FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Analysis and detection of KRACK attack against WiFi infrastructure
Student:	Bc. Jana Ernekerová
Supervisor:	Ing. Tomáš Čejka
Study Programme:	Informatics
Study Branch:	Computer Security
Department:	Department of Computer Systems
Validity:	Until the end of winter semester 2019/20

Instructions

Study and describe the principle of the KRACK (Key Reinstallation Attack) vulnerability.
Propose a HW or SW testing tool that can check whether a WiFi device under test is resistant against this type of attack.
Study the traffic generated during the KRACK attack and find some characteristics of such traffic that can make the attack detectable.
Design a system for detection of KRACK attacks in real-time.
Test the developed detection system using the testing tool.

References

Will be provided by the supervisor.

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague August 31, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Analysis and Detection of KRACK Attack Against WiFi Infrastracture

Bc. Jana Ernekerová

Department of Computer Systems
Supervisor: Ing. Tomáš Čejka, Ph.D.

January 10, 2019

Acknowledgements

I want to thank my supervisor Ing. Tomáš Čejka, Ph.D., for all his valuable advice and for the time he spent helping me with this thesis. Also, I am grateful to my colleagues and friends for their comments and words of support.

I am very grateful to my parents and family for supporting me throughout my studies, especially during the studies abroad. Finally, I must express my very profound gratitude to my partner for all his love, patience, and support he provides me every day. Thank you.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on January 10, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Ernekerová, Jana. *Analysis and Detection of KRACK Attack Against WiFi Infrastracture*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce analyzuje princip útoku KRACK a navrhuje metody jeho detekce. Kromě toho se práce zabývá návrhem, implementací a testováním systému pro detekci útoku KRACK proti čtyřcestnému handshaku v reálném čase. V analytické části práce jsou nejprve představeny relevantní části standardu 802.11, na které je útok zaměřen. Poté je popsán princip útoku, jeho praktické dopady a protiopatření. Jsou zmapovány dostupné nástroje pro detekci zranitelnosti zařízení k tomuto útoku. Práce se zvlášť zaměřuje na útok proti čtyřcestnému handshaku a analyzuje provoz generovaný během tohoto útoku. Ten je poté srovnán se standardním provozem na síti během čtyřcestného handshaku. Na základě monitorovaného provozu a také analytické části práce jsou pak navrženy charakteristiky k detekci útoků KRACK. Systém pro detekci útoku proti čtyřcestnému handshaku je navržen, implementován a úspěšně otestován.

Klíčová slova KRACK, WPA2, 802.11, bezpečnostní protokoly, CCMP, GCMP, TKIP, čtyřcestný handshake, detekce, útok, Wi-Fi, bezpečnost

Abstract

This thesis analyzes the KRACK attack principle and proposes methods of its detection. Also, it deals with the design, implementation, and testing of a system for detection of the KRACK attack against the 4-way handshake in real-time. In the analytical part of the thesis, first, there are introduced relevant parts of the 802.11 standard which are the target of the attack. Then, the principle of the attack is described, its practical impact and countermeasures. Besides, we map available tools for the detection of device vulnerability to this attack. The thesis is mainly focused on the attack on the 4-way handshake and analyzes the traffic generated during this attack. This malicious traffic is then compared to the standard traffic generated during the 4-way handshake. Based on the monitored traffic and analysis part of the thesis, characteristics for detection of the KRACK attacks are proposed. A system for detection of the 4-way handshake is designed, implemented and successfully tested.

Keywords KRACK, WPA2, 802.11, security protocols, CCMP, GCMP, TKIP, 4-way handshake, attack detection, Wi-Fi, security

Contents

List of Figures

Introduction

Many people these days cannot imagine their lives without the Internet. Not only that the communication and sharing information with people around the world is the simplest it has ever been, but it also became our helper in everyday situations. One of the reasons why this is possible is a speedy development of the Wi-Fi technology. In nearly twenty years of its development, it has become a standard that almost every mobile device can connect to the Wi-Fi. Besides, its installation is cheaper and easier than the installation of the wired networks and provides its users higher comfort when using it. Also, it is cheaper than cellular data which can be a good alternative. There are not only mobile devices that can connect to these wireless networks today, but also televisions, security cameras, and other home appliances. We got used to having free Wi-Fi available everywhere and to everyone. Unfortunately, this technology also brings security risks, because anyone in the signal range can potentially monitor our communication.

To keep data transmitted through Wi-Fi private, we use data encryption. In the original standard, there was an optional security algorithm for data encryption defined; it was called Wired Equivalent Privacy (WEP). Unfortunately, it was broken quite shortly after releasing the initial document, already in 2001 [1, 2, 3]. The Wi-Fi Protected Access (WPA) protocol temporarily replaced it. Although it is still used by some devices mostly because of the backward hardware compatibility, it is not considered to be secure [2]. In 2004 [4], it was finally replaced by the Wi-Fi Protected Access 2 (WPA2) protocol. For more than a decade, this protocol was considered secure, assuming usage of a strong password.

The Wi-Fi security is becoming a more and more important topic. Exposure of the personal data could make our life deeply uncomfortable. But for companies, the breach of their network security can have an even more profound impact. It could lead to an exposure of their know-how, their corporate strategy, or the destruction of the company reputation.

The KRACK attacks is an abbreviation standing for the Key Reinstalla-

tion Attacks. This family of attacks exploits the vulnerability found in the 802.11i amendment which defines the protocols whose implementation is certified under the trademark WPA and WPA2. Specifically, it targets mainly the initial 4-way handshake. This handshake is used for mutual authentication and negotiation of the session key between a client and an access point. Both, the WPA2 and WPA protocols use it. All of the modern protected Wi-Fi devices use some version of these protocols for data encryption. Therefore, most of these devices are affected by this vulnerability. These attacks trick the victim to reinstall an already-in-use key and so, let the adversary replay, decrypt and possibly forge frames. After the publication of the attacks, in October 2017 [5], people were assured that all of the vendors were notified about the vulnerability in advance and the patches they are about to release are going to protect the devices. Unfortunately, there are many devices in the market that are not supported or cannot be patched.

The detection of the KRACK attacks in real-time can help to identify affected devices in a Wi-Fi network. Then, these devices could be updated (if possible), replaced, or in case of a network with a lot of passing clients, the suspected ones could be possibly de-authenticated from the network to avoid the attacks that target the access point. The detection can also help to re-examine the question that has not yet been answered, whether these attacks are being carried out in the real world or not. And if they are, then, on what scale. It could be an excellent future study or experiment which would lead to the creation of a statistic how often are these attacks exploited. Results could be an illustrative instrument to convince common users to be more aware of network security.

This thesis analyzes the principle of the KRACK vulnerability, the traffic generated during the attack against the 4-way handshake and necessary countermeasures for avoiding the attack in general. In this manner, it follows up research about the KRACK attacks by Mathy Vanhoef and Frank Piessens from the Catholic University of Leuven [6, 7, 5, 8]. Besides, the thesis maps currently available tools for detection of a device vulnerability to this attack. We tested a couple of devices by the publicly available device vulnerability detection tool and notified their users about the results. The primary purpose of the thesis is to analyze the principle and the traffic generated during the attack and propose a way how to detect it. For this purpose, the standard Wi-Fi traffic was studied first along with related parts of the Wi-Fi standard. It was necessary for understanding the attack because the vulnerability is in the standard itself. We briefly describe all the vulnerable handshakes, but we chose to focus on the attack on the 4-way handshake. We decided so because this attack has the most significant impact and also because the implementation of the attack was necessary for the detection. Based on the analytical part of the work, a system for detection of the attack to the 4-way handshake in real-time is designed and implemented. The implemented system runs on an independent device physically located in a Wi-Fi network. The system de-

tects triggered retransmissions of the specific frames on the network running in 2.4 GHz band and then logs these security incidents. The system is then verified by performing the attack on the tested network, and the results are expressed as a ratio of false positives and correctly detected events.

The structure of this master's thesis is as follows: In Chapter 1, we describe relevant parts of the 802.11 standard — structure of work-related frames and the handshakes that are vulnerable to the attacks. In Chapter 2, we describe the principle of the KRACK attack vulnerability. Also, we describe its practical impact and countermeasures that should be taken to avoid its exploit. In Chapter 3, we describe the traffic generated during the standard 4-way handshake and during the KRACK attack against it. Based on the traffic and analysis from 2 and 3, we create a list of characteristics we can use for detection of the KRACK attack. In Chapter 4, we first propose a software testing tool that can check whether a Wi-Fi device under test is resistant against this type of attack, second we design a system for detection of KRACK attacks in real-time. Because the topic is extensive, we focused on the attack against the 4-way handshake. In Chapter 5, we describe our implementation of such a system and in Chapter 6, we test the system and evaluated results. Besides, in 6, we test several devices on their vulnerability.

Chapter 1

Wi-Fi Technology

This chapter provides background information to understand the essentials about the traffic sent on a Wi-Fi network, architecture of the designed system and background behind the KRACK attacks principle. First, we describe general technical information about Wi-Fi network and standards which define it. Then, we talk about the structure of the frames sent on such a system. This part is necessary to understand both regular traffic and the one generated during the attack. The association and authentication on a Wi-Fi network are after that explained along with the handshakes vulnerable to the KRACK attacks.

Wi-Fi is a trademark for a wireless network technology based on the IEEE 802.11 family of standards. As the name implies, these standards were made by the Institute of Electrical and Electronics Engineers (IEEE). IEEE 802 refers to a family of IEEE standards dealing with a Local Area Network (LAN) and a Metropolitan Area Network (MAN). This family is maintained by the LAN/MAN Standards Committee (LMSC). An individual Working Group (WG) provides the focus for each area (for example, 802.3 Ethernet, 802.15.1 Bluetooth, and 802.16 WiMAX, etc.). IEEE 802.11 family of standards deals with the implementation of the Wireless Local Area Network (WLAN).

The services and protocols specified in IEEE 802 target data link layer (DLL) and physical layer (PHY), the lowest two layers of the seven-layer OSI networking reference model. IEEE 802 splits the OSI Data Link Layer into two sublayers, logical link control (LLC) and Media access control (MAC). The layers structure is shown in Figure 1.1.

Wi-Fi networks, as other wireless technologies, use radio waves to send data over the air. A device that is used to send out a wireless signal is called transmitter. A device which can pick up that wireless signal and understands the information is called a receiver. A transceiver is a device that has both, a transmitter and a receiver. Different wireless technologies vary in used frequency and a type of modulation. Therefore, a device can only understand

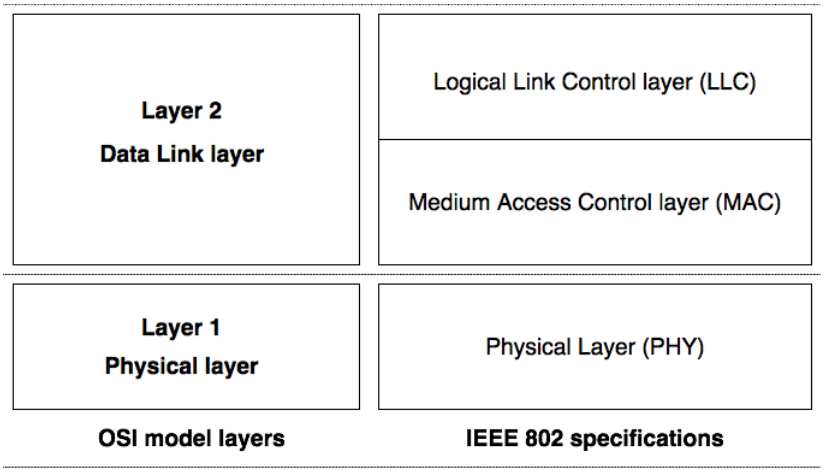

Figure 1.1: OSI model layers vs IEEE 802 specification layers.

a specific wireless signal. The 802.11 standard provides several distinct radio frequencies ranges for use in Wi-Fi communication. In most countries, including European, it commonly operates in two frequency ranges: 2.4 GHz and 5 GHz bands. These bands are part of the Industrial, Scientific and Medical (ISM) radio band, which is reserved internationally. Usage of these frequencies unlike others does not require a government license.

The 2.4 GHz band is the most widely used of the bands available for Wi-Fi (used by 802.11b, g, n amendments). On one hand, devices operating on this band are cheaper to be manufactured than the ones used on the higher frequency. Besides, it is determined by physical laws that its signal, at the lower frequency, more easily penetrates through physical obstacles. On the other hand, there are significantly more devices using it. Thus, it is more prone to interference. The 5 GHz band (used in standards 802.11a and ac) is considerably faster at short distances without obstacles. But the devices using this frequency are more expensive, and many do not support this option at all.

Each range is split into plenty of channels. Devices must be set to the the same channel to be able to communicate with each other. Countries apply different regulations to the legal channels, number of users and maximum power levels within these frequency ranges. Band 2.4 GHz is given between 2,412 and 2,484 GHz, which is divided into 14 independent channels after 5 MHz steps. The fourteenth channel is an exception with a gap of 12 MHz from the previous thirteenth one.

Wi-Fi communication requires a channel width of 20 MHz to operate. Therefore, to avoid the interference between them, there can be only three running networks next to each other. The convention is to use the channels 1, 6 and 11 to avoid interference at the point of transmission. In Europe, the

usage of the fourteenth channel is forbidden. The overlapping channels of the 2.4 GHz band are illustrated in Figure 1.2. The range of the 5 GHz band is significantly higher. There are 19 available channels at this band in Europe. The first 8 for indoor usage and another 11 for outdoors. Channels in 5 GHz band, are further away from each other (20 MHz gaps). Hence, every device has its channel, and the signal does not interfere with others. The Access Points use the channels for regulation of the traffic.

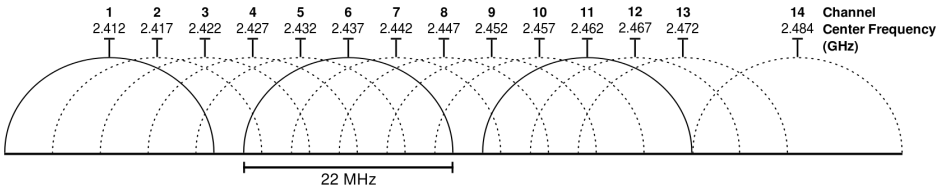

Figure 1.2: Graphical representation of 2.4 GHz band channels overlapping, figure taken from [9].

The area where the wireless network is accessible to users is called a hotspot. When a group of wireless devices is operating with the same network parameters, it is called a service set. Basic service set (BSS) is a group of devices running with the same MAC characteristics (radio frequency, modulation scheme, etc.). Extended service set (ESS) is a set of logical units of one or more basic service sets on the same logical network segment (IP subnet, VLAN, etc.). A system interconnecting a set of BSSs and LANs to ESS is called distribution system (DS). Basic service sets are identified by Basic service set identifiers (BSSIDs) (48-bit label that conforms to MAC address convention) and Logical networks are identified by Service set identifiers (SSIDs). The SSID is commonly called a network name.

A Wi-Fi station (STA) is any addressable unit in a wireless network. The station can have one of three roles in a network. The networks are then built out of combinations of devices operating in these different modes. The first is an Access Point (AP). It is a networking hardware device that connects the Wi-Fi network to a wired network. This device creates a wireless local area network and devices on the network than communicate with this device to access the other networks or to communicate with each other. These connected devices are called clients or non-AP stations. Any laptop, mobile device or IoT device able to connect to a Wi-Fi network can be a client. The third possible device role is the ad-hoc node. This node is a device at the ad-hoc wireless network. The ad-hoc networks are the opposite of the managed (infrastructure) wireless networks, which are based on the AP and clients connected to it. It is decentralized, and every node participates in routing by forwarding data for other nodes.

The presence of a wireless network interface controller (WNIC) gives a

device the ability to connect to a Wi-Fi network. It is a hardware component that connects to a wireless radio-based computer network and uses an antenna to communicate via microwave radiation. A WNIC in a desktop computer is traditionally connected using the PCI bus. Most of the laptops have a wireless network interface built into the motherboard. Another connectivity option is by external WNIC usually connected by USB. The WNIC can operate in two modes: infrastructure, which needs an access point, or ad-hoc mode, formed by ad-hoc nodes (without the need of AP). The WNIC specification usually consists of three crucial information: wireless data transfer rates (speed measured in Mbit/s), wireless transmitted power (measured in milliwatts or dBm) and supported wireless network standards.

The WNIC, as others embedded systems, is on its lowest level controlled by firmware. This small piece of software stored in the Read Only Memory decides how the packets are handled and send. To be able to use WNIC in any operating system a user needs a driver. The driver is OS dependent since it is the software communicating between the OS and the WNIC. The chipset of the WNIC, its firmware and the driver used decides how the device handles packets received in the network.

The other important aspect of the wireless network is its range. It is given by the standard used, the power of the transmitter (and a type of an antenna), the nature of physical obstructions and radio interference in the surrounding area. The typical range of Wi-Fi is in several up to several tens of meters. It is typically higher outdoors because there are less physical obstacles.

1.1 Development and Certification

The initial standard of this family, the IEEE 802.11-1997, was released in 1997 and is obsolete [10]. After publication of the initial document, customers were not satisfied with the degree of compatibility between devices of different vendors. This feedback from customers gave impetus to establish a certification program. For this purpose, the Wireless Ethernet Compatibility Alliance (WECA) was founded in 1999 [11]. This alliance started the certification program and founded the trademark Wi-Fi. The certification program had a significant market impact. In 2003, the WECA was renamed to Wi-Fi Alliance (WFA) [11], as we know it today.

Because of the tremendous success in the market and the perceived shortcomings, the base 802.11 standard provided has been improved and expanded over the years. Additional amendments are distinguished by different letter suffixes (for example, 802.11a, 802.11i, 802.11w, etc.). Nowadays, there are many maintenance changes and amendments to the original standard. Some of the documents contain improvements to the previous ones regarding faster theoretical speed, extended range, security improvements, and lower power consumption. Others are offshoots of the base standard and serve a particular

purpose (for example, 802.11i authentication and encryption, 802.11e Quality of Service (QoS), 802.11w protected management frames, etc.). The goal is for all the 802.11 series of standards to be compatible at the Medium Access Control or Data Link layer. The document and project title, the year of approval and brief content of the significant documents are listed in Appendix B. The amendments defining protocols and handshakes vulnerable to the KRACK attack are listed below:

802.11i (2004) Document deals with authentication and encryption. Its partial implementation is called WPA, and full is called WPA2. It defines the 4-way handshake, the PeerKey handshake, and the group Key handshake [4], all vulnerable to the KRACK attacks [5]. These handshakes are further discussed in Section 1.3.

802.11r (2008) It is also called Fast BSS Transition (or fast roaming). It describes technology to permit continuous connectivity aboard wireless devices in motion. The Fast BSS Transition handshake is also vulnerable to the KRACK attacks [5]. Thus, it will also be explained further in section 1.3. This standard also slightly extends the 4-way handshake and provides a particular state machine of the supplicant [12].

802.11v (2011) Deals with Wireless Network Management (WNM) and device configuration. This amendment defines a WNM-Sleep mode. In [7], the author of the KRACK attack describes a way how to abuse WNM-Sleep response frames to trigger key reinstallation.

802.11z (2012) Defines mechanism called Tunneled Direct Link Setup (TDLS). This mechanism enables a user to directly transfer data between two Wi-Fi clients that are part of the same Wi-Fi network. The TDLS PeerKey handshake is defined for this purpose and is also vulnerable to the KRACK attacks [7] and will be further discussed in section 1.3.

802.11ai (2016) The document provides Fast Initial Link Set-up (FILS) methods to enhance end-user experience in dense environments. This function enables a wireless LAN client to achieve a secure link setup within 100 ms. The FILS handshake is also vulnerable to the KRACK attack [7]. Thus, it will be further discussed in the section 1.3.

As a basis for defining the basic concepts, available services and formats of frames, the IEEE 802.11-2016 [13] standard was used. According to the official IEEE 802.11 working group project timeline [14], it is the latest published standard, including revisions defined in prior published amendments.

The IEEE does not test devices for compliance with their standards. The Wi-Fi Alliance does the only internationally-recognized certification program. This certification program ensures that products have met industry-agreed

standards for interoperability, security, and a range of application-specific protocols. The program also provides a seal of approval that the certified products are backward-compatible with older published standards of the family.

1.2 Establishing Communication

There are two ways how the client can find out about the existing WiFi networks in a range:

- **Actively** The client sends a Probe Request on some channel and waits for a Probe Response from any access point. Inside response frames are network parameters (such as the SSID name networks or supported transfer rates and standards). Based on the obtained information, the client can initiate a network connection. The client sends requests periodically on all channels, to detect networks across the band.

- **Passively** Access points send Beacon frame periodically with information about the network on the channel they are set to. The content of such a frame is similar to Probe Response. The client passively waits for such frames to capture network information across the band.

1.2.1 Authentication and Association

Before a client is allowed to communicate via an AP, it has to become authenticated and associated with the AP. In Wi-Fi networks, the process of authentication is used instead of the wired media physical connection. After authentication, the node has to be associated with an AP. Every node can be associated with no more than one AP, while an AP might be associated with many nodes at the same time. This way it can be unambiguously determined where to send the data specified for a specific node. When using WPA or WPA2 protocols, the system at first uses Open System Authentication, which allows any device to authenticate. The real authentication is performed later during the 4-way handshake. The process of establishing the IEEE 802.11 association is shown in Figure 1.3.

After this process is completed, there follows an optional 802.1x authentication process. This process is habitually used in enterprise and campus networks. After that, the 4-way handshake is performed to derive fresh session keys. The one for the unicast traffic is called Pairwise Transient Key (PTK), while the one for multicast or broadcast traffic is called Group Temporal Key (GTK). These keys are then used by a data-confidentiality and integrity protocol to encrypt data frames.

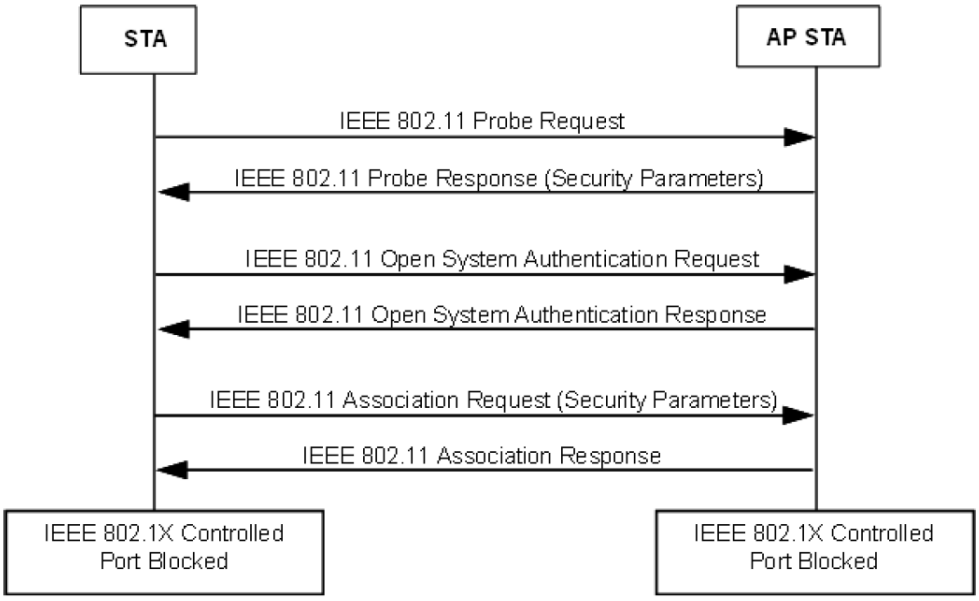

Figure 1.3: Process of establishing the IEEE 802.11 association, figure taken from [4].

1.2.2 The 4-way Handshake

The 4-way handshake is defined in the IEEE 802.11i amendment [4]. According to the standard, during this handshake, a client is called a supplicant, and an AP is called an authenticator.

This handshake provides authentication based on the Pairwise Master Key (PMK). In personal networks, the PMK is based on the pre-shared key, which is the password to the network. While in enterprise networks it is negotiated using 802.1x authentication stage. After authentication, the pairwise transient key (PTK) is derived. Concretely from the PMK, the authenticator nonce (ANonce), the supplicant nonce (SNonce) and the MAC addresses of both the AP and the client. The handshake is also used to transport the group temporal key (GTK) from the authenticator (the AP) to the supplicant (the client). When the GTK is transported, it is saved in the key data field, thus, encrypted using the PTK. The handshake uses EAPOL-Key frames. Their form will be shown in more detail in Subsection 1.5.2. The authenticity of each message except the first one is protected using Message Integrity Check (MIC). The MIC is calculated using the subkey of the PTK called KCK. Every message also contains a replay counter. When the authenticator is sending a message, it always increments the replay counter, and the supplicant is replying to this message with the same replay counter. The 4-way handshake

takes place in four steps:

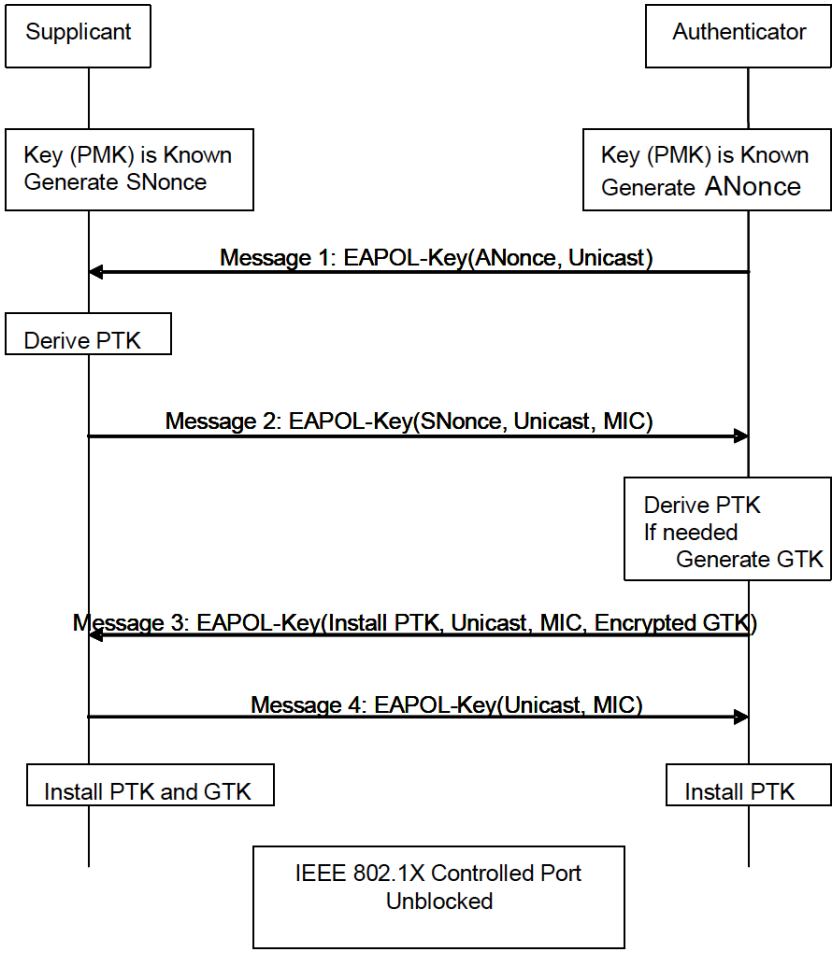

Figure 1.4: The 4-way handshake, the figure was taken from [4].

1. At first, the authenticator sends an EAPOL-Key frame *message 1*. This message contains random generated ANonce and has the replay counter of r.

2. Then, the supplicant derives a PTK from ANonce and SNonce and sends an EAPOL-Key frame *message 2* containing SNonce. This message also contains information from the (Re)Association Request frame about pairwise and group cipher suites supported. Besides, it contains MIC and replay counter r.

3. Now the authenticator derives PTK from ANonce and SNonce and validates the MIC in the received EAPOL-Key frame *message 2*. After the validation, the authenticator sends an EAPOL-Key frame *message 3* again containing the same ANonce as in *message 1*. It also contains information from its Beacon or Probe Response messages about pairwise and group cipher suites supported. Besides, it contains MIC, information whether to install the temporal keys and the encapsulated GTK. The message contains replay counter $r + 1$.

4. Finally, the supplicant sends an EAPOL-Key frame *message 4* to confirm that the temporal keys are successfully installed. This message also contains replay counter $r + 1$.

After completion of the 4-Way Handshake, the authenticator and supplicant have been mutually authenticated and can send encrypted general data traffic to each other. The diagram of the 4-way handshake is shown in Figure 1.4.

When a new 4-way handshake is initialized, the messages of the handshake are encrypted by the already installed PTK. Also, the nonces of the authenticator and supplicant are refreshed, and a new PTK is negotiated.

1.2.3 The Group Key Handshake

The group key handshake is also defined in the amendment IEEE 802.11i [4]. The authenticator uses this handshake to send a new GTK to the supplicant. It is done periodically; the period depends on settings of the AP. Also, the supplicant may trigger a group key handshake by sending an EAPOL-Key frame with the Request bit set to 1 and the type of the group Key bit. It can be done only after the 4-way handshake was successfully performed. Thus, all messages of the group key handshake are encrypted using the data-confidentiality protocol.

The group key handshake takes place in two steps:

1. The authenticator initiates the handshake by sending EAPOL-Key frame containing the GTK as *message 1* with a new encapsulated GTK, along with the last sequence number used with the GTK (RSC). The GTK is, as in the 4-way handshake, stored in the Key data field, thus, it is encrypted using KEK.

2. On receiving the EAPOL-Key frame, the supplicant validates the MIC, decapsulates the GTK, and installs the GTK and the RSC. The supplicant then constructs and sends an EAPOL-Key frame *message 2* in acknowledgment to the authenticator.

There are two variants when the authenticator installs the group key. First is after sending *message 1*, the second is after receiving *message 2* from all

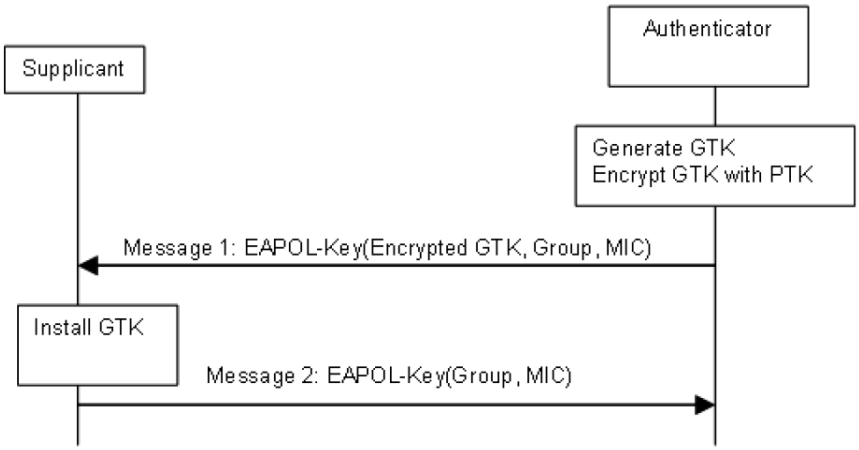

Figure 1.5: The group key handshake, the figure was taken from [4].

the supplicants. The specific variant used by a device depends on the implementation, this is further described in [6]. When a client sends a group frame (broadcast or multicast), it is first sent to the AP as a unicast frame, and then the AP encrypts it using group key and sends it to all clients. The diagram of the group key handshake is in Figure 1.5.

1.3 Security

In the base 802.11 standard published in 1997, there was defined a security algorithm for data encryption. This algorithm was called WEP. Due to the absence of the certification program at the beginning, its usage was only optional. Shortly after its release, already in 2001, the protocol was broken [1, 2, 5, 3]. In response, the IEEE developed the 802.11i standard to upgrade the encryption algorithm. In this standard, the protocols WPA and WPA2 were defined along with the 4-way and group key handshakes [4]. The WPA protocol is based on Temporary Key Integrity Protocol (TKIP), a robust encryption algorithm built around WEP. This algorithm was able to run on the same hardware as the original WEP. Thanks to this, only updating firmware was necessary to upgrade from WEP to WPA. It was certified by Wi-Fi Alliance based on a draft version of the 802.11i amendment which was still in development during that time. The protocol was meant as a temporary compensation before the amendment was finished and the Wi-Fi Alliance later deprecated the TKIP for the insufficient level of security [15]. The full implementation of the 802.11i standard is certified as WPA2, and it is built on the Counter-Mode protocol (CCMP) of the Advanced Encryption Stan-

dard (AES) algorithm. Due to the higher computational complexity of the (AES-)CCMP encryption, it was necessary to replace previously used hardware to use WPA2. The data-confidentiality and integrity protocol used is the main difference between WPA and WPA2. Otherwise, they are very similar to each other, and both use the 4-way handshake, either in personal or enterprise networks [4]. Additional data-confidentiality and integrity protocol was defined in amendment 802.11ad published in 2012. It is called Galios/Counter Mode Protocol (GCMP) [16].

In January 2018, the Wi-Fi Alliance announced the release of the WPA3 with several security improvements over WPA2 [17]. Unfortunately, WPA3 does not provide users with absolute assurance that they will not be vulnerable to KRACK attacks. Although the WPA3 protocol provides a new Dragonfly handshake, it will be used in combination with the 4-way handshake [18]. Therefore, the vulnerability of the WPA3 certified products will again depend on the specific implementation of the 4-way handshake.

1.3.1 Data Confidentiality and Integrity Protocols

Nowadays, there are three data confidentiality and integrity protocols in use (we exclude the deprecated WEP): TKIP, CCMP, and GCMP. These protocols are used for encryption of data frames. There is also an algorithm used for authentication (not encryption) of group-addressed management frames called Broadcast/multicast Integrity Protocol (BIP) but we will not deal with it in this work. The implementation of the CCMP is mandatory in all WPA2 certified devices. It also optionally supports TKIP but only for backward compatibility reasons. The WPA is exactly the other way; it mandates support for TKIP and optionally supports CCMP.

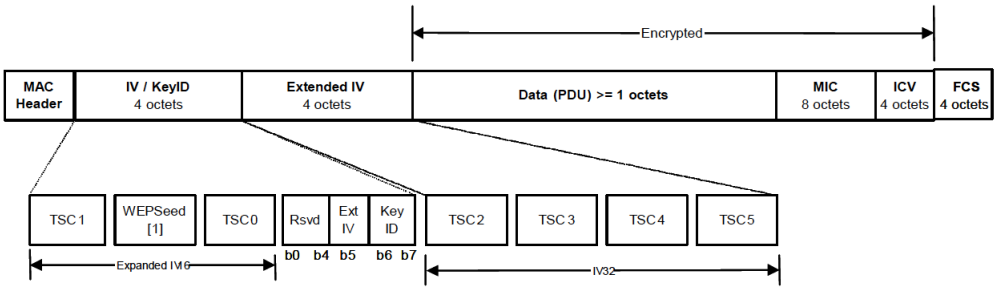

Figure 1.6: Expanded TKIP MPDU, figure taken from [4].

The TKIP was designed to be able to operate on the machines which used to use WEP and should only be used when communicating with devices that are unable or not configured for encryption using CCMP or GCMP. It uses the same cipher as older WEP, the RC4. When using TKIP, the Temporal key (TK) which is part of the session key (PTK) is further split into a 128-bit

encryption key and two 64-bit MICs. One is used for communication from an AP to a client and the second one in reverse. MIC provides a defense against forgery attacks. TKIP uses per-frame TKIP sequence counter (TSC), which is incremented after transmitting a frame and initialized to 1 when installing the TK. Besides, it is used as a replay counter. The WEP seed, which is the keystream used to encrypt the data and MIC, is a mix of the 128-bit encryption key, the sender MAC address, and an incremental TSC. This TSC is by Vanhoef denoted as nonce [6]. Message authenticity is provided by the Michael algorithm. The expanded MAC Protocol Data Unit (MPDU) using TKIP is shown in Figure 1.6.

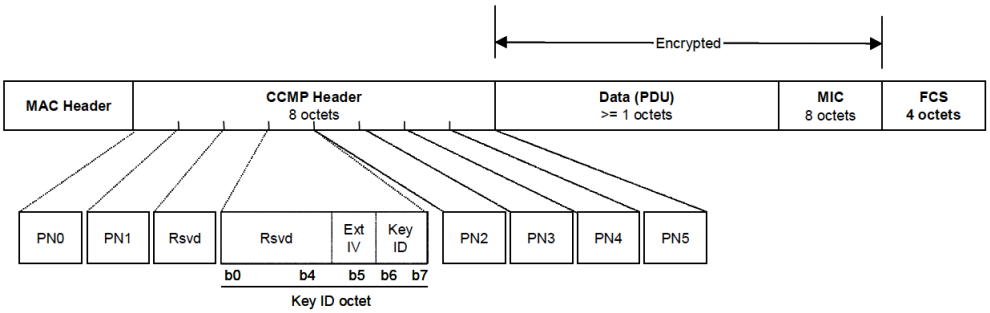

Figure 1.7: Expanded CCMP MPDU, figure taken from [4].

The CCMP is based on the Counter with CBC-MAC (CCM) mode of the AES encryption algorithm. The nonce of the CCMP is the concatenation of the sender MAC address, a 48-bit packet number (PN), and some additional flags derived from the transmitted frame. According to standard, the encryption algorithm is secure, as long as, the nonce is used only once for the same key. This nonce is by Vanhoef called the initialization vector (IV), and the packet number is what he calls the nonce [6]. The number is also used as a replay counter by the receiver, and it is initialized to 0 when installing the temporal key. The expanded MPDU when using CCMP is shown in Figure 1.7.

The GCMP is based on the AES cipher with Galois/Counter Mode. Besides, it uses GMAC for authentication and integrity. As the CCMP, GCM requires a fresh temporal key for every session and a unique nonce value for each frame protected by a given temporal key. Similarly, as the CCMP, the GCMP uses a nonce that is a concatenation of the sender MAC address and a 48-bit packet number. Also, the PN is used as a replay counter by the receiver, and it is initialized to 0 when installing the temporal key. The packet number is what is called nonce by Vanhoef [6]. The expanded MPDU when using GCMP is shown in Figure 1.8.

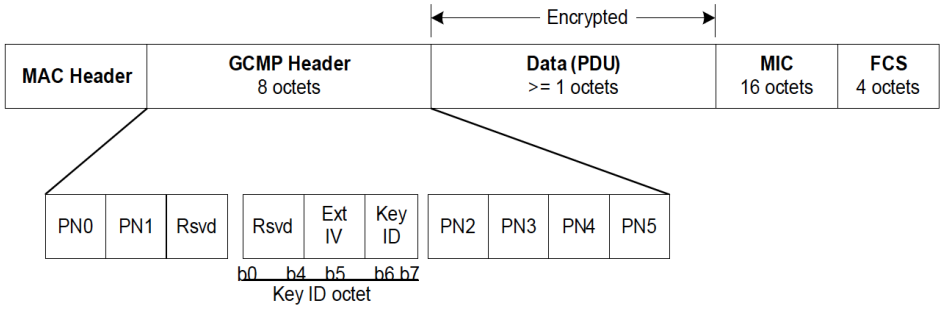

Figure 1.8: Expanded GCMP MPDU, figure taken from [16].

1.4 Other Vulnerable Handshakes

This section briefly introduces other handshakes that are vulnerable to the KRACK attack.

1.4.1 The PeerKey Handshake

This handshake was defined in IEEE 802.11-2016 [13] and is used to provide data confidentiality between the two STAs. The handshake consists of two phases. First, the SMK handshake is performed to achieve station-to-station link (STSL) master key security association (SMKSA), which is then installed in both the STAs. This message exchange goes via the AP and is protected using the PTK. The second phase is the 4-way STK handshake. This handshake runs as well as the standard 4-way handshake, except the initial authentication is done using the installed SMKSA, and as a result of this, the STKSA gets installed in both the STAs. This handshake is very rarely used, and in an IEEE report from January 2018 [19], it is marked obsolete. Its deprecation is planned in the future standard revision. Since the vulnerable phase of this handshake is the 4-way handshake which runs in the same manner as the original 4-way handshake, the diagram of this handshake will be omitted from the description.

1.4.2 The Fast BSS Transition (FT) Handshake

This handshake is defined in the amendment IEEE 802.11r-2008 [12]. It is used when the client is roaming between APs of the same network. It reduces time by setting up security and QoS parameters before reassociation to a new AP, and thus, speeds up the time-critical reassociation process. This handshake is important because it is the only handshake vulnerable on the side of the AP. If the AP does not support this standard, it cannot be vulnerable to the KRACK attacks.

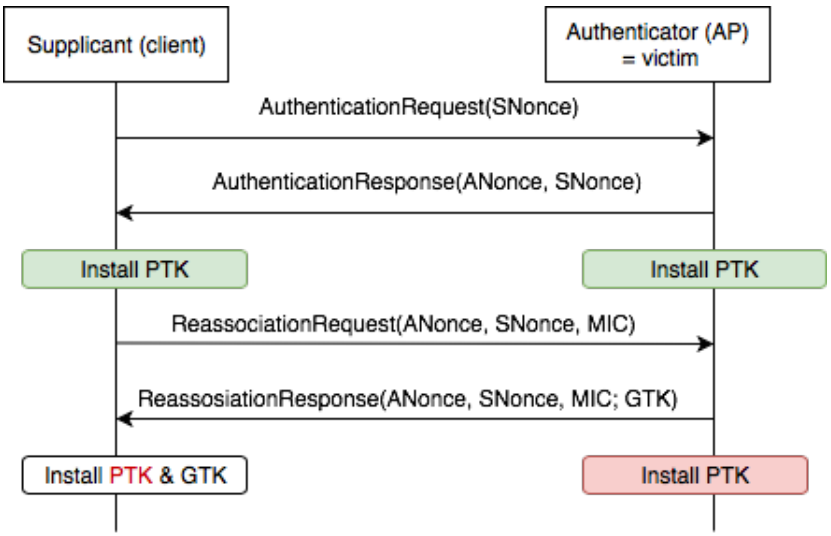

Figure 1.9: The Fast BSS Transition handshake with denoted installation of the PTK defined in standard (green — not vulnerable) and found in specific implementations by Vanhoef (red — vulnerable).

This handshake is done when the client is already connected to the network. Therefore, it relies on master keys derived during the previous connection. Functionally, it performs the same operations as the 4-way handshake but using the Authentication and (Re)Association frames. Unlike the 4-way handshake, it is initialized by the supplicant, not the authenticator. But the messages have the same purpose. The first two messages serve as authentication, the other two transport the GTK to the client. The third and fourth message is protected by MIC, but none of the messages contain a replay counter. It only relies on random ANonce and SNonce values which identify the handshake (the authenticator and the supplicant generate random nonce value for each handshake).

According to the standard, this handshake is not vulnerable to the KRACK attack, because the PTK must be installed after the authentication response is sent or received. This would be after the first two messages of the handshake, as you can see in Figure 1.9, where it is colored green. On the contrary, Vanhoef found by testing specific devices, that they install the PTK after the whole handshake. It is further discussed in [6]. This is colored red in the Figure 1.9.

1.4.3 FILS Handshake

The FILS handshake was published in the IEEE 802.11ai-2016 amendment [20]. This protocol serves to establish a secure connection to an AP and simultaneously initialize higher layer protocols.

The FILS authentication can be performed by public or shared secret key. When using a shared secret key, it could be either PMK or re-authentication Root Key (rRK) which is derived during the 802.1x. The handshake is again very similar to the 4-way handshake, but it uses Authentication Request and Response frames, following by (Re)Association Request and Response frames. The authentication frames also contain EAP-initiate/Re-auth packet which is defined in RFC 6696 [21]. The (Re)Association Request and Response again serve to confirm negotiation of the same PTK but can additionally include a DHCP request and response. The messages in the handshake do not include a replay counter.

1.4.4 TDLS PeerKey (TPK) Handshake

The TPK handshake is defined in standard IEEE 802.11z-2010 [22]. This handshake is used to establish a direct link secure tunnel between two STAs, an initiator, which initiates the link established, and a responder. Establishing of the tunnel runs via an AP. For that purpose, both the initiator and the responder have to have a secure link between them and the AP established at first. Assuming they have it, the TPK 3-way handshake can proceed. First, the initiator sends TDLS Setup Request frame with TPK handshake *message 1* included. The responder sends TDLS Setup Response frame with TPK handshake *message 2*. When the initiator receives the message, he or she installs the TPK and as a response, sends the TDLS Setup Confirm frame with TPK handshake *message 3*. The responder installs the TPK after receiving the third message of the handshake. When the direct link is established, they become TDLS peer STAs and can communicate directly with each other in a secure manner. This connection is used to directly stream data between devices, for example, between a PC and a printer or for storing data to external storage. The transmitted information between these devices might be easily confidential. The standard does not provide any state machine of the handshake, and this thesis is not going to examine this handshake further in more detail. Thus, the diagram will be omitted from description.

1.5 MAC Frames

Communication on the link layer takes place using frames in the following format (defined according IEEE 802.11-2016 [13]).

1.5.1 General 802.11 Frame

A general 802.11 frame always consists of three following basic components:

(a) A *MAC header*

(b) A variable-length *frame body*

(c) A *Frame Check Sequence (FCS)*

The format consists of a set of fields that occur in a fixed order in all frames. The Figure 1.10 depicts the general MAC frame format. The first three fields (Frame Control, Duration/ID, and Address 1) and the last field (FCS) form the minimal frame format. These fields are present in all frames which includes reserved types and subtypes. Other fields and Frame Body are present only in certain frame types and subtypes.

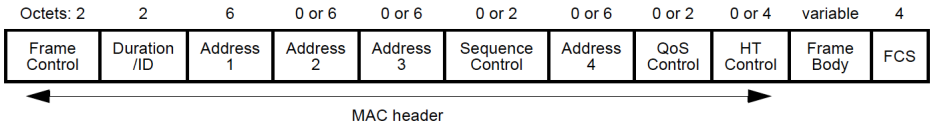

Figure 1.10: General MAC frame format, figure taken from [13].

The first three subfields of the Frame Control field, Protocol Version, Type, and Subtype, are the same for all types and subtypes of the frame. The remaining subfields of the Frame Control field depend on the combination of the Type and Subtype subfields. The structure of the frame Control field can be seen in Figure 1.11. This holds for all frames except the frame of subtype Control Frame extension, which is not relevant to this work.

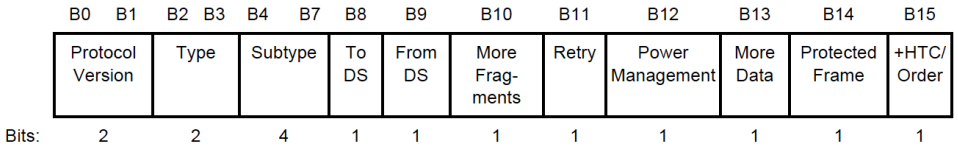

Figure 1.11: Frame Control field structure except for the Control Frame Extension subtype, figure taken from [13].

The Frame Control field contains flags about type and subtype of the frame. Additionally, it indicates the direction of the communication, if it is from the client to the DS or vice versa. Other flags designate if the packet was split into multiple frames for transmission or if it was retransmitted. Additionally, there are flags helping with power management of the stations. When a station is going to the "Power Save Mode", it notifies other devices about being in this mode by sending a Null Data frame with only Power Management bit set to 1. According to Vanhoef [8], the WNM-Sleep response frames, used by an AP to confirm or disprove the client's WNM-Sleep Request (frame indicating that client wants to enter sleep mode), can be abused to trigger key reinstallations. Another subfield then designates if the AP has buffered some frames for this device in the low power consumption mode.

The important subfield regarding encryption is the Protected Frame flag. It indicates if the frame body was encrypted by an encryption algorithm, such as WEP, WPA, or WPA2.

An 802.11 frame can have up to four address fields. According to the type and subtype of the frame, the frame fields Address 1 to Address 4 contain a combination of the following addresses:

BSSID This is an identifier of a set of devices which can communicate with each other in a wireless network. In infrastructure networks, it is the address of an AP.

Source address (SA) Source address of the device which created and sent the frame to the network.

Destination address (DA) Destination address of the device which is supposed to receive the frame.

Transmitting STA address (TA) Address of the device which sent the frame to the network.

Receiving STA address (RA) Address of the next device in the network which will receive the frame.

The address can be individual, designating a specific station on the network, or group address. The group address then can be either multicast-group address or broadcast address. Multicast address brings together a group of logically related STAs. Broadcast is predefined group address that always denoted the set of all STAs on a given LAN.

Most of the frames (depends on type) have the Sequence Control field. This field is a two-byte section consisting of Sequence Number of the frame and the Fragment Number field indicating the number of each fragment.

Each frame has the FCS. It allows for integrity check of retrieved frames. Before sending a frame, the FCS is calculated and appended as the last four bytes. A receiving station calculates the FCS and checks if they are equal with the one received. This way, the station checks that the frame was not malformed during transmission.

In the 802.11 standard, three major frame types exist:

- **Control Frames (Type bits = 01)** Control frames facilitate the exchange of data and management frames between stations. These frames, unlike other types, does not contain frame body. For example, subtype RTS is the request-to-send frame, and CTS is the clear-to-send frame which is often sent as a response to RTS. A special frame is ACK, which is sent as an acknowledge to confirm reception of a frame.

- **Management Frames (Type bits = 00)** 802.11 management frames enable stations to establish and maintain communications. These frames are used to support authentication, association, and synchronization. Some examples of these frames are Beacon frame, Probe Request, and Response, (Re)Association Request and Response, which were already mentioned. The opposite to them is the Disassociation frame which is sent to terminate the association of a station and Deauthentication frame which is sent for terminating the authentication of a station.

- **Data Frames (Type bits = 10)** Data frames carry higher-level protocol data in the frame body. The Address 1, Address 2, Address 3 and Sequence Control are present in all data frame subtypes. Data frames with a value of 1 in the QoS subfield of the Subtype subfield are collectively referred to as QoS Data frames. Each of these data subtypes contains QoS in their names, and this frame format is distinguished by the presence of a QoS Control field in the MAC header. EAPOL frame is also one of the QoS data frames, and its subtype EAPOL-Key frame is used in the 4-way, group key and PeerKey handshake. Its format will be further discussed in Section 1.5.2.

The table of existing subtypes is listed in Appendix D. The frames with the type field bits set to 11 are Reserved frames.

1.5.2 EAPOL-Key Frames

An eapol-key frame is a type of an EAPOL frame used to exchange cryptographic keying information between supplicants and authenticators. The format of an EAPOL-Key frame is shown in Figure 1.12.

Important characteristics of the key are determined by flags which are stored in field Key Information. Its structure is shown in Figure 1.13 and its subfields are described below:

Key type: If set to 1, the frame is part of the 4-way handshake (deriving PTK). If not, it is part of the group key handshake (distributing GTK).

Install: Determines if the key should be installed by the receiving station or not.

Key ack: If set to 1, the message is from the authenticator, and so, it requires acknowledgment message. The acknowledgment message should use the same replay counter as this message.

MIC: Determines if the frame is protected by MIC or not.

Secure: Both the authenticator and the supplicant set it to 1 when both the PTK & GTK were sent (thus, the initial key exchange is completed).

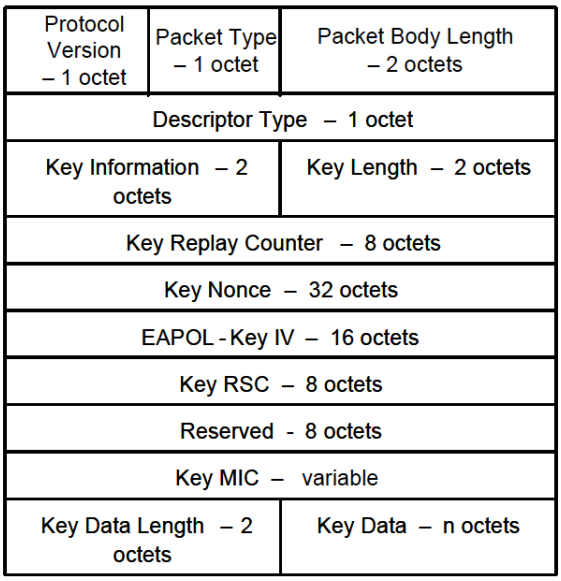

Figure 1.12: EAPOL-Key frame structure, figure taken from [13].

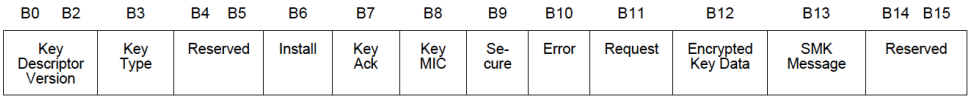

Figure 1.13: Key Information structure — Bx designate individual bits, figure taken from [13].

Request: The supplicant sets it to 1 when it wants to initialize a new 4-way or group key handshake or in case of MIC check failure. The authenticator never sets it to 1.

Encrypted Key Data: It is set to 1 in case Key Data field is encrypted. This happens in the case when GTK or STK keys are present in the Key Data field.

SMK message: The flag specifies whether this EAPOL frame is part of an SMK handshake (the second phase of the PeerKey handshake).

The meaning of other relevant fields is as follows:

Key Length: This field determines the length of the PTK; it depends on the cipher used for encryption.

Key Replay Counter: It is set to 0 when PTK is installed. The authenticator increments it by 1 at each successive EAPOL-Key frame. The

supplicant uses the same key replay counter in replays to the received frames.

Key Nonce: Contains ANonce (from the authenticator) or SNonce (from the supplicant).

Key RSC: Contains the received Receive Sequence Counter for the GTK being installed.

Key Data Length: It determines the length of the Key data field. If its encryption is required, it contains the length after the encryption.

Key Data: This field is a variable-length field that is used to include any additional data required for the key exchange that is not included in the fields of the EAPOL-Key frame.

If any of the fields is not required in the message, it contains 0. If the EAPOL-Key frame does not have the required form, it is dropped.

Chapter 2

The KRACK Attacks

This chapter describes the principle of the KRACK attacks vulnerability which was found and published by a Belgian researcher Mathy Vanhoef from the Catholic University of Leuven in 2016 and published a year later in October 2017 [5]. The research was also presented at the Computer and Communications Security (CCS) conference [23], and at the Black Hat Europe conference [24], both in 2017. We also introduce practical impact of KRACK attacks and countermeasures that should be taken to avoid them.

KRACK attacks is an abbreviation for Key Reinstallation Attacks. It is a family of attacks; their principle is that the adversary tricks a victim into reusing the value which is used as nonce with the same key in the encryption algorithm used to encrypt data frames. This allows the adversary to be able to replay, decrypt and possibly forge frames. These attacks target a similar vulnerability in various handshakes of the 802.11 family of standards. So far, it was shown that six handshakes are vulnerable to this attack: the 4-way, PeerKey, group key, Fast BSS Transition (FT) handshake, Tunneled direct-link setup PeerKey (TPK) handshake and Fast Initial Link Setup (FILS) (first four in [6], the last two in [7]). The assigned CVE identifiers designating found vulnerabilities are listed in Appendix C. As a proof-of-concept, the author executed a key reinstallation attack against an Android smartphone. In this demonstration, the attacker was able to decrypt all data that the victim transmitted without knowledge of the pre-shared key [25].

It is interesting how it is even possible that such a serious weakness is found in the protocol WPA2 that was for more than a decade considered to be secure. Especially when the 4-way handshake [26] and the CCMP encryption algorithm [27] were both mathematically proven to be secure. The reason is probably the vague standard definition which does not specifically states how to implement defined services. The 4-way handshake and the CCMP encryption algorithm were never formally tested in combination and so was not their implementation. A good improvement into the future might be a more precise definition of the standard but in the meaning of implementation —

something that would ensure that regarding security, all devices corresponding to the same protocol behave the same.

2.1 The Attack Principle

First, we are going to define used notation. We will use similar terminology and notation as the original research [6], except the nonce. To do not mismatch this nonce with the ANonce and SNonce values during the handshakes, we will call it PN, as it is defined in the standard 802.11 in context of CCMP and GCMP data confidentiality protocols. It also corresponds to per-frame TSC in the context of TKIP protocol. Notation in the diagrams will be as follows:

$$\text{MsgN(r, Nonce; GTK)} \tag{2.1}$$

The expression (2.1) represents message N of the 4-way handshake with a replay counter of r, and with the given nonce (if present). Parameters after the semicolon are stored in the key data field.

$$Enc_n^k\{.\} \tag{2.2}$$

The expression (2.2) denotes data are encrypted and authenticated using a data-confidentiality protocol. Here n denotes the packet number being used. The parameter k denotes the session key which is used. This is the key that is negotiated during a handshake. In the case of the 4-way handshake, it is the PTK; it is called GTK in the group key handshake, etc., as it was described in Section 1.3.

$$\text{Data(payload)} \tag{2.3}$$

The expression (2.3) denotes data transmitted in a unicast frame.

2.1.1 The Aim of the Attacker

The attacker aims to force the victim into reinstalling an already-in-use key or reuse older key (this is a bit more complicated but theoretically possible). When the victim reinstalls it, he or she also resets the incremental transmit packet number and receive packet number (i.e., replay counter). We will briefly explain why it is wrong that these variables are reused.

At first, we will show why it is wrong to reuse the packet number. For that purpose, a very simplified data encryption process by the data confidentiality and integrity protocols (TKIP, CCMP, GCMP) is shown in Figure 2.1. All these three protocols behave like stream ciphers, meaning a keystream is generated and XORed with the plaintext data.

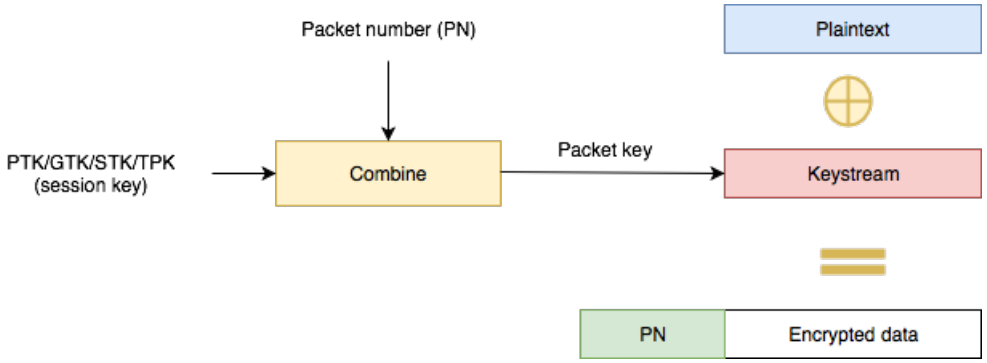

Figure 2.1: Simplified data encryption by data confidentiality and integrity protocol.

In the diagram, we can see the session key[1] is mixed with the packet number to establish a packet key. The keystream obtained depends only on the session key and the packet key. Then, keystream is XORed with the plaintext data of the data frame. Finally, we prepend the used packet number to the final packet, to let the receiver decrypt data.

Different packet numbers are used to ensure distinct ciphertexts are produced even when the same plaintext is encrypted multiple times independently with the same key. It is why Vanhoef calls this value nonce because this is precisely its role[2]. We can see in Figure 2.1, that the encrypted data depend only on these three variables — session key, packet number, and plaintext data of the frame sent. Hence, when the victim reinstalls an already-in-use session key and resets the packet number, the same combination will be used twice. It is also explicitly stated in the standard revision from 2016 [13] that packet number re-usage breaks the security assumptions of the data confidentiality and integrity protocols.

The victim also resets the replay counter. As mentioned in the 1.3, this value serves as a protection against replayed frames. Its resetting allows the attacker later replay frames towards the victim.

2.1.2 Found Vulnerability

The aim of the attacker as described in Section 2.1.1 is achieved by manipulating and replaying cryptographic handshake messages. The reason why it is possible is because of a vulnerability found in the handshake definition in the standard itself. We are going to show the vulnerability on the 4-way handshake. This attack is targeted against the client. The vulnerability of

[1] Actually, it is only its subkey called transient key (TK), but this detail is not crucial for understanding of the attack principle and could be just confusing fact for the reader.

[2] According to Merriem-Webster dictionary [28], a nonce is "value occurring, used, or made only once or for a special occasion".

this handshake is that, when the supplicant receives retransmitted *message 3*, it reinstalls the key already installed, thus, resets the packet number and the replay counter. We will show it in more detail after introducing the state machine of the supplicant as it is defined in the standard.

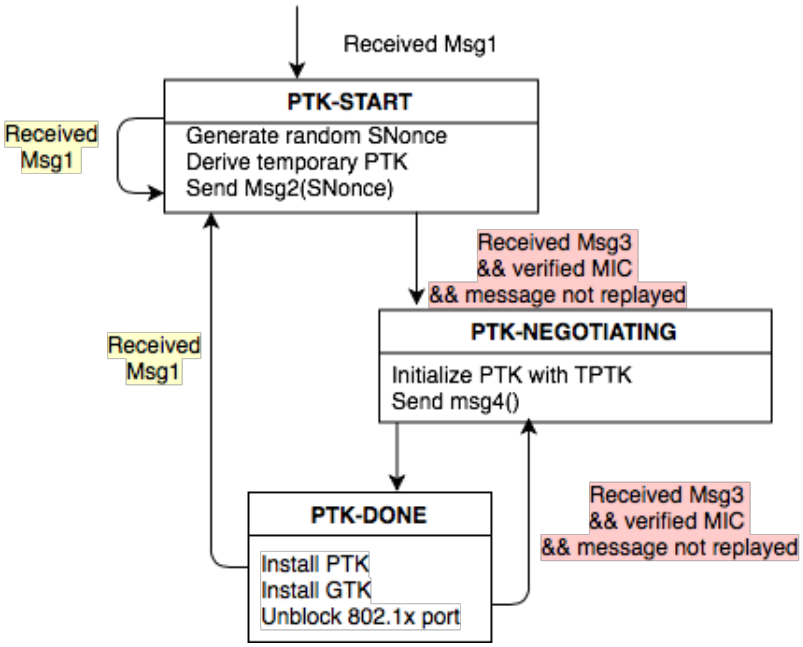

Figure 2.2: Simplified supplicant state implementation machine of the 4-way handshake.

A simplified state machine describing how the client (the supplicant) must implement the 4-way handshake is shown in Figure 2.2 — the supplicant transitions to the PTK-START stage when receives *message 1*, either when it is connected to the network for the first time, or when the session key is being refreshed. In this state, the supplicant assembles *message 2* and sends it. The authenticator will reply with *message 3*, which is accepted by the supplicant if the MIC and replay counter is valid. In this case, the supplicant moves to the PTK-NEGOTIATING state, initializes the PTK variable with the temporary PTK variable, sends *message 4* to the authenticator, and transitions to the PTK-DONE state. Now, the PTK and GTK are installed for usage by the data-confidentiality protocol; it unblocks the 802.1x port and thus, the client can send and receive encrypted data frames. [13]

In Section 2.2, we explicitly highlighted how the supplicant handles retransmitted *messages 1* (yellow) and *3* (red). The attacker uses the fact that the client accepts the *message 3* even if the key is already installed (PTK-DONE), gets back to PTK-NEGOTIATING state and immediately back to PTK-DONE state, where the supplicant installs the PTK and the GTK once

again. Thus, reset the packet number and the replay counter. We can also see that if the supplicant gets *message 1* when in PTK-DONE, it goes back to the PTK-START. This can also be used against the client as we will see when monitoring the traffic of the attack in Section 4.2.

2.1.3 Execution

We have already explained what the aim is and what is the exploited vulnerability, now we are going to describe how to achieve it. In an infrequent occasion, the vulnerability can be exploited, meaning the packet number can be reused with the same session key, by standard traffic in the Wi-Fi network. This situation can happen due to interference. In Figure 2.3, we show this hypothetical situation.

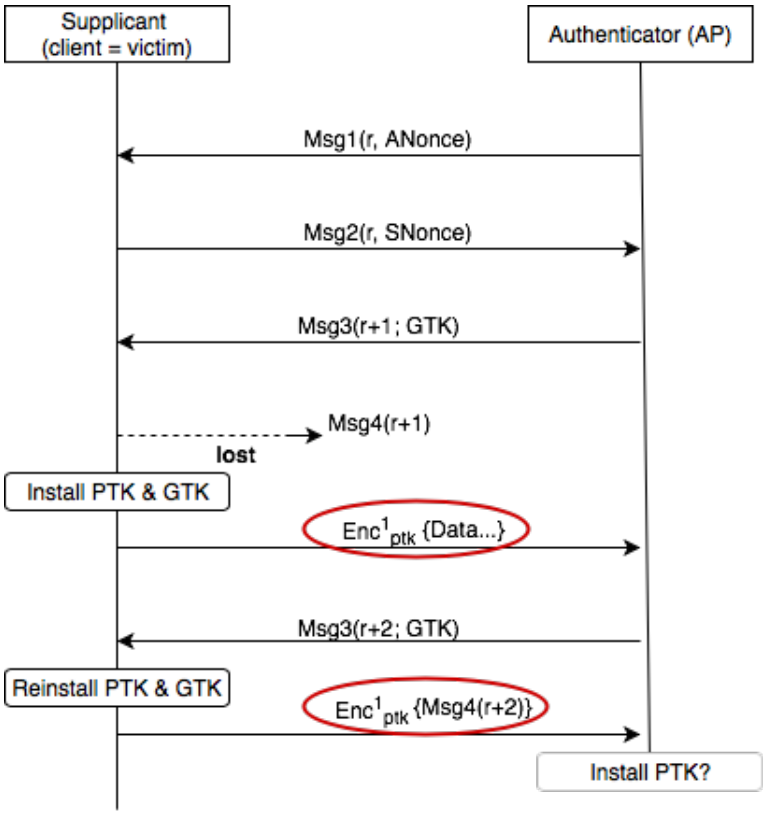

Figure 2.3: KRACK attack vulnerability exploited because of interference without an attacker being present. We assume the victim still accepts the plaintext *message 3* after the PTK is installed; messages using the same packet number are circled red.

First, the authenticator sends *message 1* to the client with replay counter r and randomly generated value ANonce. The supplicant replies to the message

by sending *message 2* with the same replay counter r and randomly generated value SNonce. Now, the authenticator derives the PTK as described in Section 1.2.2 and sends *message 3* with incremented replay counter $r+1$ and GTK encapsulated in Key data field. In response to *message 3*, the supplicant sends *message 4* with the corresponding replay counter $r+1$. The supplicant considers the handshake as completed, installs both the PTK and GTK and in context of the state machine in Figure 2.2, the victim transitions to PTK-DONE state. Unfortunately, the message is lost due to interference. The client starts sending data towards the AP in belief that the handshake was successfully finished. The client uses the packet number of value 1. But the authenticator still did not get the *message 4* of the 4-way handshake. Thus, he thinks that one of the messages *3* or *4* was lost. Therefore, he re-sends the *message 3*. Note, he used incremented replay counter $r+2$. Its incrementation is important. Otherwise, the victim would not accept the message, because it could have been sent by anyone sitting in the middle and monitoring and replaying the traffic. This replay counter is added by a WNIC driver.

Now, we describe how the attacker can trigger the reinstallation. For accomplishing this, the attacker has to, in some cases, establish a channel based Man-in-the-Middle (MitM) position. This attack was found by the same author as the KRACK attacks, in 2014 [29]. It is a low-layer attack which allows the adversary to intercept and manipulate the traffic reliably. It solves several problems with establishing a MitM in a Wi-Fi network. Creating a MitM position in a Wi-Fi network is difficult because of the 4-way handshake which negotiates the session key depending on the MAC address of both the AP and the client. Hence, if we use a rogue AP with a different MAC address, the handshake will fail. If we use the real address, the AP and the client would directly communicate with each other. The attack solves these problems by cloning the real AP to a different channel. To perform the attack, we need two Wi-Fi network cards, at least one external. One of them communicates on the channel of the real AP, the other cloning the AP on a different channel. For the implementation of this attack, it is necessary to patch firmware of the wireless network interface card used as a rogue AP. After establishing this position, the attacker can capture, manipulate and block traffic between the client and the real AP. It does not let the adversary decrypt any captured data. Besides other things that had to be patched in the driver, to implement this attack, it was necessary not to let the rogue AP increment the replay counter when sending messages [29]. This is important for the implementation of the KRACK attacks.

When we established the channel based MitM position, we can manipulate and block the data transmitted between the AP and the victim (client). We again assume a client that accepts plaintext retransmissions of *message 3* in the 4-way handshake, when a PTK is already installed. In Figure 2.4, we can see several changes. There is now the MitM obtained, the communication with the client on the left is transmitted on a different channel than the

communication on the right between the attacker and the real AP. Again, there is shown the 4-way handshake. The principle is the same as in the previous example where no attacker was present except the fact that now the communication goes through the MitM. The MitM only captures and forwards the first three messages and stores these messages to be able to use them later in the communication.

When *message 4* comes from the client, the adversary blocks it, meaning she does not send it further to the real AP. The victim again thinks the 4-way handshake was successfully finished and starts transmitting data. Notice, the victim uses packet number 1 after installation of the PTK and GTK. The AP did not get the acknowledgment in the form of *message 4*. Thus, it resends the *message 3* with incremented replay counter towards the MitM, and it forwards it towards the victim. Because the PTK has been installed, the victim answers with the encrypted *message 4* in response and reinstalls the PTK (according to the state machine in Figure 2.2). Next data the victim will send are going to have again the packet number 1.

Additionally, in this particular case, it is easy to decrypt the data because we captured similar (not the same) encrypted and plaintext messages. A possible way of decrypting these data is shown in Figure 2.4. We can see, that in case a message that reuses keystream has known content, it becomes trivial to derive the used keystream. When there is no known content, it is harder to decrypt packets. To do it, we can try to identify packets with known content, for example by their usual length (e.g., DHCP packets).

2.1.4 Another variants

There is also a different variant of the attack against the 4-way handshake. The option depends on the acceptance of the *message 3* by the client. Specifically, it depends if the client accepts the plaintext *message 3* or only encrypted one after installation of the PTK. The process stays the same, but for the implementation, it is necessary to encrypt some captured frames. These specific variants depend on the firmware implementation of the victim. Vanhoef provides a very detailed explanation of different variants of the attack dependent on the handshake attacked and on the implementation of the firmware [6, 7].

The attack can be targeted either to the AP, in case it is against the the Fast BSS Transition handshake, or the client.

The attack on each specific handshake is a bit different, but the principle remains the same. The attacker always wants to trick the victim into reinstalling the negotiated or distributed (in case of group key handshake) session key. This can be done by manipulating messages. In some cases, it is necessary to establish the channel based MitM position. When handshake messages do not contain the replay counter, this position is not needed. This situation arises when attacking the Fast BSS Transition and the Fast Initial Link Setup handshake where Authentication and (Re)Association frames are

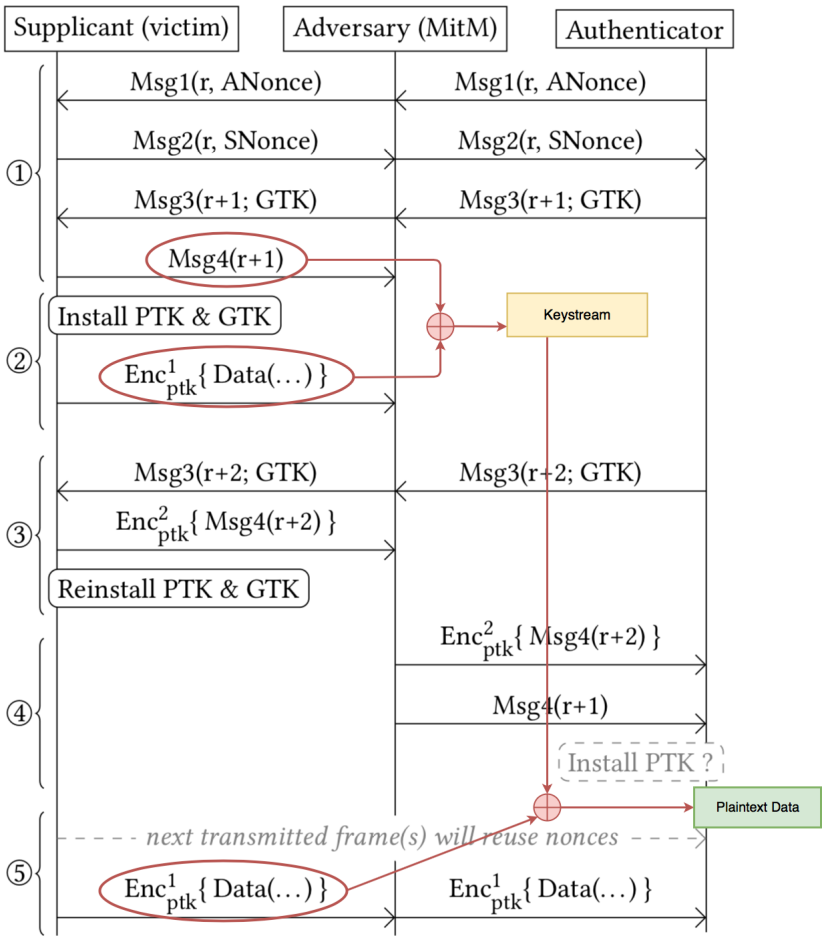

Figure 2.4: Key reinstallation attack against the 4-way handshake, when the supplicant (victim) still accepts plaintext retransmissions of message 3 if a PTK is installed, possible decryption of data is designated.

used. Assuming we have the position established (if necessary). Now, we need to monitor data on the channel (or channels, in case of MitM) and store messages that can help us trick the victim to reinstall the session key later in the communication. For this purpose, we primarily need the message that triggers the installation on the side of the victim. In case of the 4-way or PeerKey handshakes, it is *message 3*. In case of the group key handshake, it is *message 1*. *Message 4* or *Message 2* respectively, from the client to the AP then has to be blocked to trick the AP to send another *message 3* or *message 1* respectively, with an incremented replay counter. In case of the Fast BSS Transition handshake, it is the Reassociation Request that we need to store for later re-sending towards the victim. It is similar for remaining

vulnerable handshakes. We also have to store the message with reset packet number to be able to decrypt other messages with the same PN used.

2.2 Practical impact

In the time of publication of the exploits, there was a huge number of vulnerable devices. This was because the vulnerability was discovered in the standard itself. The highest impact regarding the number of devices had the 4-way handshake since it is used to mutually authenticate a client and an AP in both WPA and WPA2 protocols. The vulnerability was either in APs or clients. Fortunately, the AP is affected only if supports the standard 802.11r which was released quite recently in 2016. Besides, both personal and enterprise networks were affected. The list of affected vendors is available in [30] and a community maintained list of available patches in [31].

It is also convenient to point out what the attack does not do (by itself). The attack never reveals the WPA pre-shared key or does not compromise corporate credentials via EAP. Besides, when a proper method of encrypted communication is used on a higher layer (VPN, HTTPS, SSH, etc.), it cannot decrypt these data. We assume these services are properly configured. However, Vanhoef [5] warns that this additional layer of protection can be bypassed in non-browser software: in Apple's iOS and OS X [32], Android [33, 34] and VPN [35]. Some of the links provided by the author about this topic were unavailable or out-dated, which may cause some doubt to the reader. Nevertheless, the attack can be used as a base layer for another attack. For example, by decrypting TCP SYN packet, the TCP connection can be hijacked, and malware can be injected into the unencrypted HTTP connections [5].

The specific impact of the key reinstallation depends on the attacked data confidentiality and integrity protocol, handshake, and a specific implementation on the side of the victim [6].

2.2.1 Attacked Data Confidentiality Protocol

All three used data confidentiality protocols (TKIP, CCMP, and GCMP) are affected by possible reuse of the keystream which can lead to decryption of the packets. Also, they are vulnerable to replay attacks. The principle of how this is achieved is described in Section 2.1. The possibility of forging frames is given by the attacked encryption algorithm. When TKIP is used, and we decrypt the packet, we are also able to recover the MIC by attacking the weak Michael algorithm used to authenticate TKIP data frames. When we get the plaintext frame and its encrypted MIC value, we can recover the MIC key [36]. Different MIC is used in each communication direction. Thus, the direction in which we can forge frames depends on the attacked handshake [6]. When CCMP is used, it is only possible to decrypt and replay frames. There are only theoretical attacks that cannot lead to forging packets when reusing

the nonce [37]. When GCMP is used, besides decryption and replay, we can forge frames in both directions. It is possible because we can recover the authentication key which is used in both directions [38].

2.2.2 Attacked Handshake

The direction in which packets can be decrypted (and possibly forged) depends on the handshake being attacked. When the 4-way handshake is attacked (and so, the client), it is possible to decrypt and forge frames sent by the client to the AP. Also, it is possible to replay group addressed frames towards the victim. However, when we attack the AP by attacking the FT handshake, we can decrypt, replay or/and forge frames in the opposite direction. [6]

2.2.3 Specific Implementation

The most vulnerable devices where using wpa_supplicant of version 2.4 and above. This issue is further described in the original research [6]. This open source software implementation of the supplicant is used on Linux and Android 6.0 and above. It turned out this implementation could be tricked into reinstalling an all-zero key, allowing full decryption of all data sent by the client.

Also, it was found that the implementation of iOS and Windows operating systems did not properly follow the standard and thus, they were not vulnerable to PTK reinstallation.

In different implementation specifics, there were no such differences, and all the small nuances are precisely described in the original research [6, 7].

2.3 Countermeasures

We can look at the countermeasures necessary to be taken from two perspectives. The first is from the affected user — the other one then from the developer.

2.3.1 User Point of View

To be protected against the attack, it is necessary to update affected devices. There are lists of available patches to check if the specific device has been patched or not [31]. As it has been said, an AP is vulnerable only if it supports standard 802.11r. Otherwise, it does not have to be updated. To be protected against all other KRACK attacks, it is necessary to patch the vulnerable client. The AP can be patched to protect a client side too. But if the client connects to another network and the AP does not also have the setting to protect clients, the client is still vulnerable. This protection on the side of the

AP is based on the prohibition of the retransmission of message 3 of the 4-way handshake. One of the APs running with this protection setting available is with operating system OpenWrt. However, this setting can lead to worse performance of the network, especially in case of a place with high interference.

2.3.2 Developer Point of View

From the developer point of view, the main countermeasure that should take place is to prevent reinstallation of the key that was already used with the same session key or is in use right now. Vanhoef proposes two mitigation techniques. The first should be taken by the entity implementing the data-confidentiality protocol. This entity should check whether an already-in-use key is being installed and in such a case, take care that associated nonces and replay counters are not reset. In this case, it is necessary to increase the replay counter of received group key handshake messages. Otherwise, an adversary can use an old group *message 1* to make a victim temporarily install an old (different) key, to subsequently reinstall the current group key using a more recent group *message 1*. [6] A second proposed solution is to add a boolean variable to the state machine illustrated in Figure 2.2. This variable would be initialized to false and set to true when a new PTK in PTK-START state is generated. If the supplicant is entering the PTK-DONE and the variable is true, the PTK is installed, and the boolean is set back to false. Otherwise, the installation of the PTK is skipped. [6] The author also notified vendors about the vulnerability and helped to develop or design some implemented countermeasures [5].

Chapter 3

Traffic Monitoring

A very important part of the thesis was to monitor the Wi-Fi traffic. Also, it was a prerequisite for all the following parts. Therefore, in this chapter, we also explain some facts that might be expected in the design section. We define hardware and software used for traffic monitoring along with the environment where we monitored the traffic. It is an important variable that influences Wi-Fi performance. At the end of the chapter, we will sum up all characteristics found based on both theoretical studies of the attack and practical traffic monitoring. We will refer to these conclusions and definitions in the following chapters.

3.1 Prerequisites

To be able to analyze the traffic in the Wi-Fi network, we need to be able to set our Wireless Interface Card (WNIC) into monitor mode. Ideally, we want to find a way how to cover the whole spectrum of channels in a band.

3.1.1 Monitor Mode

The monitor mode allows watching all Wi-Fi traffic on a specific channel without associating to an access point [39]. This mode is necessary to monitor 802.11 layer management and control frames. Also, the data frames would be encapsulated into a ”fake” Ethernet packets. This layer is added by a wireless card driver [39]. Some commonly available wireless network cards can be switched to the monitor mode, but it is only possible with an appropriate driver. The availability of the driver depends on the combination of the OS and the chipset of the wireless card. The monitor mode does not change other characteristics of the card. Hence, with one card, it is possible to reliably capture traffic only on one channel at a time [39].

3.1.2 Hardware and Software

As it has been mentioned, it is necessary to plug in the Wi-Fi card and control it by driver compatible with the OS. All current operating systems have support for wireless networks including tools and interfaces for network adapter control. For this work, we predominantly used Kali Linux operating system. Excellent overview of the options with other operating systems is available in [39] and [40].

The availability of the drivers depends on the chipset of the wireless network card. Appropriate chipsets for Kali Linux are Atheros AR9271, Ralink RT3070, Ralink RT3572, Realtek 8187L, and Realtek RTL8812AU. If the built-in wireless network card does not have any of these chipsets, it is possible to use some Wi-Fi USB adapter. According to a few community bloggers, an Atheros AR9271 chipset is a decent choice for Kali Linux [41, 42]. Hence, chose to use devices with this chipset. Very widespread used adapter with this chipset is the TP-LINK TL-WN722N v1. Unfortunately, other versions of this adapter (v2 and v3) have different chipsets, and the first version is not available anymore. Thus, we decided to use several Alfa AWUS036NHA adapters. The driver for this chipset is available in the kernel of the Kali Linux; and it is called `ath9k_htc`. There are many command line tools that can be used for control of the wireless adapter, some of them are `airmon-ng`, `iw`, `iwconfig` and `iwlist`. To see the captured traffic, we can use programs `wireshark`, `tshark`, and `tcpdump`. All of them use the `libpcap` library.

3.1.3 Monitoring Multiple Channels

The KRACK attack is targeted against both — an AP and a client. Although we decided to focus on the attack on the 4-way handshake, we aimed to design such a tool that can be later extended by detection of other attacks of the family. Therefore, the system has to be independent of other devices in the network. It means that it has to be able to monitor the whole spectrum of channels. The Wi-Fi card can monitor on more channels by the so-called channel switching mechanism. It means it switches periodically between different channels. This mechanism is usually used for detecting the amount of traffic on different channels to find the least busy one [43, 44, 45, 46]. It is used to set the AP to a channel with the least interference and so, increase the speed of the connection. We need to monitor the channel for the whole time a handshake is in progress. Thus, this mechanism is not convenient for our solution. It implicates that the amount of channels we cover with our solution depends on the amount of Wi-Fi cards we will use. Also, we find out, it is a good practice to keep a distance at least 1 meter between interfaces monitoring on different channels. Otherwise, it might interfere with each other.

3.2 Environment

In this section, we are going to specify the environment in which we were monitoring the traffic. The traffic was monitored in two different environments with a different level of interference.

I. On campus, building A, i.e., many Wi-Fi devices, very high interference. Data measured during the semester, during the day.

II. A house with more housing units, four flours, second floor, two-room apartment, data measured during the day, less interference.

3.3 Standard 4-way Handshake

First, the standard traffic of the 4-way handshake was monitored. In Figure 3.1, there is an example of the captured 4-way handshake in program wireshark. This handshake was captured without any anomalies. We filtered it from other captured traffic by using `eapol` filter. In this specific example, there was no other handshake captured. Thus, no additional filtering was necessary. The type of the frame is QoS Data and used protocol is EAPOL. In the Info field, we can see the order of the messages. Wireshark identified the number of each message by used flags in Key Description field and by information contained in the frame. We will also need to watch field Replay counter, as explained in Section 2.1 and WPA Key Nonce which will help us uniquely identify each 4-way handshake between the same pair of devices. Also, we can see in the Address fields Source and Destination, that this handshake took place between the AP (MAC address `Comtrend_f9:77:f7`) and the client (`Apple_a5:99:ca`). Below the list of frames, we can see details of the *message 3* of the handshake. Specifically, the 802.1x layer, which is expanded, contains the WPA Key Nonce. Below, we can see the raw payload that was parsed by wireshark to get all this information.

The 4-way handshake shown in Section 3.1 is perfectly corresponding to the standard, but during the analysis of the traffic, two anomalies were often detected. It often happened that one of the messages of the handshake was lost due to interference. Indeed, in the network on campus. The other detected non-standard behavior is often immediate repetition of *message 1* of the 4-way handshake. Thus, these sequences of messages will be considered normal, non-malicious behavior.

3.4 Malicious 4-way Handshake

For the study of the traffic generated during the attack, we used scripts [47] that author published as a proof-of-concept. It was necessary to modify them

to be able to successfully perform the attack. Also, we studied publicly available scripts [47] made for detecting a device vulnerability to this attack. There is a *.pcap* file captured during the attack in Figure 3.2. We filtered the pair of the AP and the client performing the handshake. For the sake of clarity, we have filtered out Action and Null function frames. Note, that you can see there is the same ANonce in all messages 1 and 3 in the figure. This means it is still the same handshake and the same PTK is negotiated. During monitoring of the attack traffic, we found several often or always occurring characteristics:

Retransmitted message 3: The most important characteristics of the attack is retransmitted *message 3* of the handshake. We can see the frame with No. 1954 in Figure 3.2. In case this happens, we should evaluate the situation as a possible KRACK attack in the network.

Repeated message 1 after message 3: The attacker sometimes tries to abuse the vulnerability of the supplicant by sending *message 1* before retransmitting *message 3*. We can see this as frame No. 1952 in Figure 3.2. This vulnerability corresponds to the state machine of the vulnerable supplicant in Figure 2.2. Also, Vanhoef states [6] that it is a necessary step when attacking specific devices. However, this characteristics does not mean the attack is completed.

Reset of the packet number after retransmitted message 3: This is the main point of the attack. If we can capture this, this means the frames can be decrypted, either it happened due to the interference, or it was triggered by an attacker. We can see this in Figure 3.2 as frames No. 1964 and No. 1973. They both use the same packet number, in this case, the CCMP encryption algorithm is used, so the packet number corresponds to CCMP Ext. Intialization Vector. We are not always able to capture the first data frame to confirm the hypothesis.

Very high occurrence of message 3: Monitoring traffic during the attack, it was found that there is a much higher occurrence of the *message 3* of the handshake in the traffic. We can see this in Figure 3.2 as frames No. 1146, 1147, 1226, 1227, 1311, 1312. Even though they do not have to lead to a successful exploit of the vulnerability, their occurrence very likely means the presence of an attacker in the network or very high interference of the environment. We can distinguish between them by counting ration of captured *messages 1* to *messages 3* in the same handshake. As we stated in 3.3, the repetition of *message 1* is a standard behavior. Thus, when there are significantly more messages 3 then messages 1 of the handshake, it is likely that an attacker is trying to exploit this vulnerability.

Frequent failure of the handshake: Another characteristic that can make us suspicious is the number of handshakes that fail. When the attacker

is performing the attack, there are many handshakes that are triggered again during the handshake. Thus, they fail.

3.5 Characteristics For Detection

This section provides a summary of all found characteristics we could use for detection of the KRACK attack against the 4-way handshake.

3.5.1 Based on the Analysis of the Captured Traffic

Based on Section 3.3 and Section 3.4, we found several characteristics for the detection. There are two points of view of how to look at it. The first one is by searching concrete sequences of messages of the handshake. The second is by establishing characteristics either for the environment in general or for each handshake by using statistical methods.

Concrete Sequence of Messages

We can look at the concrete sequence of messages that we see in both malicious and standard communication and create a state machine that would have an acceptance state in case the attack was performed.

General Characteristics of the Environment

Another option would be to create a statistical model of the situation. Based on the monitored traffic, we can create a model of a standard handshake with all the anomalies found in normal communication. Then, we can detect anomalies against it. Good characteristics to use in such a model would be the frequency of each message of the handshake (e.g., a high occurrence of *message 3* in the communication). To use it, we would have to find a threshold, which can significantly differ in networks with different level of interference. It would not be as reliable as a concrete sequence of messages.

3.5.2 Based on Theoretical Study of the Attack

The detection would also be possible by characteristics found in the theoretical description of the attack.

One option available for detection of the attack against the 4-way handshake is detecting the channel based MitM position. In the research which introduced this attack in 2014 [48], the author states, the attack could be detected on the client side, by including the channel of the AP in the 4-way handshake. Unfortunately, this would require a change of the protocol itself because the channel is not included in the frame. Interesting question for future research is if the detection of channel based MitM would be possible with monitoring devices on multiple channels. But it is beyond the scope of

this work. There are some products in the market dealing with detection of MitM in the Wi-Fi network, for example, [49].

According to the vulnerabilities found following characteristics should be exactly what leads to a successful exploit. Since the Wi-Fi network is very prone to the interference, some of the messages can be lost even if it is successfully transmitted between the attacker and the victim. Thus, when using only these characteristics, some attacks might be missed completely.

the 4-way handshake: Retransmitted *message 3* of the 4-way handshake with incremented replay counter. Then, the client will receive *message 3* multiple times and because of the increased replay counter, he or she should accept it. If the next sent data frame from the client to the AP has lower packet number than the previous packets, the packet number was reset, and so the PTK reinstalled.

the PeerKey handshake: The 4-way handshake used in the second phase of the PeerKey handshake is attacked in the same manner. Thus, for detection this attack, we can use the same techniques as for detection of the attack to the 4-way handshake. The difference is only in used flags which identify used keys and the handshake.

Group key handshake: Retransmitted *message 3* of the group key handshake from AP and reinitializing of the replay counter of the GTK at the client side.

The Fast BSS Transition Handshake (FT) Handshake: This is the only attack against the AP. Looking for retransmitted Reassociation request which serves as *message 3* of the FT handshake. Following reset of the packet number and replay counter in next sent data frame means certain to exploit.

Fast Initial Link Setup handshake: Again, we can look for replayed (Re)Association requests and following reinstallation of the PTK and re-usage of packet number.

TDLS PeerKey handshake: In case of this handshake, we need to detect retransmitted TDLS Setup Response frame and following reuse of packet number after completion of the handshake.

WNM-Sleep response frames: These frames can be abused to trigger key reinstallations. It would be a good practice to detect retransmitted WNM-Sleep exit frames to avoid it. How these Null frames are used to trigger key reinstallation is further explained in [7].

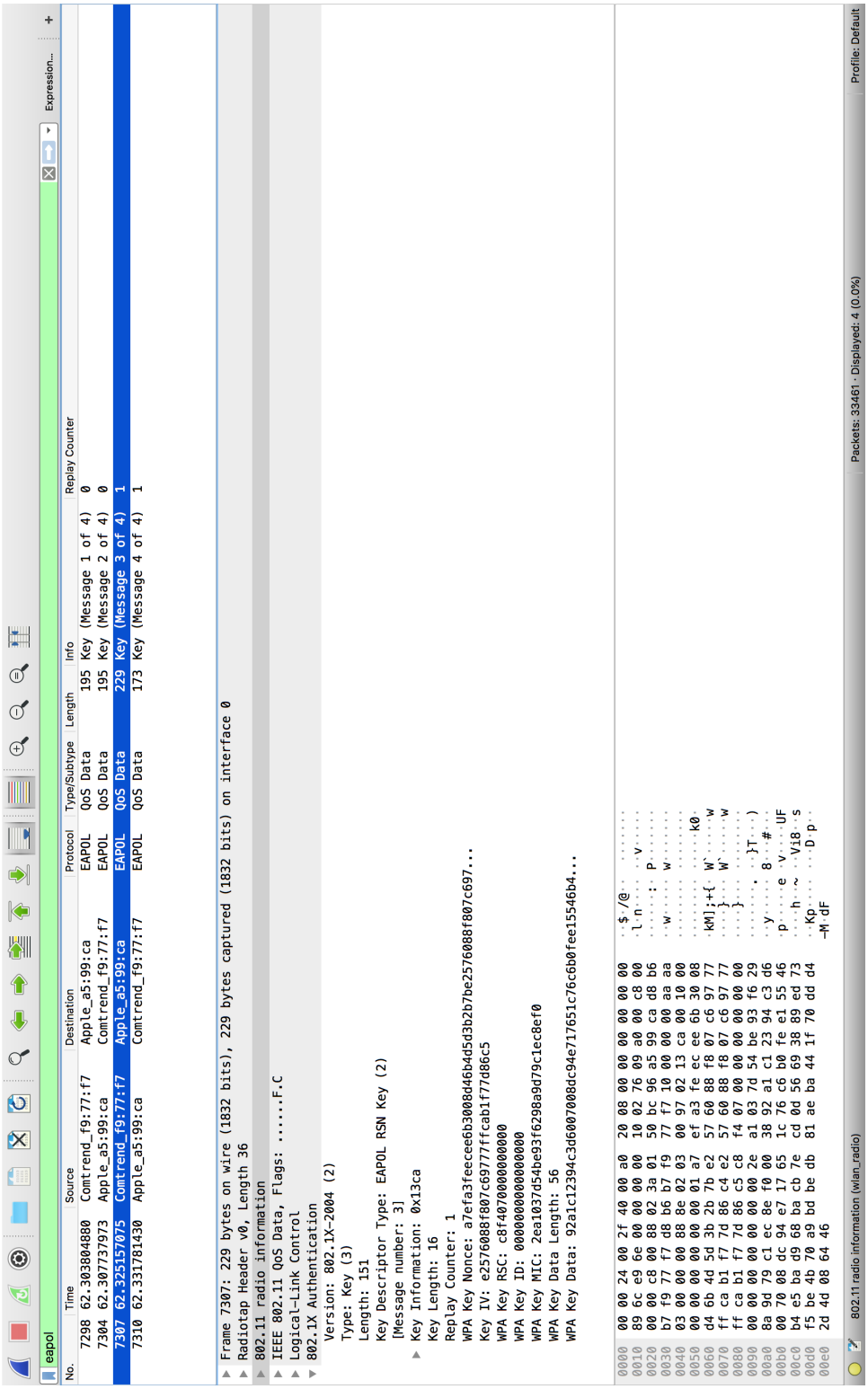

Figure 3.1: The 4-way handshake captured in wireshark.

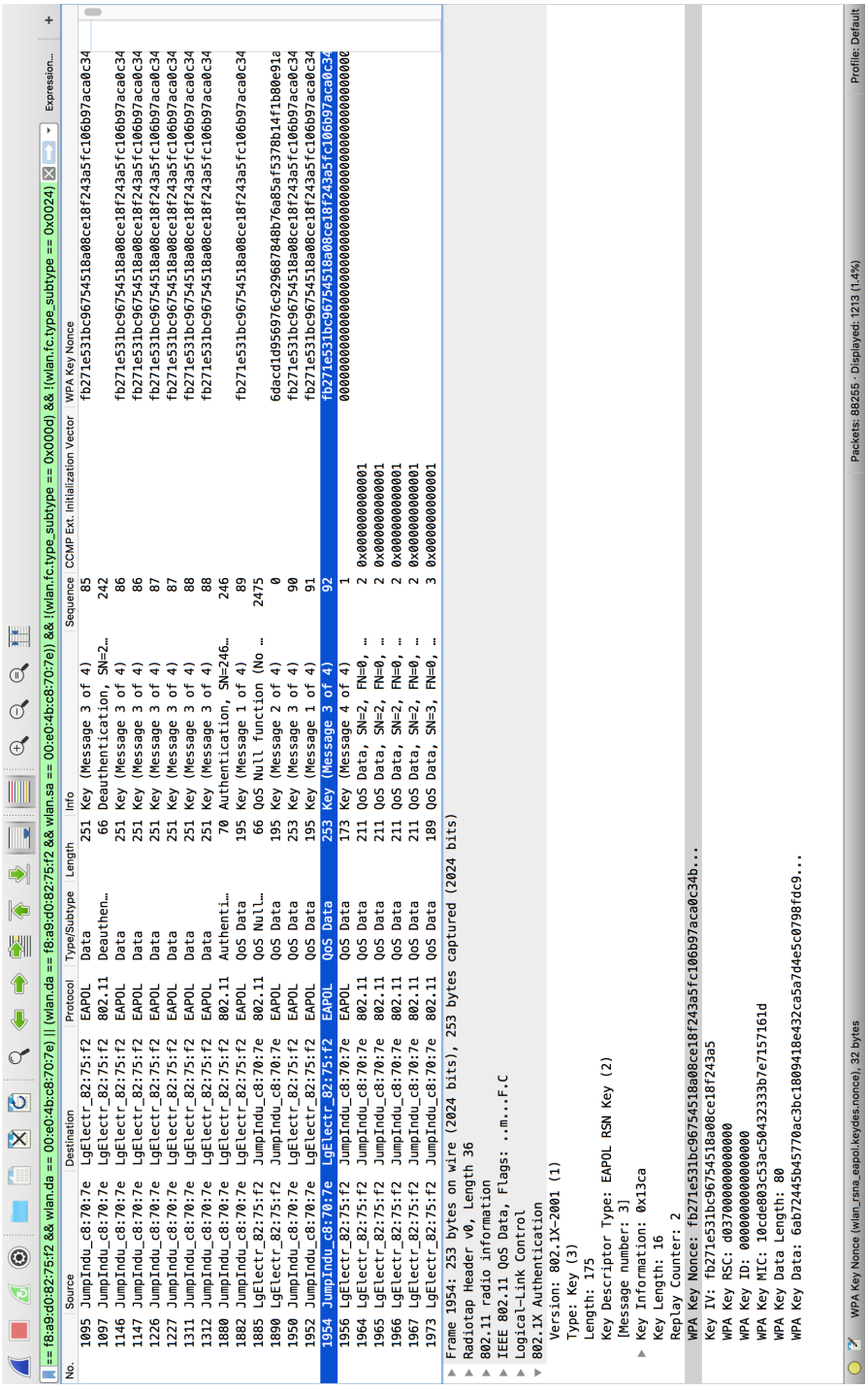

Figure 3.2: Traffic generated by successful KRACK attack against the 4-way hanshake.

Chapter 4

Design

The main aim of this chapter is to design a system for the detection of the KRACK attacks in real-time. First, we propose how a testing tool for detection of a device vulnerability should work.

4.1 Testing Device Vulnerability

Part of the assignment of the thesis was to propose a hardware or a software testing tool to detect device vulnerability to the KRACK attacks. Because the whole problem can be solved by a software tool with the necessary hardware used, we will omit possible full hardware solutions from the analysis. For testing the vulnerability, it is essential to partly or wholly perform the attack and monitor the behavior of the tested device. The principle of the attack is already described in section 2.1. Thus, we assume that the reader is now familiar with it. By tested device, we mean any wireless card in combination with the OS and the driver used. We point this out because some wireless cards might in rare cases use a different driver on the same system which might cause various behavior. We will again focus on testing the vulnerability of the 4-way handshake. Then, we will briefly discuss other handshakes.

First, we present the requirements of the tool on hardware and software. It is again necessary to have a WNIC (other than the one tested) and an OS of such combination that we can enable WNIC to be set to monitor mode. This wireless card also has to be configurable as an AP because we will run it as one. This AP simulates the rogue AP in an attacked network. Also, the AP will run our evaluating tool which should consist of a few components:

1. **L2 Socket**: This component can communicate on the layer 2 of the ISO/OSI model and receives data sent to a configured AP.

2. **Parser**: This component parses data from received communication. Thus, decides which type and subtype came and what are the data

in it. Its input is the received frame and the output are the information about the frame (e.g., it is a *message 1* of the 4-way handshake containing nonce *xy*, sent from the authenticator with a MAC address *a* to the supplicant with a MAC address *b*, etc.).

3. **Evaluator**: This component evaluates the behavior of the tested device based on the output from the parser.

4. **Logger**: This component alerts the user about results.

The tested device connects to our rogue AP. It does not necessarily have to forward the Internet to the client. Its main goal is to behave as an attacker and watch the answers of the client. This scenario is much easier to accomplish than the real attack because we provide the pre-shared key to the network and no MitM position is necessary to be established. The whole process of tricking the client into connection to the rogue AP, which is the hardest part, is omitted when testing device vulnerability. The AP behaves like a rogue AP regarding retransmitting *message 3* of the handshake when the supplicant is in state PTK-DONE according to the state machine in Figure 2.2. Based on the behavior of the client, we decide if it is vulnerable or not. To confirm the client is not vulnerable, we need to verify, that it does not reinstall the PTK (or another session key— according to used handshake) and that it does not reset replay counter and packet number, as described in Chapter 2.

Testing of the client side would be also similar to other handshakes. When testing the vulnerability of an AP, we would have to simulate a client which can run the evaluator component. Again, the client behaves as an attacker, and we only verify, how the victim reacts.

4.1.1 Existing Solution

There are only a few testing tools available for testing this vulnerability.

Official Testing Scripts

These scripts were published by the author of the attack and are available in [50]. They work on the principle described above meaning they simulate the rogue AP. They can be run on Kali Linux, and some users were able to run it on Ubuntu 16.04. There are a couple of prerequisites necessary to be installed prior to the execution. The scripts support a few tests each testing a different vulnerability in the 4-way, group key and Fast BSS Transition handshakes. For testing the vulnerability of the Fast BSS Transition handshake, it is necessary to have additional AP supporting 802.11r than the one tested. In these scripts, the author uses wpa_supplicant as a rogue client and hostapd as a rogue AP. We used these scripts to test device vulnerability. The results are listed in Chapter 6 in Table 6.1.

Wi-Fi Alliance Scripts

These scripts are based on the scripts by Vanhoef, and their capabilities are overlapping, as described in [50]. Unfortunately, they are available only for Wi-Fi Alliance members; more information is in [51].

Tests Implemented in *hostapd* and *wpa_supplicant*

Hostapd (more information: [52]) is a user space daemon software enabling a network interface card to act as an access point. wpa_supplicant (more information: [53]) is a free software implementation of an IEEE 802.11i supplicant. They include a number of extensions that allow special tests built to be used for testing functionality related to correct implementation of IEEE 802.11. For example, there are tests of the reinstallation of the GTK, PTK. Also, there is a test for FT Reassociation Request frame retransmission on an AP device. The documentation and source code can be found [54].

4.2 Detection system

In this section, we are going to design the system for detection of the KRACK attack against the 4-way handshake. We assume that the reader is now familiar with hardware and software requirements for network monitoring (as described in Chapter 3) and that he knows the principle of the KRACK attack (as explained in Chapter 2).

We set out to design a detection mechanism that will detect an attack on an independent device in the network. This means that the tool runs neither on a client nor on an AP in the network. Another possible way would be to detect the attack on an AP. This option is much easier and less prone to interference. However, this way there is a situation when we cannot verify that the attack was successful or not. Specifically, this happens in case the attacker does not forward the data frames right after the handshake. Besides, our way of solution allows us to monitor all networks on the channels we decided to monitor and also, it does not have any requirements on the operating system or memory of the AP. This decision made a problem more complicated. First, we need a monitoring device for each channel we want to monitor, as shown by the analysis in Chapter 3 and was also experimentally confirmed. Besides, we monitor all the traffic going around, meaning we also see frames that for example will not be accepted by a device driver because of a wrong value of the replay counter or because of being malformed. Also, we have to distinguish different handshakes between the same two devices and different pairs of devices. The channels work completely separate in different processes. It was experimentally examined how many messages we can see in side channels (with wireless network card Alfa AWUS036NHA). It was found, we are not able to see the whole 4-way handshake, in most cases, we have seen only two

messages. That is why we neglect these extra messages we could eventually see twice (on different channels).

In Figure 4.1 on the left, we can see the detection system with three plugged interfaces on channels 1, 6 and 11. This picture serves for the better understanding of the designed solution. On the right, we can see a hypothetical attacker attacking a client connected to Wi-Fi router on channel 1 (only for illustrating of the design).

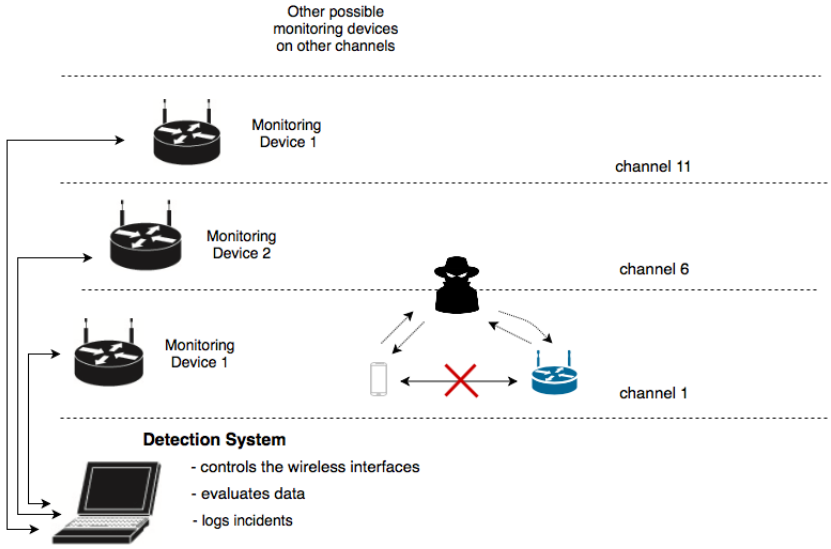

Figure 4.1: Design of the detection system.

There is a process running on each channel. The process must be able to:

- Control the channel interface (set it up, set it to monitor mode, change the channel, etc.).
- Listen on the interface and monitor data.
- Parse the received data and evaluate as an event (i.e., received frame is a *message 1* of the handshake, etc.).
- Evaluate events that happened in sequence according to non-malicious behavior of the 4-way handshake detect the KRACK attacks.
- Log events and save traffic to `.pcap` file for later examination.

These functions correspond to the components we proposed for the testing tool in Section 4.1. Thus, we will not list them again. We can see the relationship of these components in Figure 4.2.

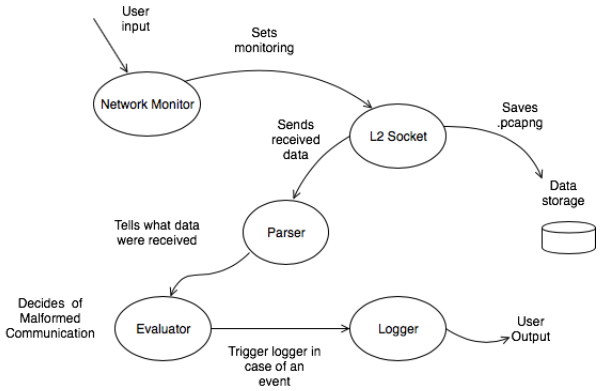

Figure 4.2: Relationship of components in the process.

State Machine for Detection

To evaluate the traffic and detect the KRACK attacks, we created a state machine of the 4-way handshake between a client and an AP. This handshake is identified by MAC addresses of both (the client and the AP) and by unique nonces (ANonce, SNonce) the AP and the client generate before sending *message 1* and *message 2* respectively. In Chapter 3, we found characteristics that we are now going to use for this purpose. In Figure 4.3, there is shown the final automata with only the scenarios that are relevant to detection of the KRACK attacks.

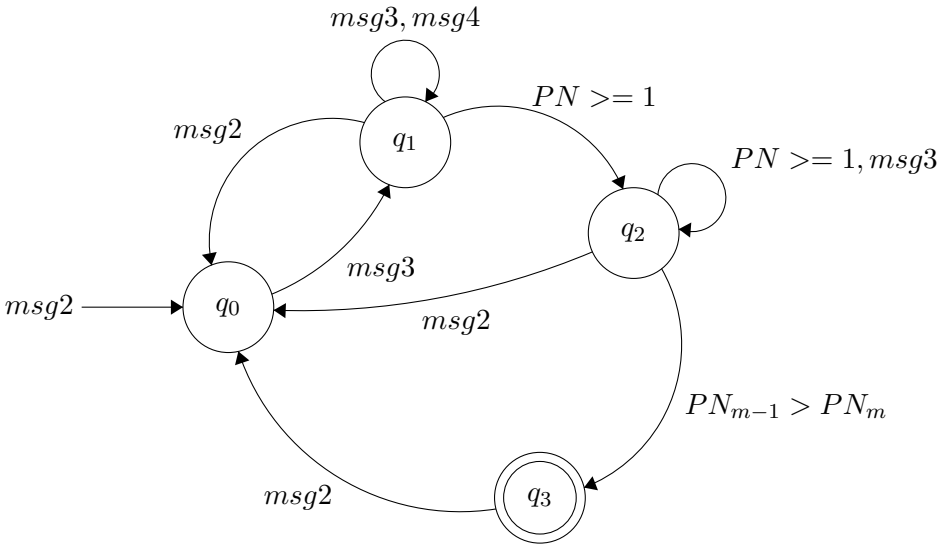

Figure 4.3: State machine of the 4-way handshake with acceptance state of the KRACK attack.

We receive many different types of frames but except the EAPOL frames and encrypted data frames, we ignore them. When we get in a specific state of the machine a different input than we expected (that is not listed next to transitions), we ignore it and stay in the same state. The initial state of the machine is the state q_0. We transition there when we receive *message 2* and we store a value of the SNonce used in this message. When we get *message 3*, we transition to the state q_1. We store the ANonce used in this message and now, we uniquely identified the handshake. We are waiting for encrypted data frames in q_1. When we get them, we are looking at the packet number received. It can and it will be reset after the handshake is finished, but we have to store the information about it to recognize when it happened for the second time. We look at two scenarios with one uniquely identified handshake. First, we detect the packet number is reset twice. Second, we missed the encrypted data frame with the reset packet number but we got another data frame and its packet number is lower than the one we have seen before. These two scenarios are listed in states q_1, q_2, q_3. In the acceptance state q_3, the KRACK attack is detected.

Chapter 5

Implementation

For our implementation, we chose to use programming language Python and package Scapy. It allows us to handle incoming and outcoming frames and also provides helpful tools for parsing the data. We ran it on Kali Linux and five Alfa AWUS036NHA Wi-Fi USB adapters. They have chipset Atheros AR9271, so we used driver ath9k_htc for all of them. For running the attack, we used the same setup with two Alfa AWUS036NHA Wi-Fi USB adapters.

5.1 Structure

The structure of the detection tool consists of six files. We will explain what they are used for:

krackdetect.py This file contains the main input function and class Detector which handles the detection process, meaning creating other classes, handling received messages and writing to possible `.pcap` file. There are two important functions `handle_EAPOL`, which handles incoming EAPOL frames and `handle_encrypted_data`, which handles incoming encrypted data frame.

NetworkMonitor.py The file contains a class NetworkMonitor. The function methods handle the network-manager service, the interfaces and handle turning on and down the monitoring. It provides for example method like `configure_interface_for_monitoring`, which sets the interface up and sets it to monitor mode. For this purpose, we use `ifconfig`, `iwconfig` and `airmon-ng` tools. Other methods are for example, set the channel to interface and turn on or off the network-manager to do not interfere with the monitoring.

Util.py The file contains functions for parsing data from frames as a sequence number, replay counter, packet number and a number of a message of

the 4-way handshake. Also, there is a function which creates an identifier for a pair of a client and an AP based on their MAC addresses.

Logger.py Only handles log messages sent to the terminal. The messages have 6 levels of severity. In case of an attack it is a message of severity "ERROR".

ListenSocket.py Class initiating the L2Listen socket for monitoring, it also implements `recv` which returns 802.11 layer of the captured frame; `close` function closes the socket. We can use either this class or sniff function.

PairState.py The class represents a pair of an AP and a client performing the 4-way handshake. The function `handle_msg` reacts to incoming messages and stores necessary data like nonces.

5.2 Encountered Problems

We encountered several problems during the implementation of the detection tool. The attack itself is quite hard to be implemented. It is necessary to have an extensive background knowledge of the Wi-Fi standard and proper hardware and software. Additionally, we have to find vulnerable devices. Also, some of the handshakes that are vulnerable are very hard to monitor and study because either their messages are always encrypted or only some devices support them. Thus, it is more complicated to trigger them. Besides, different devices behave a bit different, meaning, for example, some of them will not accept the retransmitted *message 3* until they get a retransmitted *message 1* first. And still, every device can monitor only on one channel. It means that for reliable monitoring of other attacking device we need four monitoring Wi-Fi cards, two for the attack itself and two for monitoring the attack from an outside perspective. After all, the detection itself is pretty straightforward.

5.3 Usage

It is possible to use more interfaces and more channels. The first interface will be put to the first channel listed in the command, the second to the second one, etc. When it is not possible to set the listed interface to the monitor mode, it will throw an error, if none of the interfaces can be set to monitor mode, the process exits. To reduce interference between individual interfaces, try to keep them at least a meter in distance to each other.

The system is run as a command-line tool and can be used as follows:

```
usage: krackdetect.py [-h] -i INTERFACE [INTERFACE ...] -ch CHANNEL
                      [CHANNEL ...] [-d DUMP] [-q [Q]]

Detection of Key Reinstallation Attacks (KRACKs)

optional arguments:
  -h, --help            show this help message and exit
  -i INTERFACE [INTERFACE ...], --interface INTERFACE [INTERFACE ...]
                        interfaces for monitoring the network
  -ch CHANNEL [CHANNEL ...], --channel CHANNEL [CHANNEL ...]
                        channel at which the traffic will be monitored
  -d DUMP, --dump DUMP  dumps captured data to .pcap file
  -q [Q]                quiet
```

Chapter 6

Testing and Evaluation

This chapter deals with the testing part of the thesis. First, we tested a set of devices for vulnerability. Second, we tested our developed system for detection of the KRACK attacks against the 4-way handshake.

6.1 Device Vulnerability

We used the testing tool published by Vanhoef and introduced in Subsubsection 4.1.1 to test a set of devices predominantly with iOS and Android operating systems. The test results are listed in Table 6.1. We can see that according to the testing script, all of the devices are vulnerable to the GTK reinstallation during the 4-way or group key handshake. We found four Android devices vulnerable to the PTK reinstallation. One of them still reinstalls the all-zero key and so, allows the adversary to decrypt all data. Besides, we can see none of the iOS devices tested were vulnerable to the PTK reinstallation, even though, we also tested iOS version 9.3.5 which is not supported by the vendor anymore.

We also tested the vulnerability of the FT handshake on two the same AP devices UniFi AP-AC-Pro (testing only one, without another device supporting 802.11r is not possible). And both were not vulnerable to the attack.

6.2 Attack Detection

Due to the nature of the implementation, we have decided to test our detection system by measuring the percentage of correctly detected attacks and the percentage of reported false alarms. For both, we tested the system in environment defined in Section 3.2, environment II.

6.2.1 False Alarms

To figure out if our detection system triggers alerts as false positives, we assumed the tested environment is secure when we are not performing the attack. We set up two Wi-Fi APs on channels 1 and 9 in the environment. To run our script, we used five external Wi-Fi cards and set them to channels 1, 3, 5, 7, and 9 respectively. We let it run for three hours in the defined environment during the day during the standard operation. And we watched detected attacks. We did not detect any false alarms so, we consider our script in this term reliable.

6.2.2 Detected Attacks

We used two Wi-Fi NICs for the attack and three for the detection. We assumed we do not know which channel will the attacker use to clone the real AP to. However, we know which channels are our Wi-Fi APs running on. Thus, we used the channels 1 and 9 and the third we decided by random choice to set to 5. We made a few different scenarios in terms of the relative position of the detection tool, the attacker, attacked client and the real AP. In all these scenarios, we were still inside the flat described in Section 3.2, environment II, thus, we had a decent signal for monitoring of at least one of the three participants in the attack (the AP, the client, the attacker). The detection tool was able to detect all successful attacks in all scenarios. Thus, we can say that in an average environment in terms of interference and device distances, we are able to detect the attack reliably. Also, it is enough to be able to monitor either the side of the AP or the side of the client, in channel based MitM, to be able to detect the KRACK attacks.

6.3 Comparison to Other Tools

During the research, we found only one other tool for dealing with the same problem. It is available in [55]. This tool runs on the AP in a network and detects only retransmission of the *message 3* of the handshake. This retransmission can happen even if there is no attack and the script does not have any chance to verify that the attack was performed or in case it was that it was successful. Optionally, in case it detects this retransmission, it disassociates and deauthenticates the client from the AP. This approach can lead to a lot of false positives and worsen the service provided by the AP. It runs only on APs with Python available.

6.4 Evaluation

There were many mobile devices tested for their vulnerability to the KRACK attacks. Even though, it has been more than a year since the publication of

the vulnerability, some of them are still not patched.

According to testing of our detection system, in case we are in a decent range for monitoring the attacked client or the AP, in a place with no significant interference, we reliably detect the attack.

Device	**OS**	**PTK reinstall**	**GTK reinstall**
Asus Z007	Android 4.4.2	vulnerable	vulnerable
Samsung SM-G530H	Android 4.4.4	vulnerable	vulnerable
Asus Nexus 7	Android 5.1.1	vulnerable	vulnerable
myPhone Pocket	Android 6.0	not vulnerable	vulnerable
LGE Nexus 5	Android 6.0.1	all-zero key	vulnerable
OPPO R9s Plus	Android 6.0.1	not vulnerable	vulnerable
AllView P41 eMagic	Android 7.0	not vulnerable	vulnerable
Huawei VNS-L21	Android 7.0	not vulnerable	vulnerable
Xiaomi Redmi Note 4	Android 7.0	not vulnerable	vulnerable
Huawei Ane-LX1	Android 8.0	not vulnerable	vulnerable
Huawei ATU-L31	Android 8.0	not vulnerable	vulnerable
Huawei MHA.AL00	Android 8.0	not vulnerable	vulnerable
Samsung SM-A320FL	Android 8.0	not vulnerable	vulnerable
Samsung SM-G950F	Android 8.0	not vulnerable	vulnerable
Samsung SM-J330F	Android 8.0	not vulnerable	vulnerable
LGE Nexus 5X	Android 8.1.0	not vulnerable	vulnerable
Xiaomi Mi A2 Lite	Android 8.1.0	not vulnerable	vulnerable
iPad mini	iOS 9.3.5	not vulnerable	vulnerable
iPhone 6	iOS 11.4	not vulnerable	vulnerable
iPhone 6 Plus	iOS 11.4.1	not vulnerable	vulnerable
iPhone 7	iOS 12.1.1	not vulnerable	vulnerable
iPhone X	iOS 12.1.2	not vulnerable	vulnerable

Table 6.1: Results of device vulnerability testing.

Conclusion

This thesis aimed to study the KRACK attacks and analyze its traffic. It also proposes a tool for testing device vulnerability and creates a system for KRACK detection in real-time.

The KRACK attacks exploit a vulnerability found in the standard 802.11i defining WPA2 protocol. It is the only Wi-Fi data security protocol that was considered secure for more than a decade. Most of us use Wi-Fi every day and often do not think about data we send over it. Thus, when Mathy Vanhoef found this vulnerability and published it in October 2017, it became a media stunt. The vast majority of Wi-Fi devices were affected by this vulnerability. Also, it meant that all data we send over Wi-Fi might be possibly decrypted.

In this work, we studied the principle of the KRACK attacks vulnerability. For this purpose, it was necessary to become acquainted with parts of the Wi-Fi standard and with research behind it. Based on what we studied, the thesis describes the process of Wi-Fi client connection, and the principle of the KRACK attacks vulnerability, its practical impact, and countermeasures. We found that the implementation of the attack is quite complicated, but we modified the proof-of-concept scripts and managed to perform the attack against a few vulnerable mobile devices. Thus, we were able to monitor both standard, and malicious traffic of the 4-way handshake and data frames sent after it. We have created a lab consisting of five Wi-Fi NICs that were switched into monitor mode, and tools to capture and transmit Wi-Fi frames. The lab was used for practical experiments described in this thesis. Based on what we studied and analyzed, we made a list of characteristics that make the attack detectable. The results are also contained in this thesis.

We have proposed the tool for testing device vulnerability and used an existing solution for testing more than twenty mobile devices. We found some of them vulnerable and also, we found a device that still reinstalls the all-zero-key. Besides, we proposed a system for detection of the KRACK attack in real-time. The developed scripts detect the attack against the 4-way handshake by detecting retransmission of the third message of the 4-way handshake and

following reinstallation of the session key. In a typical environment with not extremely high interference, the script works reliably and does not trigger any false alarms.

Even though the thesis has met the set goals, the extension of the detection system might be a beneficial future work. Also, the analysis of other handshakes from the standard for vulnerabilities could help improve Wi-Fi security. The topic is extensive, and I found this work a decent basis for future extension.

Bibliography

[1] Cam-Winget, N.; Housley, R.; et al. Security Flaws in 802.11 Data Link Protocols. *Commun. ACM*, volume 46, no. 5, May 2003: pp. 35–39, ISSN 0001-0782, doi:10.1145/769800.769823.

[2] Bittau, A.; Handley, M.; et al. The final nail in WEP's coffin. In *2006 IEEE Symposium on Security and Privacy (S P'06)*, May 2006, ISSN 1081-6011, pp. 15 pp.–400, doi:10.1109/SP.2006.40.

[3] Mekhaznia, T.; Zidani, A. Wi-Fi Security Analysis. *Procedia Computer Science*, volume 73, 2015: pp. 172 – 178, ISSN 1877-0509, doi: https://doi.org/10.1016/j.procs.2015.12.009, international Conference on Advanced Wireless Information and Communication Technologies (AWICT 2015).

[4] IEEE. IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements. *IEEE Std 802.11i-2004*, July 2004: pp. 1–190, doi:10.1109/IEEESTD.2004.94585.

[5] Vanhoef, M. KRACK Attacks: Breaking WPA2. [online], [Accessed 25 May 2018]. Available from: `https://www.krackattacks.com`

[6] Vanhoef, M.; Piessens, F. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*, ACM, 2017.

[7] Vanhoef, M.; Piessens, F. Release the Kraken: new KRACKs in the 802.11 Standard. In *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS)*, ACM, 2018.

[8] Vanhoef, M. Auditing KRACKs in Wi-Fi. [online], 2018, [Accessed 30 October 2018]. Available from: https://www.krackattacks.com/followup.html

[9] Wi-FiNigel. Which Channels Can I Use On The 2.4GHz Band for Wi-Fi Networks? [online], Jan 2017, [Accessed 20 November 2018]. Available from: https://wififorbeginners.com/2016/11/27/which-channels-can-i-use-on-the-2-4ghz-band-for-wi-fi-networks/

[10] Gast, Matthew S. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005, ISBN 0596100523.

[11] Hiertz, G. R.; Denteneer, D.; et al. The IEEE 802.11 universe. *IEEE Communications Magazine*, volume 48, no. 1, January 2010: pp. 62–70, ISSN 0163-6804, doi:10.1109/MCOM.2010.5394032.

[12] IEEE. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition. *IEEE Std 802.11r-2008 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008)*, July 2008: pp. 1–126, doi:10.1109/IEEESTD.2008.4573292.

[13] IEEE. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, Dec 2016: pp. 1–3534, doi:10.1109/IEEESTD.2016.7786995.

[14] IEEE, Inc. Official IEEE 802.11, Working Group Project Timelines - 2018-11-21. [online], 2018, [Accessed 2 December 2018]. Available from: http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm

[15] Wi-Fi Alliance. Technical Note Removal of TKIP from Wi-Fi Devices. [online], Mar 2015, [Accessed 2 November 2018]. Available from: https://www.wi-fi.org/download.php?file=/sites/default/files/private/Wi-Fi_Alliance_Technical_Note_TKIP_v1.0.pdf

[16] IEEE. IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band. *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by*

IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012), Dec 2012: pp. 1–628, doi:10.1109/IEEESTD.2012.6392842.

[17] Wi-Fi Alliance. Wi-Fi Alliance introduces security enhancements. [online], Jan 2018, [Accessed 25 May 2018]. Available from: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-security-enhancements

[18] Vanhoef, M. WPA3: A Missed Opportunity. [online], Jun 2018, [Accessed 12 November 2018]. Available from: https://www.mathyvanhoef.com/2018/06/wpa3-missed-opportunity.html

[19] IEEE. PeerKey Deletion Cleanup. Technical report, IEEE, February 2018, [Accessed 10 December 2018]. Available from: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwiZkvqwoN_fAhVSMewKHbrCDIcQFjAAegQIBBAC&url=https%3A%2F%2Fmentor.ieee.org%2F802.11%2Fdcn%2F18%2F11-18-0480-01-000m-peerkey-deletion-cleanup.docx&usg=AOvVaw0JNdGiAj2bfqyG2yqreWVI

[20] IEEE. IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Fast Initial Link Setup. *IEEE Std 802.11ai-2016 (Amendment to IEEE Std 802.11-2016)*, Dec 2016: pp. 1–164, doi:10.1109/IEEESTD.2016.7792308.

[21] Cao, Z.; He, B.; et al. EAP Extensions for the EAP Re-authentication Protocol (ERP). RFC 6696, RFC Editor, July 2012.

[22] IEEE. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 7: Extensions to Direct-Link Setup (DLS). *IEEE Std 802.11z-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11w-2009, IEEE Std 802.11n-2009, and IEEE Std 802.11p-2010)*, Oct 2010: pp. 1–96, doi:10.1109/IEEESTD.2010.5605400.

[23] ACM CCS. CCS 2017 - Session 3F · ACM CCS Blog. [online], 2017, [Accessed 25 May 2018]. Available from: https://acmccs.github.io/session-F3/

[24] Black Hat. Black Hat Europe 2017. [online], [Accessed 25 May 2018]. Available from: https://www.blackhat.com/eu-17/briefings/

schedule/#key-reinstallation-attacks-breaking-the-wpa2-protocol-8861

[25] Vanhoef, M. KRACK Attacks: Bypassing WPA2 against Android and Linux. [online], [Accessed 25 May 2018]. Available from: https://www.youtube.com/watch?v=Oh4WURZoR98

[26] Zhang, F.; Ma, J.; et al. The Security Proof of a 4-way Handshake Protocol in IEEE 802.11I. In *Proceedings of the 2005 International Conference on Computational Intelligence and Security - Volume Part II*, CIS'05, Berlin, Heidelberg: Springer-Verlag, 2005, ISBN 3-540-30819-9, 978-3-540-30819-5, pp. 488–493, doi:10.1007/11596981_72.

[27] He, C.; Sundararajan, M.; et al. A Modular Correctness Proof of IEEE 802.11I and TLS. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, New York, NY, USA: ACM, 2005, ISBN 1-59593-226-7, pp. 2–15, doi:10.1145/1102120.1102124.

[28] Culpepper, J. C. Merriam-Webster Online: The Language Center The Staff of Merriam-Webster. *Electronic Resources Review*, volume 4, no. 1/2, 2000: p. 9–11, doi:10.1108/err.2000.4.1_2.9.11.

[29] Vanhoef, M.; Piessens, F. Advanced Wi-Fi Attacks Using Commodity Hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3005-3, pp. 256–265, doi:10.1145/2664243.2664260.

[30] Carnegie Mellon University. CERT Coordination Center. [online], Nov 2017, [Accessed 1 November 2018]. Available from: https://www.kb.cert.org/vuls/id/228519/

[31] Kristate. Project krackinfo. [online], Jan 2018, [Accessed 20 November 2018]. Available from: https://github.com/kristate/krackinfo

[32] Langley, A. ImperialViolet. [online], Apr 2014, [Accessed 20 November 2018]. Available from: https://www.imperialviolet.org/2014/02/22/applebug.html

[33] Goodin, D. Android apps still suffer game-over HTTPS defects 7 months later. [online], Apr 2015, [Accessed 20 November 2018]. Available from: https://arstechnica.com/information-technology/2015/04/android-apps-still-suffer-game-over-https-defects-7-months-later/

[34] Onwuzurike, L.; De Cristofaro, E. Danger is My Middle Name: Experimenting with SSL Vulnerabilities in Android Apps. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile*

Networks, WiSec '15, New York, NY, USA: ACM, 2015, ISBN 978-1-4503-3623-9, pp. 15:1–15:6, doi:10.1145/2766498.2766522.

[35] Goodin, D. Majority of Android VPNs can't be trusted to make users more secure. [online], Jan 2017, [Accessed 20 November 2018]. Available from: https://arstechnica.com/information-technology/2017/01/majority-of-android-vpns-cant-be-trusted-to-make-users-more-secure/

[36] Beck, M.; Tews, E. Practical attacks against WEP and WPA. *IACR Cryptology ePrint Archive*, volume 2008, 01 2008: p. 472, doi:10.1145/1514274.1514286.

[37] Fouque, P.-A.; Martinet, G.; et al. On the Security of the CCM Encryption Mode and of a Slight Variant. In *Applied Cryptography and Network Security : 6th International Conference, ACNS 2008*, *Lecture Notes in Computer Science*, volume 5037, edited by S. M. Bellovin; R. Gennaro; A. D. Keromytis; M. Yung, New York, United States, 2008, pp. 411–428, doi:10.1007/978-3-540-68914-0_25, the original publication is available at www.springerlink.com. Available from: https://hal.inria.fr/inria-00556684

[38] Joux, A. Authentication failures in NIST version of GCM. *Universite de Versailles St-Quentin-en-Yvelines*, 01 2006.

[39] Wireshark Wiki Contributors. CaptureSetup/WLAN - The Wireshark Wiki. [online], [Accessed 1 November 2018]. Available from: https://wiki.wireshark.org/CaptureSetup/WLAN

[40] Samek, J. *Fyzická lokalizace útočníků v sítích WiFi.* Master's thesis, České vysoké učení technické v Praze, Fakulta informačních technologií., 2017.

[41] WirelesSHack. Best Kali Linux Compatible USB Adapter / Dongles. [online], Sep 2018, [Accessed 1 October 2018]. Available from: https://www.wirelesshack.org/best-kali-linux-compatible-usb-adapter-dongles.html

[42] Klaviatur; Chetouani, Y.; et al. Buy the Best Wireless Network Adapter for Wi-Fi Hacking in 2018. [online], Nov 2018, [Accessed 12 November 2018]. Available from: https://null-byte.wonderhowto.com/how-to/buy-best-wireless-network-adapter-for-wi-fi-hacking-2018-0178550/

[43] NirSoft. WifiChannelMonitor v1.58. [online], 2014, [Accessed 10 December 2018]. Available from: https://www.nirsoft.net/utils/wifi_channel_monitor.html

[44] TamoSoft. Multi-Channel Capturing. [online], [Accessed 10 December 2018]. Available from: https://www.tamos.com/htmlhelp/commwifi/multi-channel_capturing.htm

[45] Levshova, K. scanner, 3D analyzer and monitor for Windows 10 / Mobile. [online], 2015, [Accessed 10 December 2018]. Available from: http://wificommander.com/

[46] Netspotapp software. Choose the best Wi-Fi channel with NetSpot Scanner. [online], Jan 2019, [Accessed 3 January 2019]. Available from: https://www.netspotapp.com/wifi-channel-scanner.html

[47] Vanhoefm. Project krackattacks-poc-zerokey. [online], Mar 2018, [Accessed 31 October 2018]. Available from: https://github.com/vanhoefm/krackattacks-poc-zerokey

[48] Vanhoef, M.; Piessens, F. Advanced Wi-Fi Attacks Using Commodity Hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3005-3, pp. 256–265, doi:10.1145/2664243.2664260.

[49] ZIMPERIUM. How to Detect KRACK Man in the Middle Attacks — zIPS. [online], Sep 2018, [Accessed 20 December 2018]. Available from: https://blog.zimperium.com/detecting-krack-man-in-the-middle-attacks/

[50] Vanhoefm. Project krackattacks-scripts. [online], 2018, [Accessed 4 July 2018]. Available from: https://github.com/vanhoefm/krackattacks-scripts

[51] Wi-Fi Alliance. Security Update October 2017. [online], Oct 2017, [Accessed 10 October 2018]. Available from: https://www.wi-fi.org/security-update-october-2017

[52] Malinen, J. hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. [online], Jan 2013, [Accessed 1 December 2018]. Available from: https://w1.fi/hostapd/

[53] Malinen, J. Linux WPA/WPA2/IEEE 802.1X Supplicant. [online], Jan 2013, [Accessed 1 December 2018]. Available from: https://w1.fi/wpa_supplicant/

[54] Malinen, J. Hostapd tests. [online], Dec 12AD, [Accessed 10 December 2018]. Available from: https://w1.fi/cgit/hostap/tree/tests/cipher-and-key-mgmt-testing.txt

[55] Securingsam. Project krackdetector. [online], Oct 2017, [Accessed 1 January 2019]. Available from: https://github.com/securingsam/krackdetector

[56] Tektronix, Primer. Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements. *Tektronix*, 11 2013.

[57] Electronics Notes. IEEE 802.11g Wi-Fi Tutorial. [online], [Accessed 20 November 2018]. Available from: https://www.radio-electronics.com/info/wireless/wi-fi/ieee-802-11g.php

[58] Wi-Fi Alliance. Wi-Fi 6. [online], [Accessed 20 November 2018]. Available from: https://www.wi-fi.org/discover-wi-fi/wi-fi-6

Appendix A

Acronyms

AES Advanced Encryption Standard.

AP Access Point.

BIP Broadcast/multicast Integrity Protocol.

BSS Basic service set.

BSSID Basic service set identifier.

CCM CBC-MAC.

CCMP Counter-Mode protocol.

DA Destination address.

DLL data link layer.

DS distribution system.

DSSS direct sequence spread spectrum.

ESS Extended service set.

FCS Frame Check Sequence.

FHSS frequency hopping spread spectrum.

FILS Fast Initial Link Set-up.

FT The Fast BSS Transition.

GCMP Galios/Counter Mode Protocol.

GTK Group Temporal Key.

IEEE Institute of Electrical and Electronics Engineers.

ISM Industrial, Scientific and Medical.

IV initialization vector.

LAN Local Area Network.

LLC logical link control.

LMSC LAN/MAN Standards Committee.

MAC Media access control.

MAN Metropolitan Area Network.

MGWS Multiple Gigabit Wireless System.

MIC Message Integrity Check.

MPDU MAC Protocol Data Unit.

PHY physical layer.

PMK Pairwise Master Key.

PN packet number.

PTK Pairwise Transient Key.

QoS Quality of Service.

RA Receiving STA address.

rRK re-authentication Root Key.

SA Source address.

SSID Service set identifier.

STA station.

STSL station-to-station link.

TA Transmitting STA address.

TDLS Tunneled Direct Link Setup.

TK Temporal key.

TKIP Temporary Key Integrity Protocol.

TPK TDLS PeerKey.

TSC TKIP sequence counter.

WECA Wireless Ethernet Compatibility Alliance.

WEP Wired Equivalent Privacy.

WFA Wi-Fi Alliance.

WG Working Group.

WiGig Wireless Gigabit.

WLAN Wireless Local Area Network.

WNIC wireless network interface controller.

WNM Wireless Network Management.

WPA Wi-Fi Protected Access.

WPA2 Wi-Fi Protected Access 2.

Appendix B

802.11 Significant Family Standards and Amendments

802.11 (1997) The initial standard which provided 1 or 2 Mbps transmission in the 2.4 GHz band using either frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS).

802.11a (1999) It operates at the 5 GHz ISM[3] band with the theoretical data rates up to 54 Mbps. The modulation techique used is OFDM.

802.11b (1999) IEEE 802.11b was the first wireless LAN standard to be widely adopted and built in to many laptop computers and other forms of equipment [56]. It operates at the 2.4 GHz ISM band with theoretical data rates up to 11 Mbps. Although the IEEE 802.11a standard was introduced at the same time and was capable of higher speeds, it did not catch on in the same way. The main reason for this was that it operated in the 5 GHz ISM band rather than the 2.4 GHz of 802.11b, and this made it more expensive [56].

802.11d (2001) International (country-to-country) roaming extensions

802.11e (2005) Quality of Service (QoS) and prioritization

802.11g (2003) Like 802.11b, its predecessor, 802.11g operates in the 2.4 GHz ISM band. It provides a maximum raw data throughput of 54 Mbps. Although the system is compatible with 802.11b, the presence of an 802.11b participant in a network significantly reduces the speed of a net. In fact, it was compatibility issues that took up much of the working time of the IEEE 802.11g committee. The main modulation method chosen for 802.11g was that of OFDM. It soon took over from the b standard and it became the dominant Wi-Fi technology [57].

[3] ISM band - ISM band

802.11h (2003) Power control

802.11i (2004) Document deals with authentication and encryption. Its partial implementation is called WPA and full is called WPA2. It defines the 4-way handshake, the PeerKey handshake and the group Key handshake [4], all vulnerable to the KRACK attacks [5]. These handshakes are further discussed in the section 1.3.

802.11k (2008) Measurement reporting

802.11r (2008) It is also called Fast BSS Transition (or fast roaming). It describes technology to permit continuous connectivity aboard wireless devices in motion. The Fast BSS Transition handshake is also vulnerable to the KRACK attacks and so [5], it will be also explained further in the section 1.3. This standard also slightly extends the 4-way handshake and provides a detailed state machine of the supplicant [12].

802.11n (2009) It uses multiple antennas to increase data rates. As the first Wi-Fi standard that introduced MIMO (Multiple-Input and Multiple-Output) support. The purpose of the standard is to improve network throughput over the two previous standards—802.11a and 802.11g—with a significant increase in the maximum net data rate from 54 Mbit/s to 600 Mbit/s with the use of four spatial streams at a channel width of 40 MHz. It can be used in the 2.4 GHz or 5 GHz frequency bands. The devices supporting this technology are certified as Wi-Fi 4 by the WFA [58].

802.11p (2010) Wireless Access for the Vehicular Environment

802.11s (2011) Mesh networking

802.11u (2011) It adds features that improve interworking with networks like 3G / cellular and other forms of external networks.

802.11v (2011) Deals with WNM and device configuration. This amendment defines a WNM-Sleep mode. In [7], author of the KRACK attack describes a way how to abuse WNM-Sleep response frames to trigger key reinstallation.

802.11w (2009) Protected Management Frames security enhancement

802.11y (2008) Contention Based Protocol for interference avoidance

802.11z (2010) Defines mechanism called TDLS enabling user to directly transfer data between two Wi-Fi clients that are part of the same Wi-Fi network. The TDLS PeerKey handshake is defined for this purpose and is also vulnerable to the KRACK attacks [7] and will be further discussed in the section 1.3.

802.11aa (2012) Enhancements to 802.11 MAC for robust audio streaming while maintaining coexistence with other types of traffic.

802.11ac (2013) Standard providing high-throughput WLANs on the 5 GHz band. The specification has multi-station throughput of at least 1 gigabit per second and single-link throughput of at least 500 megabits per second (500 Mbit/s). This is accomplished by extending the air-interface concepts embraced by 802.11n: wider RF bandwidth (up to 160 MHz), more MIMO spatial streams (up to eight), downlink multi-user MIMO (up to four clients), and high-density modulation (up to 256-QAM). The devices supporting this technology are certified as Wi-Fi 5 by the WFA. This standard also extends GCMP data-confidentiality protocol by adding support for 256-bit keys.

802.11ad (2012) It was developed to provide a Multiple Gigabit Wireless System (MGWS) standard at 60 GHz frequency. Nowadays, it is rolled out under trademark Wireless Gigabit (WiGig). It has limited range (just a few meters and difficult to pass through physical obstacles) compares to other conventional Wi-Fi systems. However, the high frequency allow it to utilize more bandwidth which in turn enable the transmission of data at high data rate up to multiple gigabit per second. This standard also defines a new data-confidentiality protocol for data encryption called GCMP.

802.11af (2013) Wi-Fi in TV spectrum white spaces (often called White-Fi).

802.11ai (2016) The document provides FILS methods to enhance end user experience in dense environments. This function enables a wireless LAN client to achieve a secure link setup within 100ms. The FILS handshake is also vulnerable to the KRACK attack [7] and will be further discussed in the section 1.3.

802.11ax (under development) New standard under development which release is planned on to 2019. The devices supporting this technology will be certified as Wi-Fi 6 by the WFA.

Appendix C

Assigned CVE identifiers

The following Common Vulnerabilities and Exposures (CVE) identifiers were assigned to track which products are affected by specific instantiations of the key reinstallation attack:

- CVE-2017-13077: Reinstallation of the pairwise encryption key (PTK-TK) in the 4-way handshake.
- CVE-2017-13078: Reinstallation of the group key (GTK) in the 4-way handshake.
- CVE-2017-13079: Reinstallation of the integrity group key (IGTK) in the 4-way handshake.
- CVE-2017-13080: Reinstallation of the group key (GTK) in the group key handshake.
- CVE-2017-13081: Reinstallation of the integrity group key (IGTK) in the group key handshake.
- CVE-2017-13082: Accepting a retransmitted Fast BSS Transition (FT) Reassociation Request and reinstalling the pairwise encryption key (PTK-TK) while processing it.
- CVE-2017-13084: Reinstallation of the STK key in the PeerKey handshake.
- CVE-2017-13086: reinstallation of the Tunneled Direct-Link Setup (TDLS) PeerKey (TPK) key in the TDLS handshake.
- CVE-2017-13087: reinstallation of the group key (GTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame.

- CVE-2017-13088: reinstallation of the integrity group key (IGTK) when processing a Wireless Network Management (WNM) Sleep Mode Response frame. Note that each CVE identifier represents a specific instantiation of a key reinstallation attack. This means each CVE ID describes a specific protocol vulnerability, and therefore many vendors are affected by each individual CVE ID.

Appendix D

802.11 Frames Subtypes

Type Value	Subtype Value	Subtype Description
00	0000	Association Request
00	0001	Association Response
00	0010	Reassociation Request
00	0011	Reassociation Response
00	0100	Probe Request
00	0101	Probe Response
00	0110-0111	Reserved
00	1000	Beacon
00	1001	ATIM
00	1010	Disassociation
00	1011	Authentication
00	1100	Deauthentication
00	1101	Action
00	1110-1111	Reserved
01	0000-0111	Reserved
01	1000	Block Ack Request
01	1001	Block Ack
01	1010	PS-Poll
01	1011	RTS
01	1100	CTS
01	1101	ACK
01	1110	CF-end
01	1111	CF-end + CF-ack
10	0000	Data
10	0001	Data + CF-ack
10	0010	Data + CF-poll
10	0011	Data +CF-ack +CF-poll

10	0100	Null
10	0101	CF-ack
10	0110	CF-poll
10	0111	CF-ack +CF-poll
10	1000	QoS data
10	1001	QoS data + CF-ack
10	1010	QoS data + CF-poll
10	1011	QoS data + CF-ack + CF-poll
10	1100	QoS Null
10	1101	Reserved
10	1110	QoS + CF-poll (no data)
10	1111	Qos + CF-ack (no data)
10	0000	Reserved

Table D.1: Subtypes of 802.11 management, control and data frames

Appendix E

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `krackdetect` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
  - `traffic` ..... example of captured traffic
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format