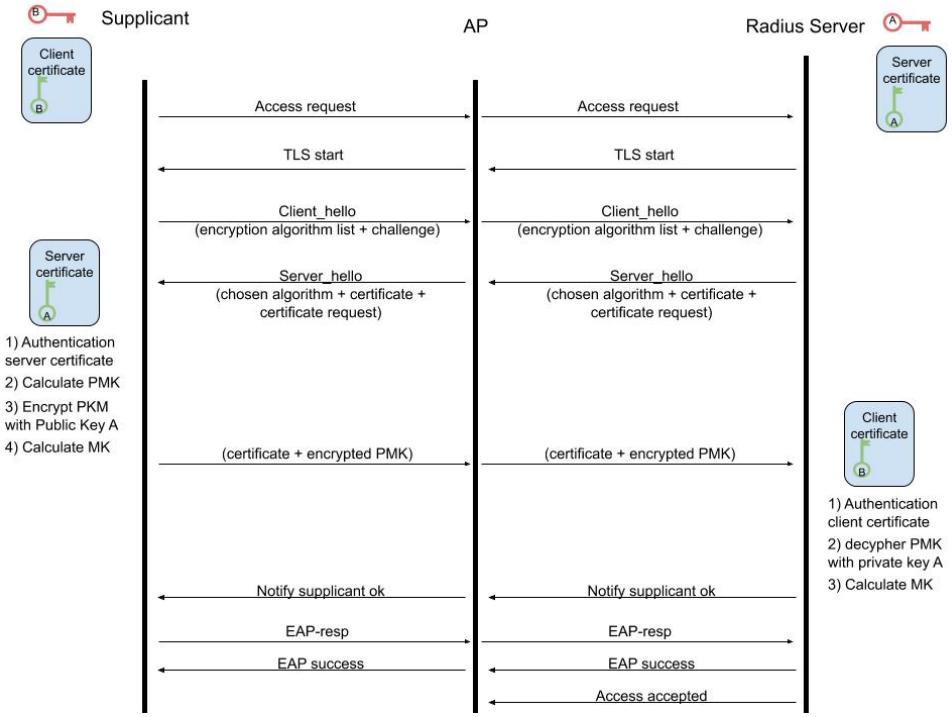

Figure 7: Supplicant, authenticator and authentication server

Appendix E Results for Challenge 3

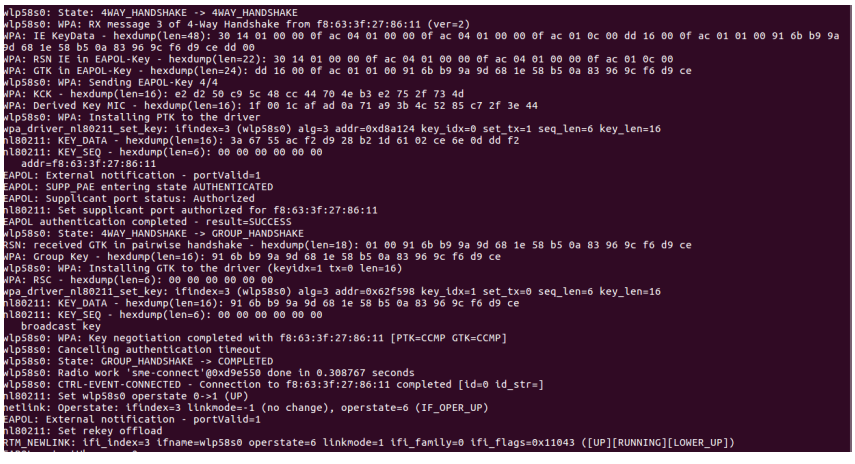

Figure 8: WPA-Supplicant Log

Appendix F Experimental procedure in Challenge 3

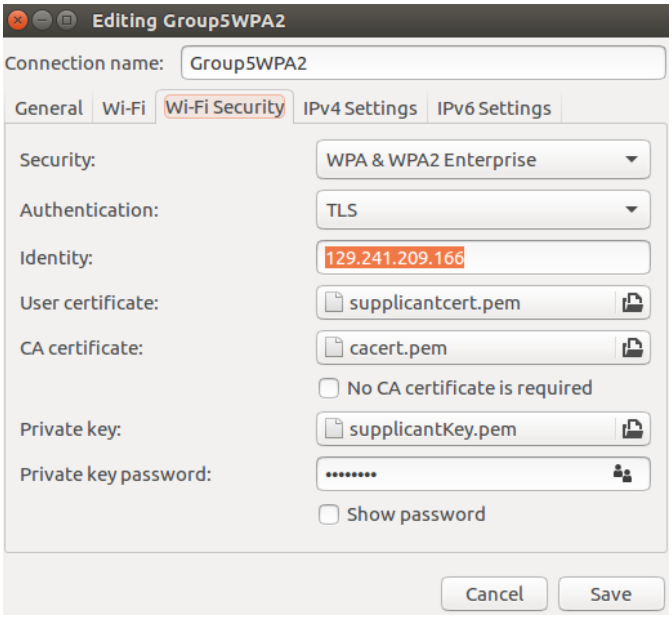

Figure 9: Network Manager Configuration

```
#########################
# WPA Supplicant config file suggestion
# WPA2-EAP/CCMP using EAP-TLS
#########################
ctrl_interface=/var/run/wpa_supplicant
network={
ssid="Group5WPA2"
key_mgmt=WPA-EAP
proto=WPA2
pairwise=CCMP
group=CCMP
eap=TLS
ca_cert="/home/ttm4137/group5/task3/cacert.pem"
client_cert="/home/ttm4137/group5/task3/supplicantcert.pem"
private_key="/home/ttm4137/group5/task3/supplicantKey.pem"
private_key_passwd="Rainbow2"
identity="129.241.209.166"
}
```

Figure 10: WPA-Supplicant Configuration

Appendix G Open Registration Mode in Challenge 4

```
OpenBTS> tmsis -l
IMSI            TMSI     IMEI            AUTH CREATED ACCESSED TMSI_ASSIGNED PTMSI_ASSIGNED AUTH_EXPIRY REJECT_CODE ASSOCIATED_URI ASSERTED_IDENTITY WELCOME_SENT
901700000012925 0x5922d 356489058004670 2    5m      11s      1             0              -           0                                                1
901700000011205 0xf881f 866252040895550 2    24m     24m      1             0              -           0                                                1
```

Figure 11: Open Registration

Appendix H OpenBTS configuration in Challenge 4

OpenBTS Configuration

```
config GSM.Radio.Band 1900
config GSM.Radio.C0 554
config GSM.Identity.MNC 05
devconfig GSM.Radio.RxGain 0
texttttControl.GSMTAP.TargetIP 127.0.0.1
Control.GSMTAP.GSM 1
Control.LUR.OpenRegistration enabled
```

Appendix I Cached Authentication in Challenge 4

```
ttm4137@ttm4137-desktop:~/dev/NodeManager$ ./nmcli.py sipauthserve subscribers read
raw request: {"command":"subscribers","action":"read","key":"","value":""}
raw response: {
        "code" : 200,
        "data" : [
                {
                        "imsi" : "IMSI901700000012925",
                        "msisdn" : "1",
                        "name" : "name"
                },
                {
                        "imsi" : "IMSI901700000011205",
                        "msisdn" : "2",
                        "name" : "name"
                }
        ]
}
```

Figure 12: Registered subscribers

```
OpenBTS> tmsis -l
IMSI           TMSI     IMEI           AUTH CREATED ACCESSED TMSI_ASSIGNED PTMSI_ASSIGNED AUTH_EXPIRY REJECT_CODE ASSOCIATED_URI ASSERTED_IDENTITY WELCOME_SENT
901700000012925 0x5922d 356489058004670 2    20m     33s      1            0             -           0                                                1
901700000011205 0xf881f 866252040895550 2    39m     38s      1            0             -           0                                                1
```

Figure 13: Cached Authentication

Appendix J Full Authentication in Challenge 4

```
OpenBTS> tmsis -l
IMSI           TMSI     IMEI AUTH CREATED ACCESSED TMSI_ASSIGNED PTMSI_ASSIGNED AUTH_EXPIRY REJECT_CODE ASSOCIATED_URI ASSERTED_IDENTITY WELCOME_SENT
901700000012925 0x1fabe      1    165s    11s      1            0             -           0                                                0
901700000011205 0xe0ba7      1    119s    33s      1            0             -           0                                                0
```

Figure 14: Encrypted Authentication

References

[1] E. Tews, R.-P. Weinmann, and A. Pyshkin, "Breaking 104 bit wep in less than 60 seconds," in *International Workshop on Information Security Applications*. Springer, 2007, pp. 188–202.

[2] darkAudax, "Tutorial: How to crack wep with no wireless clients." [Online]. Available: https://www.aircrack-ng.org/doku.php?id=how_to_crack_wep_with_no_clients

[3] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *International Workshop on Selected Areas in Cryptography*. Springer, 2001, pp. 1–24.

[4] A. Klein, "Attacks on the rc4 stream cipher," *Designs, codes and cryptography*, vol. 48, no. 3, pp. 269–286, 2008.

[5] "Tutorial: How to crack wep with no wireless clients." [Online]. Available: https://www.aircrack-ng.org/doku.php?id=aircrack-ng

[6] D. o. I. S. NTNU and C. Technology, "Ttm4137 wireless security lab assignment 2019," 2019.

[7] NTNU, "The norwegian university of science and technology, met.no buy new supercomputer." [Online]. Available: http://www.ntnu.edu/news/new-supercomputer-to-be-installed

[8] IANA, "Extensible authentication protocol (eap) registry," 10 2019. [Online]. Available: https://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml

[9] J. Edney and W. A. Arbaugh, *Real 802.11 security: Wi-Fi protected access and 802.11 i*. Addison-Wesley Professional, 2004.

[10] M. Iedema, *Getting Started with OpenBTS: Build Open Source Mobile Networks*. " O'Reilly Media, Inc.", 2014.

[11] I. Range Networks, "Openbts application suite user manual," 2014.