Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Identity-based PSK in WPA2 and WPA3 Networks

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Jonas Panizza
geboren am:
geboren in:

Gutachter/innen: Prof. Dr. Jens Peter Redlich
Prof. Dr. Ernst-Günter Giessmann

eingereicht am: ..... verteidigt am: .....

Abstract

Wireless local area networks have become an essential component of many households today. The widely used wireless encryption standards WPA2 and WPA3 offer comprehensive protection against attacks originating from outside the network. However, entities within a network have numerous possibilities to violate the confidentiality, authenticity and integrity of data transmitted by other network participants. This thesis addresses the problem by proposing an amendment of the above mentioned protocols that enforces the use of identity-based keys. It is shown that many insider attacks are no longer feasible due to the protocol modification, while the impact on performance and usability remains moderate.

Contents

Acronyms

ACL Access-Control List

AES Advanced Encryption Standard

ANONCE Authenticator Number Only Used Once

AP Access Point

ARP Address Resolution Protocol

ASCII American Standard Code for Information Interchange

CBC Cipher Block Chaining

CCMP Counter Mode Cipher Block Chaining Message Authentication Code Protocol

CSA Channel Switch Announcement

DDoS Distributed Denial-of-Service

DNS Domain Name System

DoS Denial-of-Service

EAP Extensible Authentication Protocol

EAPoL Extensible Authentication Protocol over Local Area Network

HMAC Hash-Based Message Authentication Code

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

iPSK identity-based Pre-Shared Key

LAN Local Area Network

MAC Message Authentication Codes

MAC address Media-Access-Control-Address

MITM Man-In-The-Middle

OSI Open Systems Interconnection

PBKDF2 Password-Based Key Derivation Function 2

PFS Perfect Forward Secrecy

PMF Protected Management Frames

PMK Pairwise Master Key

PoC Proof-of-Concept

PSK Pre-Shared Key

PTK Pairwise Transient Key

QoS Quality of Service

RADIUS Remote Authentication Dial-In User Service

RC4 Rivest Cipher 4

RSSI Received Signal Strength Indication

SAE Simultaneous Authentication of Equals

SHA Secure Hash Algorithm

SNONCE Supplicant Number Only Used Once

SSID Service Set Identification

TKIP Temporal Key Integrity Protocol

VLAN Virtual Local Area Network

WEP Wired Equivalent Privacy

WPA Wi-Fi Protected Access

WPA2 Wi-Fi Protected Access 2

WPA3 Wi-Fi Protected Access 3

WPS Wi-Fi Protected Setup

1. Introduction

1.1. Motivation

The Internet of Things (IoT) already has a lasting impact on our life. What a few years ago was still considered dreams of the future has now become a matter of course for many people. The smartphone, for example, has established itself as an all-rounder that can control not only communication but also lighting, shopping, kitchen appliances and even your own front door. More and more people are deciding to optimize everyday household processes with the help of networked devices. According to a recent study, the worldwide installed base of IoT devices will increase from 27 billion in 2017 to 73 billion in 2025 [1]. But in addition to progress and comfort, this new trend also entails risks. Most IoT devices are built with cost and efficiency as the driving factor, but suffer from poor security configurations and open designs [2]. IoT devices are usually equipped with a number of sensors and at the same time offer the possibility of establishing a network connection through which the collected information can be transmitted. The collected information can range from the location of a home to the living habits of a person. While this user data can certainly contribute to the optimization of smart systems, privacy concerns arise.

Given the weak security and valuable user data, IoT devices are also a worthwhile target for hackers. A compromised device can be utilized as gateway to further attack other units in a home network. Effects could range from simple backdoor injection to stealing user information and credentials to even robbery of the victim's property [3]. Furthermore, the discovery of *Mirai* in 2016 showed the extent of malware threats. The malware was used to build a botnet of about 500,000 compromised IoT devices that later became responsible for some of the largest Distributed Denial-of-Service (DDoS) attacks ever seen [4].

In the past, it was often assumed that all clients who had access to a home network were trustworthy. Comprehensive concepts have been developed in recent years to protect home networks from external attacks, but it has often been overlooked that threats can also originate from within the network itself. Large network infrastructures, such as those in big companies or universities, offer more protection from inside because different requirements were placed on the network protocol from the very beginning. As the number of networked devices in private households increases, so does the risk of acquiring a device that exploits its privileged position, either because it has been hacked or because it passes on confidential data carelessly. The following thesis will address this problem and propose a protocol adaptation to strengthen the security of home networks against attacks originating from inside the network.

1.2. Goals

The main goal of this thesis is to elaborate a protocol amendment for the state-of-the-art wireless security protocols Wi-Fi Protected Access 2 (WPA2) and Wi-Fi Protected Access 3 (WPA3). The amendment aims to provide clients in home networks with an encrypted channel to the Access Point (AP) that cannot even be decrypted by network members. Furthermore, it is intended to ensure that clients can be reliably identified, even if they try to change their identity. Proper identification should help to guarantee that an AP can reliably separate clients from each other. In order to deploy the proposed protocol adaptation, compatibility with the existing supplicant implementation must be ensured. Secondary goals considering the amendment are good performance and usability, which should be maintained as well as possible, while achieving the primary goals.

Another objective is to provide the reader of this thesis with a deeper understanding of the underlying problems and to raise awareness of the need to improve existing home network security.

1.3. Structure

The remainder of this thesis follows the ensuing outline. Chapter 2 introduces all major security protocols for wireless networks. The selection will be limited to WPA2 and WPA3, whose operating principles are described below. Furthermore, the vulnerabilities of these protocols and the attacks they allow are explained in detail. Chapter 3 gives an overview of related works and research on this topic. In chapter 4, the proposed protocol adaptations are presented in pseudo code and explained at length. Chapter D briefly outlines the Proof-of-Concept (PoC) implementation details and introduces the hardware and software used to run the extension. In chapter 6, the protocol adaptation will be compared to the genuine implementation in terms of a runtime analysis, memory consumption analysis and binary size comparison. Furthermore, it is discussed how the protocol modification could have been done differently to achieve similar and even better results. Finally, the last chapter will summarize the results. Last but not least, this thesis contains four appendices, which hold the documentation and configuration files of the practical work.

2. Fundamentals and Underlying Problems

In 1997, The Institute of Electrical and Electronics Engineers (IEEE) developed the IEEE 802.11 standard, which describes the guidelines on how to create a network in which clients can communicate wirelessly. The original protocol does not take any measures to ensure confidentiality, authenticity or integrity; it transmits data in pure plain text. The need for secure encryption soon became apparent and in the last 23 years several amendments have been released by the IEEE to satisfy stronger security requirements. At the time of writing, four major security protocols have been developed for the 802.11 standard: Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), WPA2 and WPA3. WEP, developed in 1999, was the first protocol used to secure wireless networks. It is nowadays regarded as insecure given significant flaws that allow one to break its encryption within minutes [5, 6]. WPA was quickly developed in 2003 to provide a secure alternative to WEP. To ensure compatibility with hardware targeting WEP, WPA's underlying Temporal Key Integrity Protocol (TKIP) uses the Rivest Cipher 4 (RC4) for encryption, just as WEP, but with per-packet RC4 keys. It was developed as a temporary solution to replace WEP without requiring the replacement of legacy hardware. Since January 2009, TKIP has been considered insufficiently secure and the IEEE advises against its use [7]. Given that WEP and WPA are no longer secure, WPA2 and WPA3 will be the focus of the rest of this paper.

2.1. Wi-Fi Protected Access 2

In 2004, the Wi-Fi Alliance launched a new version of WPA, called WPA2. It implements large parts of the 802.11i standard. While it is still possible to use the old and insecure TKIP method for legacy hardware, they introduced the Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP). CCMP provides confidentiality by encrypting the data according to the Advanced Encryption Standard (AES) in Counter (CTR) mode. Lastly, for authentication and integrity, a Cipher Block Chaining (CBC)-Message Authentication Codes (MAC) is utilized. The block cipher AES is classified in the year 2020 by the "Bundesamt für Sicherheit in der Informationstechnik" (BSI) as secure with a minimum key length of 128 bits [8].

There are two different authentication methods in all WPA protocols, called *WPA-Personal* and *WPA-Enterprise*. In WPA-Enterprise with 802.1X authentication, the client authenticates (mostly with a unique username and password pair) to an authentication server. The topology is based on three instances, the *supplicant*, the *authenticator* and the *authentication server*: The authentication process is initiated by the supplicant (e.g. the *client* or *station* who wants to connect to the network). They ask the authenticator (e.g. the *AP* or *router*) for network access. The authenticator forwards the credentials to the authentication server (e.g. Remote Authentication Dial-In User Service (RADIUS) server), which will eventually decide whether to grant access. The decision is then sent back to the authenticator. WPA-Enterprise does not require a specific authentication protocol, but uses the Extensible Authentication Protocol (EAP) for encapsulation of other authentication protocols [9]. There exists

various EAP types like the Lightweight Extensible Authentication Protocol (LEAP), EAP Transport Layer Security (EAP-TLS), EAP Tunneled Transport Layer Security (EAP-TTLS) and many more [10].

WPA-Enterprise, as the name suggests, is often used in large network infrastructures, such as for big companies or universities. Since it requires some expertise to set up and maintain the network, there exists a more lightweight authentication method called WPA-Personal. Using this method, the client can authenticate using just a secret key called Pre-Shared Key (PSK). This method is often used in private home networks, small office or home office (SoHo) environments and in small companies. When implemented correctly, both approaches offer reliable security, but include some advantages and disadvantages that will be discussed later.

In order to understand the problems which emerge with WPA2-Personal, it is worth taking a closer look how the protocol works. Since multiple network members use the same PSK, they can't be distinguished by their credentials. Therefore, the *Media-Access-Control-Address (MAC address)* assigned to the network card of each device determines the identity. WPA2 ensures that after every successful authentication new ephemeral session keys are negotiated between the supplicant and the AP. The session keys are negotiated using the 4-way Extensible Authentication Protocol over Local Area Network (EAPoL) handshake. Both the supplicant and AP generate random numbers, the so-called Authenticator Number Only Used Once (ANONCE) and Supplicant Number Only Used Once (SNONCE), and send them to each other in plain text. Subsequently, they both derive the same Pairwise Transient Key (PTK) from the two NONCEs, the MAC addresses of the supplicant and AP for identification and the Pairwise Master Key (PMK) for authentication. In WPA2-Personal, the PSK is used as PMK, whereas in WPA2-Enterprise the PMK is prior sent from the authentication server to the supplicant via EAP. The PTK is finally used to derive several session keys like the unicast and group keys used for the actual encryption with CCMP.

2.2. Weaknesses of Wi-Fi Protected Access 2 Personal

All components needed to derive the PTK, and thus be able to decrypt the actual data, are sent in plain text over the radio, excluding the PMK. Therefore, it is crucial for the security of WPA2 that the PMK is kept secret. In the event of WPA2-Personal, all supplicants share the same PSK from which they can derive the PMK. This brings us to the core problem: Clients on the same WPA2-PSK network can decrypt and encrypt each other's traffic. The only requirement is that they have recorded another client's traffic from the very begin of the session. The information in the recorded 4-way handshake combined with the knowledge of the PSK gives an adversary the opportunity to derive the PTK. The PTK can then be used to gradually decrypt all CCMP encrypted data frames. Given that packet numbers and counters are used in CCMP, it is important that the adversary has captured all packets from the client. In the event of a client already connected to the network, an adversary can exploit the fact that managements frames aren't verified or replay-protected by default. Hence, the attacker could make the target leave the network by sending forged deauthentication

packets. Supplicants are programmed to regain connection to the network right after they have been deauthenticated. This allows an adversary to record the new session from the beginning.

With the ability of decrypting other client's traffic at hand, there exists a large variety of attack vectors an adversary could perform. First things first, all clear text network protocols could be exploited without further effort. E-mails send and accessed over the Simple Mail Transfer Protocol (SMTP), the Internet Message Access Protocol (IMAP) or the Post Office Protocol 3 (POP3); data transmitted over the File Transfer Protocol (FTP), Telnet or SQLnet; or credentials sent with the Hypertext Transfer Protocol (HTTP) are no longer confidential. The data can easily be collected and decrypted with programs like `wireshark`, `tcpdump` or `airodump-ng`.

Attacks using more sophisticated vectors are also within the bounds of possibility: One could try to gain a Man-In-The-Middle (MITM) position with the use of *Address Resolution Protocol (ARP) spoofing*. ARP spoofing (or poisoning) is the process of actively sending out unsolicited ARP packets on the network to alter the ARP table of two victims (in most cases another client and the AP). By altering the ARP table, the adversary establishes a connection to both victim machines and relays the messages between them. Both victims believe they are communicating directly to each other, when in reality the communication flows through the host performing the attack. Some home routers have implemented proprietary solutions to detect and avoid ARP spoofing but according to the *Hole 196* vulnerability, forged ARP responses can be sent directly to the client and therefore circumvent the AP [11]. Ample research has been conducted towards developing secure address resolution protocols, like TARP [12] or signed ARP [13], but none have become a standard.

Once the MITM position is established through ARP spoofing, there are various exploits an adversary could perform. They could use a *SSL-stripping* attack on websites without HTTP Strict Transport Security (HSTS) protection to downgrade HTTPS connections to HTTP. They could conduct *Domain Name System (DNS) spoofing* to redirect an ignorant client to a fishing website and with a little luck harvest the client's credentials. They could drop all packages instead of relaying them to the original destination, which will result in a *Denial-of-Service (DoS)* attack. Or they could modify messages and inject additional content, hence violate the integrity of the data. The list of possible scenarios and the damage they may create is huge.

In addition to ARP spoofing, one can acquire the MITM position using the *Evil Twin Attack*. Here, instead of connecting to the existing AP and trying to redirect the packet flow, an adversary mimics the AP and persuades a client to connect to the new "fake" network. The adversary creates their own AP using a Service Set Identification (SSID) and PSK identical to the original AP. Existing built-in Wi-Fi supplicant implementations assumes that all the APs with the same SSID are legitimate and automatically connects to the AP with the maximum Received Signal Strength Indication (RSSI) value [14]. If an adversary accomplishes to present an AP with a higher RSSI than the legitimate AP, clients will likely connect to the "evil twin". As a result, all packets are sent to the adversary, who in turn would forward them to avoid suspicion, thus allowing for all the previously mentioned post-MITM exploits.

There is another important problem associated to WPA2-Personal: The ability of *impersonation*. As already mentioned before, all clients authenticate themselves with the same PSK. Therefore, the AP distinguishes all network members by their unique MAC address associated to the wireless network card. Unfortunately, MAC addresses can easily be forged with the appropriate software, which means that the used identification mechanism is not reliable. *Virtual Local Area Networks (VLANs)* allow network administrators to automatically limit access to a specified group of users by dividing clients into different isolated Local Area Network (LAN) segments. While this concept works well with physical network ports or 802.1X authentication, it is possible to circumvent VLAN restrictions in WPA2-Personal networks with the help of impersonation. *Access-Control Lists (ACLs)* perform a very similar task by restricting or allowing access to specific ports, IPs or protocols, but in contrast to VLANs, they operate on Open Systems Interconnection (OSI) model layer 3. However, they can also be bypassed with impersonation. *Quality of Service (QoS)* is a network feature that allows you to give priority to certain types of Internet traffic or restrict bandwidth of clients. Like VLANs and ACLs, QoS is implemented by using a user's MAC address to determine the identity.

All an adversary has to do to circumvent ACL, VLAN, QoS or tunnel policies is to discover the MAC address of a client whose identity they want to impersonate. They then wait until the target disconnects from the network, disconnect themselves and adopt the sniffed MAC address. As soon as the adversary reconnects with the spoofed MAC address, the AP will not be able to distinguish them from the genuine client. For this reason, they are assigned to the same VLAN and packets are routed according to the same ACL as the impersonated target. This method could gain an intruder access to network recourses such as file servers, databases, printers or the like. It should be noted that this type of impersonation only works when the targeted client is not connected to the network. As soon as two clients with identical MAC addresses connect to the AP simultaneously, complications arise. The router will attempt an EAPoL handshake for both clients, which will generate two different PTKs. These PTKs are then tied to an identity (MAC address) and stored on the AP. Since each identity only allows one PTK, runtime errors occur. The outcome is highly implementation specific, but in most cases the router disconnects both.

On the contrary, there is an impersonation attack which does not postulate a disconnected target. Instead of performing a proper EAPoL handshake themselves, the adversary records the target's EAPoL handshake and keeps track of the respective CCMP packet numbers and counters, much like as they would decrypt data as described previously. With this information, they are able to compose and encrypt messages just as the target does. Since the adversary is using the existing session of the target, which means they are using the existing ephemeral PTK the target negotiated with the AP, this exploit is called *session hijacking*. However, it is very important that once the session is hijacked, the legitimate client does not continue using it, otherwise the state of CCMP will be scrambled. Either a Channel Switch Announcement (CSA) beacon could prompt the client to switch channels to avoid interfering with the adversary or a forged management frame can be sent to deauthenticate the client from the network.

All mentioned exploits assume that the adversary has knowledge over the secret PSK. There are still more attack vectors in which the adversary does not need any prior knowledge, but they will not be covered in this thesis. The weaknesses with WPA2-Personal lead to the introduction of a successor, 14 years later.

2.3. Wi-Fi Protected Access 3

WPA3, announced in January 2018 by the Wi-Fi Alliance, brought several innovations to WPA networks, the most notable of which being Simultaneous Authentication of Equals (SAE). SAE was originally designed for use between peers in IEEE 802.11s mesh networks and is based on the *dragonfly key exchange*. Dragonfly is a patent-free technology that uses a zero-knowledge proof key exchange, which allows users or devices to prove knowledge of a password without having to reveal a password [10]. In WPA3-Personal, SAE replaces the old PSK authentication method and eliminates some of the shown weaknesses. The main goal of SAE is to prevent offline dictionary attacks. Given we're assuming that the adversary already possesses the PSK, we won't go into the specifics of such dictionary attacks. Besides this property, SAE promises *Perfect Forward Secrecy (PFS)* [15]. PFS is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if the private key is compromised [16]. That is, an adversary with the knowledge of the PSK cannot derive the PMK of other clients. The PSK is merely used for authentication, not for deriving the PMK [17].

Another important innovation in WPA3 it the obligatory use of *Protected Management Frames (PMF)*. The idea behind PMF has been around for a long time; it was already specified in 2009 in the 802.11w standard [18]. PMF provides protection for unicast and multicast management action frames like disassociation and deauthentication requests, fast BSS transition or CSAs. Unicast management action frames are protected from both eavesdropping and forging, and multicast management action frames are protected from forging. It is important to mention that PMF must be supported by the wireless network card's chipset and is thus incompatible with many supplicant devices at the time of writing.

Aside from SAE and PMF, WPA3-Personal enforces a minimum key-length of 192 bits, it prohibits unauthenticated key reinstallation and therefore fixes the KRACK vulnerability and introduces *Opportunistic Wireless Encryption* (OWE) which in turn enhances the security in open networks [19].

2.4. Weaknesses of Wi-Fi Protected Access 3 Personal

The advantages of the new authentication method SAE are apparent: an inside adversary with knowledge of the PSK cannot passively eavesdrop on other clients traffic. SAE makes it impossible to derive another client's PMK (thus the PTK) in order to decrypt unicast traffic. However, it is still possible to conduct ARP spoofing and redirect the victim's traffic to a malicious host performing the attack. This way, the packets will be encrypted with the adversaries PTK and therefore be transparent to

them. The same applies to the Evil Twin attack: An adversary with the knowledge of the PSK can setup a fake WPA3-Personal network with a stronger RSSI than the genuine AP and wait for clients to connect. But there is one downside: A client already connected to the genuine AP cannot be deauthenticated by network members other than the AP. PMF causes unauthenticated deauthentication or disassociation packets to be dropped. It would be possible to jam all radio traffic with a dedicated network jammer until the AP times out the session. However, since this procedure is very conspicuous, it is usually more reasonable to wait until a client reconnects themselves. They will then most likely reconnect to the "malicious" host with the stronger signal.

Impersonation, in terms of session hijacking, no longer works. Since it is not possible to derive the PMK out of the PSK, the active session cannot be pursued by the attacker, because the essential encryption keys are missing. Incidentally, due again to PMF, it isn't possible to expel the genuine client from their session. That said, impersonation attacks mounted by creating a proper session are feasible. The adversary waits until the targeted client leaves the network, they impersonate the client's MAC address and connect themselves, just like in WPA2-Personal. VLANs and firewalls can be circumvented in this manner.

2.5. Conclusion

WPA3 was introduced to replace WPA2 in the long term. Nevertheless, it will take some time for most home networks to be converted. The Wi-Fi Alliance started the certification of WPA3-capable devices in late 2018 [10], but the prevalence is still rather small at the time of writing. In addition, most supplicant-devices do not yet support PMF and are therefore not capable of WPA3-only. We have seen that WPA3 solves some of the vulnerabilities described above, but an attacker within the network still has the ability to violate confidentiality, integrity and authenticity. This problem follows inherently from the fact that every network participant in WPA2-Personal networks has the same PSK. One obvious solution would be to use WPA-Enterprise instead of WPA-Personal in home networks. Since each client negotiates their unique PMK with the RADIUS server, the PTK can't be derived by another client. An eavesdropping adversary within the network has no prior knowledge they can use to attack other network members. They are left with OSI model layer 2 encrypted 802.11 frames, thus all mentioned exploits except ARP spoofing are futile. But in reality there are many reasons that prevent people from using WPA-Enterprise. The main reason being that many devices do not support 802.1X. Especially consumer gear, such as game consoles, entertainment devices, some printers or IoT devices are often not capable to connect to a WPA-Enterprise network. But there are other reasons: It is already pointed out that an authentication server is required to set up such a network. Frankly, most routers for home networks can be configured to have a built in RADIUS server, but for the average consumer it is quite difficult to implement. Once this hurdle is cleared, there is still the question of how certificates and credentials are shared and installed on clients. Distributing generated credentials and certificates and registering them to each device is usually called device *onboarding*. Especially in home networks, people are used to

insert a password, scan a QR-code or push the WPS-button to connect to the network. Granting network access to someone by creating a new user on the authentication server, subsequently installing the server certificates at the user's device and eventually entering these credentials is considered to be too complicated by a majority of the people.

Ideally, we would like to combine the advantages of WPA-Enterprise with those of WPA-Personal. The advantage of WPA-Enterprise, that every network member has their unique credential combined with the simplicity of WPA-Personal. In the following, I will present and discuss a modification of the WPA2/WPA3-Personal protocol that aims at exactly these properties.

3. Current State of Research and Related Works

The idea behind each client having a unique PSK in WPA-Personal networks is not new. Nevertheless, it never made it into a standard. For this reason, several vendors have implemented this feature under different names. They all differ in detail, but the basic idea and functionality is the same.

First of all, almost all routers today offer the possibility to create a guest network. This network has a distinct SSID and it is also advised to define a passphrase distinct from that of the regular network. A VLAN is assigned to both networks and firewall rules prohibit clients from the guest network from accessing the regular network. In most cases, clients from the guest network are completely separated from each other, which is called *client isolation*. This segregation is reliable, since both networks have a different PSK. A potential intruder in possession of the guest passphrase cannot impersonate a client of the regular network because they are in possession of the wrong PSK. ARP spoofing is impossible since the AP isolates the client and an Evil Twin attack could only fool other guest members. In summary, it can be said that the AP can reliably distinguish clients from the guest-and regular network. Not by MAC address, but by the PSK they authenticate themselves with. However, a binary subdivision in more complex network topologies is not sufficient.

The telecommunications company Cisco Systems developed a proprietary extension of the WPA2-PSK protocol called *identity Pre-Shared-Key* (iPSK). iPSK allows unique PSK per endpoint or based on a policy. For instance, groups of like endpoints can share a same PSK value or each of the endpoint can have a unique PSK [20]. Clients can register their device utilizing an external portal and SQL endpoint database for iPSK management, called "iPSK Manager". The assigned PSK is then manually entered at the client. Identities are managed and stored on an Identity Services Engine (ISE), which answers authentication requests forwarded from the authenticator (AP) like a RADIUS server. When a client authenticates to the wireless network, the AP checks with the RADIUS server to see if the MAC address exists. If it does, the RADIUS server will respond with the stored PSK. Subsequently, the AP verifies the stored PSK in the RADIUS policy to the one the client used to authenticate. This system offers the advantages of 802.1X authentication while clients connect using the simple

WPA2-PSK method.

The company LANCOM developed a similar WPA2-PSK amendment. *LANCOM Enhanced Passphrase Security User (LEPS-U)* introduces several PSKs tied to a single network. Depending on the PSK, clients are automatically assigned to a specific VLAN. Since LEPS-U is configured exclusively on the infrastructure side, full compatibility with all types of supplicants is ensured at all times. Within *LANCOM Enhanced Passphrase Security MAC (LEPS-MAC)*, each passphrase is assigned a MAC address in an additional column of the ACL. Only the combination of passphrase and MAC address allows the authentication at the AP. Attacks on the WLAN are thus made considerably more difficult, since by linking the MAC address and passphrase, both parts must always be known in order to negotiate encryption. Both LEPS-U and LEPS-MAC can be used locally in the device as well as centrally managed with the help of a RADIUS server [21].

There are many more proprietary personal PSK implementations to mention like the *Private PSK* feature from Aerohive [22], the *Dynamic PSK* from Ruckus Wireless [23], *Multiple PSK* from Aruba Networks [24] or the *ePSK* from Cambium Networks [25], but since they all differ only marginally in their functionality, I will not go further into detail. In contrast to all the closed-source solutions I mentioned, there is a prominent open-source representative who also implemented this feature: `hostapd`.

`hostapd` is a user space daemon for AP and authentication servers. It implements 802.11 AP management, 802.1X/WPA/WPA2/EAP authenticators, RADIUS client, EAP server, and RADIUS authentication server [26]. The daemon uses netlink sockets for inter-process communication with the kernel. The current version supports various versions of Linux and FreeBSD. Since the codebase may be distributed, used, and modified under the terms of BSD license [26], most wireless distribution system vendors use large portions of `hostapd's` code for their own products or use it as reference.

In December 2004, `hostapd v.0.3.0` added support for multiple WPA PSKs [27]. They can either be read from a separate text file containing a list of PSK and MAC address pairs or they can be received from the RADIUS authentication server. Each client can be assigned its own unique PSK or groups of clients can be assigned a common PSK. The text file must be created manually before operation and its path must be defined in the `hostapd.conf`. Instead of manually entering each client's MAC address with a unique PSK in the text file, `hostapd` offers the possibility to randomly generate a per-device PSK and send it to the client via *WPS*. The generated PSK will also be appended to the text file for future authentications.

4. Methodology

The primary goal of the adaptations proposed in this thesis is to provide better security in home networks towards adversaries from within the network; more specifically, adversaries in possession of the PSK. As we have seen in section "Current State of Research and Related Works", effort has been made to deprive an adversary of their advantage by giving each client their own identity-based Pre-Shared Key (iPSK). This is done either by using a RADIUS server (similar to 802.1X) to manage and verify the iPSKs or by the AP itself keeping a database or table where all MAC addresses and their associated iPSKs are stored. In the following I will propose a protocol adaptation that allows a wireless distribution system to enforce an iPSK policy for WPA2/WPA3-Personal without RADIUS server and without having to store each iPSK and MAC address pair in a database or table. It is a stateless protocol adaptation that generates the iPSK at runtime when it is needed.

Before I go into more detail, I would like to explain the difference between the *PSK* and *WPA-passphrase*, as this is often misunderstood: When we grant a device access to our WLAN, we typically enter a human-readable password of at least 8 and at most 63 American Standard Code for Information Interchange (ASCII)-characters, called WPA-passphrase. The passphrase is converted internally by the supplicant into a 256-bit key, the so-called PSK. The conversion is carried out by applying the *Password-Based Key Derivation Function 2 (PBKDF2)* to the WPA-passphrase, using the SSID as the salt and 4096 iterations of *HMAC-SHA1* [28]. This kind of transformation is called key stretching and has the purpose of adding computational cost in order to increases the time it takes an adversary to brute-force the WPA-passphrase. The PSK itself is not human readable because not every byte is assigned a printable ASCII-symbol. To onboard a new device, the WPA-passphrase is either entered manually or transmitted via WPS. The supplicant itself then derives the PSK from the WPA-passphrase by the means of key stretching. In the interest of completeness, it should be mentioned that when using fully configurable supplicants, it is possible to enter the PSK in hexadezimal representation manually into the `wpa_supplicant.conf` file, although this is rarely used.

With this distinction in mind, it is noticeable that the CISCO extension *"identity PSK"* should actually be called *"identity WPA-passphrase"* since no iPSKs but indentity-based WPA-passphrases are initially generated. The supplicant themselves implicitly generates their iPSK by receiving an identity-based WPA-passphrase and stretching it as described above. The same applies to all the other WPA extensions presented before. But for the interest of consistency, I will adapt to this slightly confusing naming convention.

4.1. Key Derivation Function

The core of this adaptation is a key derivation function that generates an identity-based WPA-passphrase at runtime. It is a mapping function that assigns a unique key to each individual client, identified by their MAC addresses. There are a number of characteristics that this key derivation function must fulfill: First of all, the function

must be deterministic in order to derive the identical identity-based WPA-passphrase for each new connection of the same client. In addition, only the AP must be able to derive an identity-based WPA-passphrase. According to Kerckhoffs' principle the key derivation function should not be kept secret, thus a master-secret known only to the AP must be included. In the event that an iPSK or identity-based WPA-passphrase is compromised, it must be possible to revoke it. The derived identity-based WPA-passphrase itself should follow the same constraints as a conventional WPA-passphrase to ensure compatibility with the supplicants. It must consist of ASCII-printable characters with a length between 8-63 digits. And last but not least, the function must not affect the runtime of the authentication process too much. One possible key derivation function which fulfills all the characteristics is the following:

```
# Compute HMAC with
HMAC = HMAC_SHA512(key=master_secret, data=MAC_address)
# Use PBKFF2 to stretch the HMAC
stretched_HMAC = PBKDF2(data=HMAC, salt=SSID, iterations=4096, output_bits=384)
# Base64 encode the stretched HMAC
encoded_stretched_HMAC = BASE64(stretched_HMAC)
# Cut the last character to fulfill the WPA-passphrase length
# constraint
identity_based_WPA_passphrase = prefix(encoded_stretched_HMAC, 63)
```

Hash-Based Message Authentication Code (HMAC) is a mechanism for calculating a MAC involving a hash function in combination with a secret key. In this case, the *Secure Hash Algorithm (SHA) 512* is used as hash function. Unlike the conventional use of HMACs, it is not of interest to authenticate a message in this scenario, but to take advantage of the one-way property of SHA. A one-way function means that it is practically infeasible to invert the output, hence the only possibility to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match. Since SHA is a cryptographic hash function, it also offers collision resistance, thus it is very unlikely to find two inputs that hash to the same output.

In this key derivation function, two input parameters are needed to create the HMAC: The MAC address of a supplicant is used as data and a master-secret known only to the AP is used as key. The generated HMAC is then used as input for PBKDF2 with the SSID of the network as salt. To ensure that the stretched HMAC is human-readable, it is then *Base64* encoded. Since Base64 is a six-bit encoding, every four characters of Base64-encoded text represent three bytes of the unencoded stretched HMAC. Therefore the 384 bit issued by the PBKDF2 are mapped to 512 bit, which corresponds to a 64 character ASCII-string. The maximum length of a WPA-passphrase is 63 characters, the last character of the encoded and stretched HMAC is therefore omitted. It would not be necessary to shorten the output if the PBKDF2 outputs 378 instead of 384 bits, but this is not possible because it can only output multiples of whole bytes.

There are two reasons why key stretching is applied in this function. First of all, it makes it more difficult to brute-force the master-secret. Without key stretching, an adversary with knowledge of their own identity-based WPA-passphrase could randomly compute passphrases by trying all possible combinations of the master-secret and comparing the results with their own identity-based WPA-passphrase. When the generated passphrase matches the genuine identity-based WPA-passphrase, the master-secret is successfully found. Since both the calculation of an HMAC and the Base64 encoding are not computationally expensive operations, even long user-defined master-secrets could be cracked with the help of dictionary attacks. In addition, the calculation can be massively parallelized, which would accelerate the process incredibly. Just like deriving the PSK from the WPA-passphrase, key stretching adds additional computational costs for each password attempt to complicate the process. The second reason concerns *rainbow tables*. Once a comprehensive list of tuples with all possible master-secrets and their resulting WPA-passphrases is generated, this list could be used on other networks too. This assumes that the attacker uses the same MAC address that was used to create the rainbow table and that they are in possession of their own identity-based WPA-passphrase. Since PBKDF2 uses the SSID as salt, an exhaustive calculation of all possible master-secrets serves only the purpose of breaking one single network.

Both the HMAC calculation and the Base64 encoding are deterministic, therefore the entire key derivation function is also deterministic. Provided that the master-secret is kept secret, no one else can derive an identity-based passphrase but the AP. In the event that an identity-based WPA-passphrase or iPSK is compromised, the AP can change its master-secret and force new WPA-passphrase to be derived.

4.2. Integration into WPA2 and WPA3

The question remains how the key derivation function is integrated into the WPA2-PSK and WPA3-SAE protocol. In both cases the initial step is identical. Once a client connects to the network, the AP uses their master-secret, the client's MAC address and the SSID as salt to generate the identity-based WPA-passphrase for the particular client with the above defined key derivation function. From this point on, both protocols use the created passphrase differently. In the case of WPA2-PSK, the passphrase is converted into an iPSK with the help of PBKDF2 and used as PMK within the 4-way EAPoL handshake, whereas WPA3-SAE uses the identity-based passphrase instead of the normal WPA-passphrase for the dragonfly handshake of SAE. The beauty of this protocol amendment is that neither SAE nor the EAPoL handshake are modified. Merely the keys that serve as input for both protocols are exchanged by identity-based keys. Neither iPSKs nor identity-based WPA-passphrase are stored on the infrastructure side at any time.

4.3. Key Distribution

The problem of bootstrapping secure communication between wireless devices has always been a challenge. As opposed to conventional WPA networks, no single key has to be distributed to all clients, rather, each client gets their own key. As the keys are dynamically derived, no static passphrase can be printed on the back of a router for future references. An interaction with a device that can derive identity-based WPA-passphrases is therefore essential. One possibility is to extend the router's web interface in such a way that the network administrator can request the identity-based WPA-passphrase of a new supplicant. To accomplish this, the administrator logs on to the router with their credentials and enters the MAC address of the new, not yet connected device. The router derives and displays the identity-based WPA-passphrase, which is finally entered manually at the supplicant.

Most devices can be connected using this method, but there are also (mostly IoT) devices that do not have a screen or keyboard. Various solutions have been developed to address this problem. One common approach is to have the new device appear temporarily as an AP with its own SSID and passphrase, which is supplied on the manufacturer's manual. To onboard it, a smartphone is required to connect to the temporary network and enter the SSID and WPA-passphrase of the home network via an additional application or HTTP landing page. Once the credentials are transmitted, the new device switches from AP to client mode using the received credentials. Another approach is to use an additional application on a smart phone to transfer the credentials via bluetooth, Wi-Fi direct, flashes or event sound waves to the new device [29, 30]. However, it is unimportant how credentials are transferred to the client in detail, as only the infrastructure side is modified in identity-based networks. The only difference is that the identity-based WPA-passphrase obtained from the router's web interface is used instead of the conventional WPA-passphrase for the various onboarding techniques.

Besides the manual readout of the identity-based WPA-passphrase from the router's web interface, there exists another much more convenient key distribution technique: *Wi-Fi Protected Setup (WPS)*. WPS is a standard of the Wi-Fi Alliance that simplifies the process of logging end devices into an encrypted WLAN. Thanks to WPS, the WPA-passphrase does not need to be entered manually, but is transmitted as a series of EAP request/response messages, ending with the AP disassociating from the client and waiting for the client to reconnect with its newly obtained credentials. There are two mandatory WPS methods: the PIN method and the push button method. With push-button configuration, a physical or a software-implemented button exists at both the AP and the client. If both buttons are pressed or triggered within two minutes, the client can automatically join the network, as the WPA-passphrase is transmitted during the configuration phase. When using the PIN method, the AP generates an eight-digit PIN that must be entered on the client side. The credentials are then transferred as for the push button method. In order to transfer identity-based keys instead of shared keys, the WPS *registrar* must be adapted. A modified WPS registrar offers an elegant alternative, as no additional effort is required by an administrator.

5. Implementation

This chapter describes the implementation of the previously described iPSK protocol amendment. To recreate a home network topology, the *AVM FRITZ!Box 7530* has been chosen to serve as an AP. Since the router's pre-installed *FRITZ!OS* firmware is closed-source and therefore not customizable, an open-source Linux operating system for embedded devices, *OpenWrt*, was installed instead. The documentation of the flashing process can be found in appendix A. Instead of providing a single, static firmware, OpenWrt offers a fully writable filesystem with package management. All components are optimized to be small enough to fit into the limited storage and memory available in most home routers. While OpenWrt can be managed completely using SSH and the terminal, the OpenWrt full-releases ship with an extensible and easily maintainable web user interface called *LuCI*. The newest version of OpenWrt (OpenWrt 19.07) has been installed at the time of writing. The FRITZ!Box 7530 offers 128 MB of flash, which would be more than enough storage space for this PoC. However, the memory has been expanded by an external USB drive using *Extroot* to allow extra packages to be installed, making it more user-friendly for the development process. The corresponding documentation can be found in appendix B.

Like most wireless distribution systems, OpenWrt uses an adapted version of `hostapd` to handle all aspects of the wireless infrastructure like authenticating clients, set encryption keys, establishing key rotation policy etc. The difference between the original and OpenWrt's version of `hostapd` consists of some extensions added to the source code. For instance, OpenWrt's inter-process communication system `ubus` and support for the *Unified Configuration Interface* (*UCI*) are integrated.

5.1. hostapd

In order to implement the presented identity-based key policy, several changes have been made to the `hostapd` source code. The requirement that the modified version should also be portable to other systems led to the decision to use the original and not OpenWrt's version of `hostapd` as the basis of the implementation. However, this decision has the disadvantage that `hostapd` cannot be configured using the UCI. Since the web user interface LuCI depends on the UCI, `hostapd` is only controlled via the terminal in this PoC. Another drawback is that the WPS button handling managed by `procd` can no longer be accomplished using `ubus` calls to `hostapd`. For the latter, a workaround is given to preserve the functionality of the WPS button. Instead of calling `ubus` to trigger the WPS push button event, the command-line interface `hostapd_cli` is used to notify the daemon.

In this PoC, an entry has been added to the `hostapd.conf` that enables and disables iPSK mode globally for all networks created by the single `hostapd` instance. Conceptually it is also possible to toggle this mode for each SSID. Once the statement "`ipsk=1`" is included to the configuration file, `hostapd` no longer uses the traditional WPA-passphrase and the corresponding PSK for authentication of WPA2/WPA3-Personal networks. In iPSK mode, the `wpa_passphrase` value defined in the configuration file is considered

to be the master-secret for the key derivation function. Once a client associates to the network, a callback function derives the identity-based WPA-passphrase with the client's MAC address, the SSID and the master-secret. The passphrase is converted to an iPSK and cached into a data structure. The subsequent authentication uses the cached value. If the client re-associates for any reason, the cached iPSK is used instead of re-deriving it. As soon as another client associates, the cached iPSK is overwritten with the one of the new client. A larger cache may be implemented at this point.

Besides adding identity-based authentication to WPA2/WPA3-Personal, the WPS implementation has been adapted to transfer identity-based credentials. It must be noted, that the latest version of `hostapd` (version 2.9) does not support WPS onboarding with SAE-only key management. However, SAE can be enabled automatically for WPA2-PSK credentials received using WPS. When both SAE and WPA2-PSK key management is offered simultaneously at the infrastructure side, this is called *WPA3-Personal transition mode* or *WPA2/WPA3-Personal mixed mode*. The WPS implementation was modified to send the identity-based WPA-passphrase to the supplicant when using WPA2-PSK or WPA3-Personal transition mode. When the statement "`wps_cred_add_sae=1`" is added to the `supplicant.conf`, the received WPA2-PSK credentials are used for WPA3-SAE. If the supplicant supports PMF, it is automatically activated. Finally, the AP can notify authenticated clients to disable transition mode in their network profiles when WPS is completed. Once a *transition disable indication* is sent, the stations are expected to automatically disable WPA2-PSK and less secure security options. Although this procedure is not equivalent to SAE-only WPS onboarding, it yields the same result. Once the original version of `hostapd` supports SAE-only onboarding with WPS, the source code could be adapted in the same way to transfer identity-based WPA3-SAE credentials. An example configuration file to create an identity-based WPA2/WPA3-Personal mixed mode network with WPS enabled can be found in appendix D.

In order to run the modified version of `hostapd` on the FRITZ!Box 7530, the code must be compiled for the *Qualcomm Atheros IPQ40XX* processor architecture. It turned out that on-device compilation was not feasible due to limited resources and library dependencies. Fortunately OpenWrt offers a well documented build system. The build system is a set of makefiles and patches that automates the process of building a cross-compilation toolchain and then using it to build the OpenWrt firmware itself or other pieces of software. A flag has been added to the `hostapd's` compile time config file to toggle between compilation for the native system and cross-compilation for the FRITZ!Box 7530. The documentation how the build system was set up and the toolchain was configured can found in appendix C.

5.2. wpa_passphrase

`wpa_passphrase` is a command line tool which generates a minimal configuration file for `wpa_supplicant`. The sole purpose of the program is to precalculate the PSK on the basis of a specified SSID and WPA-passphrase and output it according to the `wpa_supplicant.conf` syntax. The program is called with the following parameters:

```
wpa_passphrase <ssid> <passphrase>
```

In order to facilitate onboarding in identity-based WPA2/WPA3 networks, the program has been extended to generate corresponding configuration files. The modified command line tool can be called with an optional parameter:

```
wpa_passphrase <ssid> <passphrase> [MAC address]
```

The input format of the MAC address must be either colon-hexadecimal notation or simple hexadecimal notation without colons. Once a valid MAC address is provided, the passphrase is interpreted as master-secret and the resulting configuration file contains an iPSK instead of a regular PSK. Using this method, no prior interaction with the AP is needed to readout the identity-based WPA-passphrase.

5.3. Proof of Concept Code

The source code of the PoC implementation, including `hostapd`, `wpa_passphrase` and sample configuration files, is available on GitLab.

```
https://gitlab.informatik.hu-berlin.de/panizzaj/iPSK/
```

Whenever the PoC implementation is mentioned throughout this thesis, the commit with the following hash value is referenced:

```
master: f7715f8e47081ce48444dadc092431897c84f539
```

6. Results and Critical Discussion

To evaluate the proposed modification of `hostapd`, the following sections analyze the runtime, memory usage and size of the binary compared to the original version. Later in this chapter it will be described in detail which attacks on Wi-Fi networks can be prevented by identity-based authentication. This is followed by a discussion of the advantages and disadvantages of the proposed concepts and how the key derivation function could be optimized.

6.1. Runtime Analysis

PBKDF2 is the part of the key derivation function that deliberately takes up the most runtime. Normally `hostapd` calculates the PSK from the given WPA-passphrase during the initialization phase. Once a supplicant connects, a simple lookup of the value is performed. Since the identity-based WPA-passphrase and iPSK derivation is dependent on the MAC address of the supplicant, it can only be calculated once the supplicant associates itself. It results in a delay of the connection establishment. To measure this additional execution time, it is sufficient to measure the runtime of the additional instructions added to the source code. Since the handshake has not been changed, it is expected that the execution time of the entire authentication process will not change except for the additional runtime of the key derivation.

The following benchmarks were measured on the FRITZ!Box 7530, which has the following hardware and system specifications:

- CPU: Qualcomm IPQ4019 (4 x 716 MHz)
- RAM: 256 MB
- OS: OpenWrt
- Kernel: Linux Kernel version 4.14.171

As we can see in table 1, deriving an identity-based key takes considerably longer for WPA2-Personal than for WPA3-Personal. This is due to the fact that in WPA2 the key stretching function PBKDF2 is executed twice. The first time in the key derivation function to generate the identity-based WPA-passphrase and the second time to derive the iPSK from the passphrase. WPA3-Personal directly uses the identity-based WPA-passphrase for SAE and therefore does not require any further derivation. The same applies to WPS as the identity-based passphrase is transferred and no iPSK is needed. The minor time difference between WPA3-Personal and WPS is caused by the inherent fluctuation in each execution.

Connection	δ**t** [ms]	σ [ms]
WPA2-Personal	150.891	0.535
WPA3-Personal	94.944	0.461
WPS	95.015	0.362

Table 1: Additional runtime of the identity-based implementation. 10 iterations were performed to measure the average execution time and standard deviation.

6.2. Binary Size Comparison

The available storage space on common routers is often quite limited. For this reason a binary size analysis of the program becomes essential. Table 2 shows the size of both programs in bytes. The modification increases the binary size by 0.25 %, which can be considered trivial.

Binary	**Size** [Byte]
Original	5194256
Adapted	5207480

Table 2: Binary size of original version compared to the adapted version of `hostapd`.

6.3. Memory Comparison

In order to compare the memory consumption, the heap profiler *Massif* from the *valgrind* suite was used to records the memory use of the heap. For the measurement `hostapd` was executed in WPA2/WPA3-Personal mixed mode. After the network was set up, three supplicants were connected first via WPA2-PSK, then via WPA3-SAE and finally via WPS. Figure 1 shows the memory heap consumption of the modified version of `hostapd` whereas figure 2 shows the consumption of the original version.

It is clearly recognizable that the protocol adaptation has little effect on the memory. During WPA2-PSK connection establishment, the identity-based implementation allocates about 70 kB of heap memory for a very short period of time, whereas the genuine implementation only requires about 45 kB. The comparison of WPA3-SAE and WPS in both figures does not show any noticeable difference. It should be noted that the memory consumption varies from execution to execution, as retransmissions may occur due to radio channel interference. Since all received packets from `hostapd` are dynamically stored on the heap, retransmissions increase memory consumption and bias the result.

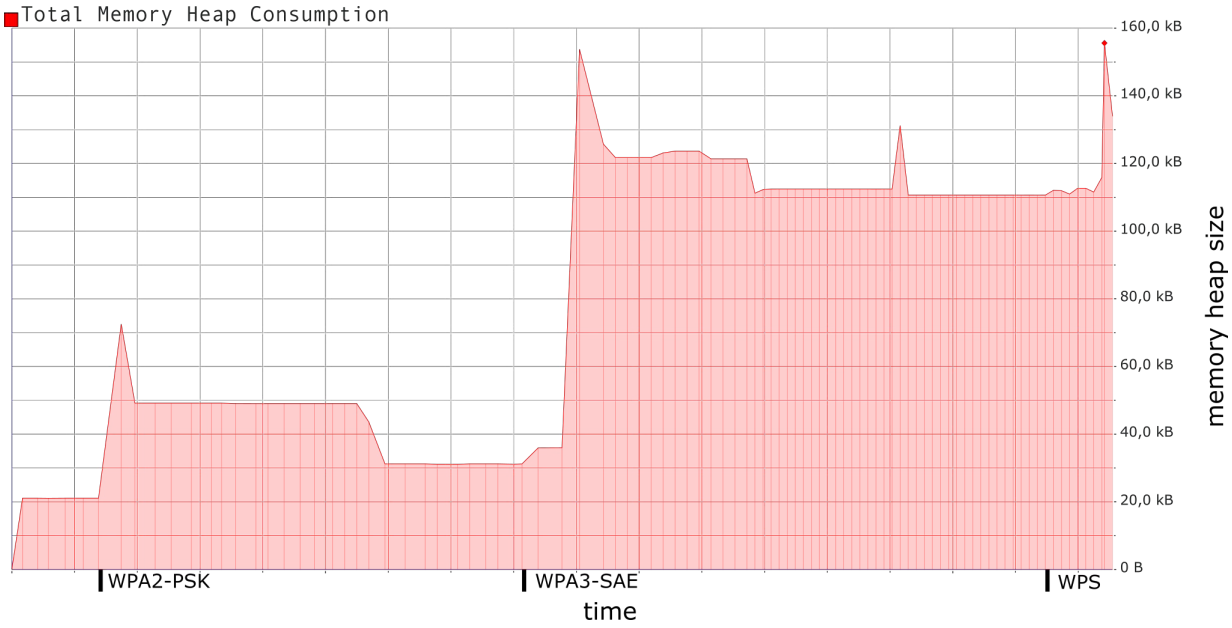

Figure 1: Memory heap consumption of the adapted version of `hostapd`. Generated with `massif-visualizer`

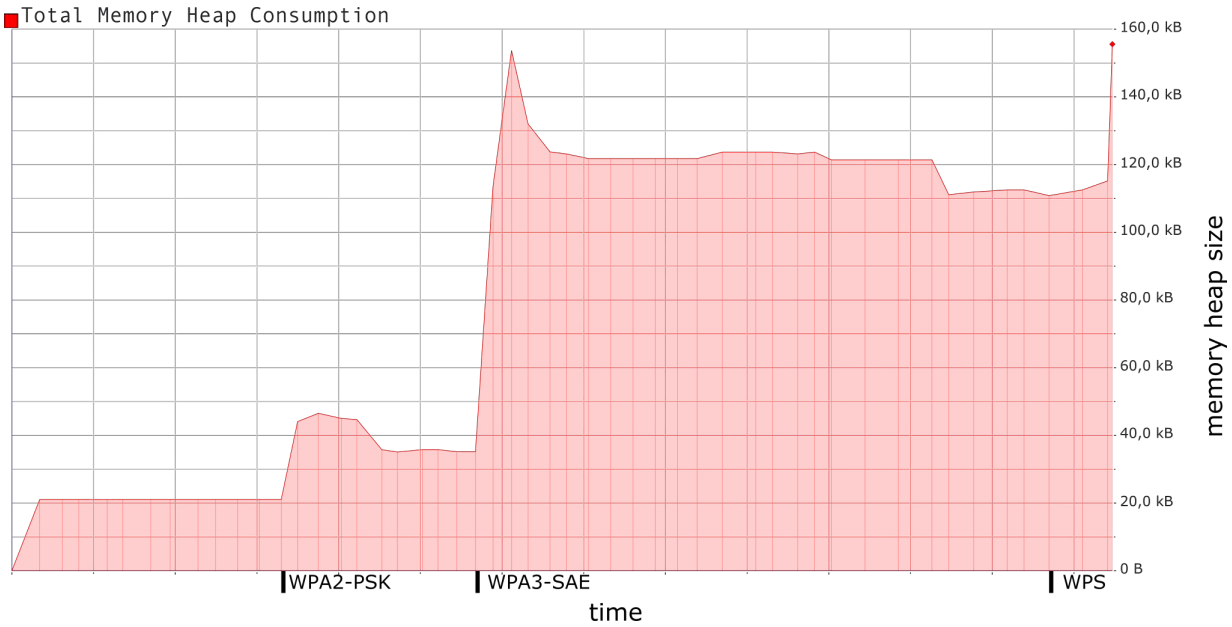

Figure 2: Memory heap consumption of the original version of `hostapd`. Generated with `massif-visualizer`

6.4. Discussion

As we have seen in chapter “Weaknesses of Wi-Fi Protected Access 2 Personal”, there are several ways to attack a WPA2-PSK network. WPA3-SAE prevents some of these attacks, but certainly not all, as outlined in chapter “Weaknesses of Wi-Fi Protected

Access 3 Personal". The following section illustrates, which of the described attacks vectors are not feasible when using identity-based authentication.

In WPA2-PSK it is possible to encrypt and decrypt packets from other clients on the network, assuming one has possession of the PSK. Identity-based WPA2-PSK prevents this because each supplicant has its own PSK and they are not shared. WPA3-SAE also solves this problem whether or not identity-based keys are used.

In WPA2-PSK an adversary can impersonate a client by hijacking an active session. Since encryption and decryption is needed for active session hijacking, identity-based WPA2-PSK also prevents this attack. WPA3-SAE has prevented this attack all along, it makes no difference whether identity-based keys are used or not. However, with impersonations in terms of a new session, the situation is different: Neither WPA2-PSK nor WPA3-SAE can distinguish clients with fake MAC addresses from the original ones. Identity-based authentication offers a remedy in both cases. Because MAC addresses are key-bound, a rogue client cannot use their identity-based key to connect with a spoofed MAC address. Nevertheless, it would be conceivable that the client adopts a spoofed MAC address before the onboarding process. However, the generated key would force the client to remain in their false identity, because as soon as the identity is changed, access to the network is lost. Since onboarding requires user interaction, it is unlikely that a device with the wrong MAC address will be added to the network. Nevertheless, the system follows a *Trust On First Use* (TOFU) model because the legitimacy of the MAC address is not always checked (e.g. WPS).

It has been shown that an Evil Twin attack is feasible in both WPA2-PSK and WPA3-SAE networks. From the moment an identity-based key is used, the network cannot be cloned because the key is unknown to the adversary. This applies equally to both WPA2 and WPA3.

Last but not least, it is worth taking a closer look at ARP spoofing. Unfortunately, identity-based WPA networks do not protect against this attack. Since the GTK is shared among all supplicants, an adversary can still poison a victim's ARP cache by sending unsolicited ARP replies. Once the ARP cache is altered, all outgoing packets are addressed to the adversary but still encrypted with the victim's identity-based key. The AP receives the packets, but before they are forwarded, they are decrypted with the victim's key and re-encrypted with the adversary's key. Since the AP acts as a proxy and re-encrypts all packets according to the recipient's key, it does not matter whether identity-based keys are used or not. As mentioned above, the AP could use *client isolation* to prevent ARP spoofing. Table 3 shows a list of all mentioned attacks and distinguishes for which key management protocols they can be performed.

The analysis shows that identity-based keys in WPA networks offer numerous advantages. However, there are different ways to implement this concept. The presented version differs fundamentally from already existing implementations: the identity-based keys are generated at runtime and never need to be stored on an AP or authentication server. This has the apparent advantage that even in networks with several hundred clients only a constant amount of disk space is needed for passwords. Furthermore, APs, repeaters and authentication servers can synchronize credentials in $\mathcal{O}(1)$, because only the master-secret and the SSID must be transferred. The data being transmitted is

Attack	WPA2-PSK	WPA3-SAE	identity-based WPA2-PSK	identity-based WPA3-SAE
En/Decryption	Yes	No	No	No
Impersonation with session hijacking	Yes	No	No	No
Impersonation with new session	Yes	Yes	No	No
Evil Twin	Yes	Yes	No	No
ARP spoofing	Yes	Yes	Yes	Yes

Table 3: Comparison of key management protocols, whether or not they are vulnerable to different attacks.

not proportional to the number of passwords. In addition, it is sufficient for a network administrator to remember only the master-secret, since all other keys can be generated from it.

However, there are also disadvantages that must be mentioned. One drawback concerns the key revocation: as soon as an identity-based key is compromised and the master-secret is replaced, all keys are changed. The protocol design prohibits changing only the compromised key, which could lead to large administrative effort if there are many clients on the network. Stateful implementations, where MAC addresses and the associated keys are stored in a file or database, do not suffer from this shortcoming.

If a device does not support WPS, an administrator needs its MAC address for onboarding. For an average user, this can already be a significant challenge. Devices with a screen usually have the ability to display the address, but IoT devices or printers, for example, rarely allow this. To work around this problem, the described web interface of a router for retrieving identity-based keys could be programmed as follows: The web interface shows a history of all supplicants who have unsuccessfully tried to connect to the network. A timestamp indicates when each authentication attempt was made and the MAC address uniquely identifies the device. A button next to each entry in the history allows direct key generation for the identity. Once an unsuccessful authentication attempt has been made with a dummy password, the corresponding device appears in the history. Thereupon, the identity-based passphrase can be generated at the click of a button, without the administrator having to enter a MAC address manually in the first place.

The main drawback, however, concerns the runtime. Since keys are derived at runtime, this inherently increases the execution time of the authentication process. While it could be argued that this constant factor does not have a major impact on the quality of the connection, as authentication is ideally performed only once, there are several ways to address this problem. The following section discusses some changes that could be made to either improve performance or enhance security or usability.

First of all, a larger cache can be used to ensure that re-authenticaions do not result in the keys being re-derived. If a history of connected MAC addresses were

stored persistently, one could also pre-calculate and cache all keys during the boot process. But since the requirement was to develop a stateless protocol modification, the latter is to be seen as a secondary solution. Moreover, it is possible not to stretch each HMAC at runtime, but the master-secret during the initialization phase. During runtime, the cached, stretched master-secret is used as key to generate the HMAC. This optimization results in key stretching not being performed at runtime at all, except for iPSK derivation from the identity-based passphrase.

Assuming that the master-secret is only stretched once, another key stretching method comes into question, which is more memory expensive but also offer stronger security against parallelization. PBKDF2 is considered vulnerable to attacks with special hardware such as *Graphics Processing Units* (GPUs), *Field Programmable Gate Arrays* (FPGAs) and *Application Specific Integrated Circuits* (ASICs) [31]. The computation of the hashes can be massively parallelized with such hardware. The key stretching function *scrypt*, on the other hand, takes advantage of the fact that memory is relatively expensive and allocates an excessively large vector of RAM thus driving up the cost of using specialized hardware. Nevertheless, no matter how much costs are imposed on an adversary per password attempt, these costs must also be paid by the AP. The complexity of the problem is not increased by key stretching, only a constant is multiplied.

On the other hand, one can also argue that key stretching is not necessary at all. In this PoC implementation it is assumed that the master-secret is defined manually by a user. When using user-defined keys, it is always recommended to use key stretching as further protection, as humans tend to use weak, easy-to-remember passwords that are prone to dictionary attacks. However, if e.g. a randomly generated 256 bit key is used as master-secret, this provides enough entropy to be protected against brute force even without key stretching. The entropy of a randomly generated 256 bit password corresponds to a randomly generated 43 character ASCII-string. But as humans rarely create randomly generated passwords, key stretching is not omitted here.

Furthermore, it should be mentioned that an HMAC is not necessarily required. Since we only want to take advantage of the one-way function property of SHA, it is also possible to concatenate the MAC address with the master-secret and hash the result with SHA. This method is usually not recommended because all hash functions based on the *Merkle-Damgård construction* (e.g. SHA2) are vulnerable to the *length extension attack*, where an adversary takes a message and its HMAC signature, and use this to construct a longer message with the same signature. However, this attack is irrelevant in this case because the generated HMAC is not used as message authentication code and never shared among clients. But since the calculation of an HMAC is about as expensive as directly executing the hash function, there is no reason not to use it.

Last but not least a different binary-to-text encoding could be used instead of Base64. In order to avoid both non-alphanumeric characters (“+” and “/”, which are part of Base64’s character set) and letters which might look ambiguous when printed (“I”, “O”, “l”, “0”) *Base58* could replace the former.

7. Conclusion

The thesis shows that identity-based authentication in WPA2/WPA3-Personal networks can counteract the threats posed by the proliferation of IoT devices in home networks. It turned out that WPA2-PSK benefits even more from the protocol adaptation, since WPA3-SAE inherently prevents the decryption of third-party unicast traffic. Nevertheless, the extension promises effective protection against impersonation attacks in the case of WPA3-SAE. Thanks to this property, routers can create profiles that reliably prevent untrusted devices from accessing the wide area network, making it impossible to leak user data. Devices for which a server connection is essential can be reliably isolated from the rest of the network. This makes it impossible for hackers or malware to use a compromised device as a gateway for further attacks. Given that it will probably take several years before most of the supplicant hardware is PMF-capable, a rapid transition to WPA3 is not expected. Until this point, identity-based authentication even provides the desired confidentiality that is missing in WPA2-PSK.

The proposed protocol adaptation differs from existing implementations because it has a stateless design. Neither identities nor identity-based keys need to be stored on the AP or authentication server. The PoC demonstrates that only minor modifications of `hostapd` are necessary to implement the presented concept. The described optimizations could further enhance the runtime of the authentication process, while different keys stretching methods can be applied depending on the threat level. Vendors of wireless distribution systems could deploy identity-based authentication as an optional feature on their routers in the form of a firmware update. Ideally, integration into the standard would also be desirable in order to disseminate a uniform implementation. Concerning usability, extra effort is to be expected during the onboarding process. Unless WPS is used, a user must actively request identity-based keys before entering them at the supplicant. However, assuming that this is possible via the router's straightforward web interface, this task becomes trivial.

References

[1] Louis Columbus. *Roundup Of Internet Of Things Forecasts And Market Estimates*. https://www.forbes.com/sites/louiscolumbus/2018/12/13/2018-roundup-of-internet-of-things-forecasts-and-market-estimates/, 2018. Accessed: July 27, 2020.

[2] Paul Marrapese. *P2P Weakness Exposes Millions of IoT Devices*. https://krebsonsecurity.com/2019/04/p2p-weakness-exposes-millions-of-iot-devices/, 2019. Accessed: July 27, 2020.

[3] Orlando Arias, Jacob Wurm, Khoa Hoang, and Yier Jin. Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, apr 2015.

[4] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. DDoS-capable IoT malwares: Comparative analysis and mirai investigation. *Security and Communication Networks*, 2018, 2018.

[5] Erik Tews. *Breaking 104 bit WEP in less than 60 seconds*. Master's thesis, TU Darmstadt, 2007.

[6] Erik Tews and Martin Beck. Practical attacks against WEP and WPA. In *Proceedings of the second ACM conference on Wireless network security - WiSec '09*. ACM Press, 2009.

[7] IEEE Jouni Malinen. 802.11mb issues list v12. https://mentor.ieee.org/802.11/file/08/11-08-1127-12-000m-tgmb-issues-list.xls, 2009. Accessed: July 5, 2020.

[8] BSI TR-02102-1 *Kryptographische Verfahren: Empfehlungen und Schlüssellängen Version: 2020-01*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf, 2020.

[9] Radomir Prodanovic and Dejan Simic. A survey of wireless security. *Journal of Computing and Information Technology*, 15(3):237, 2007.

[10] David A. Westcott David D. Coleman. *CWNA Certified Wireless Network Administrator Study Guide*. John Wiley & Sons Inc, 2018.

[11] Md Sohail Ahmad. WPA 2 Hole196. In *AirTight Networks Presentation*, 2012.

[12] Wesam Lootah, William Enck, and Patrick McDaniel. TARP: Ticket-based address resolution protocol. *Computer Networks*, pages 4322 4337, 2007.

[13] D. Bruschi, A. Ornaghi, and E. Rosti. S-ARP: a secure address resolution protocol. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.* IEEE.

[14] Hossen Mustafa and Wenyuan Xu. CETAD: Detecting evil twin access point attacks in wireless hotspots. In *2014 IEEE Conference on Communications and Network Security*. IEEE, oct 2014.

[15] Dan Harkins. Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks. In *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*. IEEE, 2008.

[16] Alfred John Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Taylor & Francis Inc, 1996.

[17] Dan Harkins. Dragonfly Key Exchange. RFC 7664, November 2015.

[18] *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 4: protected management frames*. Institute of Electrical and Electronics Engineers, New York, 2009.

[19] Dirk Westhoff. *Mobile Security*. Springer-Verlag GmbH, 2020.

[20] CiscoSystems. iPSK (Identity Pre-Shared-Key) Manager portal server for ISE. `https://community.cisco.com/t5/security-documents/ipsk-identity-pre-shared-key-manager-portal-server-for-ise/ta-p/3904265`. Accessed: June 26, 2020.

[21] Lancom. LEPS / PPSK (Private Pre-Shared Key). `https://www.lancom-systems.de/pdf/techpapers/TP_PPSK-LEPS_DE.pdf`. Accessed: June 26, 2020.

[22] Aerohive. Private Pre-Shared Key: Simplified Authentication. `https://www.aerohiveworks.com/Authentication.asp`. Accessed: June 26, 2020.

[23] Ruckus. Enabling DPSK for a WLAN. `https://docs.ruckuswireless.com/unleashed/200.3/t-EnableDPSKforWLAN.html`. Accessed: June 26, 2020.

[24] Aruba. Configuring MPSK. `https://www.arubanetworks.com/techdocs/ClearPass/6.8/Guest/Content/AdministrationTasks1/Configuring-MPSK.htm`. Accessed: June 26, 2020.

[25] CambiumNetworks. ePSK - Multiple Pre-Shared Keys. https://community.cambiumnetworks.com/t5/cnPilot-E-Series-Enterprise-APs/ePSK-Multiple-Pre-Shared-Keys/m-p/106597#. Accessed: June 26, 2020.

[26] WiFi-Aliance. hostapd. https://w1.fi/hostapd/. Accessed: June 28, 2020.

[27] WiFi-Aliance. hostapd changelog. https://w1.fi/cgit/hostap/plain/hostapd/ChangeLog. Accessed June 28, 2020.

[28] 802.11i-2004 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, jul 2004.

[29] Liang Liu, Zhaoyang Han, Liming Fang, and Zuchao Ma. Tell the device password: Smart device wi-fi connection based on audio waves. *Sensors*, page 618, feb 2019.

[30] Toni Perković, Tonko Kovačević, and Mario Čagalj. BlinkComm: Initialization of IoT devices using visible light communication. *Wireless Communications and Mobile Computing*, 2018.

[31] Markus Kasper Christof Paar Tolga Yalcin Ralf Zimmermann Markus Dürmuth, Tim Güneysu. *Evaluation of Standardized Password-Based Key Derivation against Parallel Processing Platforms. In Computer Security ESORICS 2012.* Springer Berlin Heidelberg, 2012.

A. Flashing OpenWrt to the FRITZ!Box 7530

The following is a step-by-step documentation on how to flash OpenWrt 19.07.02 on the AVM FRITZ!Box 7530. An Ubuntu 18.04.4 LTS running inside a virtual environment was used to accomplish the flashing process. There are two types of images used in this documentation: The `initramfs-kernel` image is the first part of a multi-stage installation. It is used as a one-time boot as a stepping stone toward installing the regular `squashfs` version. The `initramfs` version runs entirely from RAM and is therefore not persistent.

```
# Download flashing scipt
wget "https://git.openwrt.org/?p=openwrt/openwrt.git;a=blob_plain;f=scripts/
    flashing/eva_ramboot.py;hb=HEAD" -O eva_ramboot.py

# Download OpenWrt images
wget https://downloads.openwrt.org/releases/19.07.2/targets/ipq40xx/generic/openwrt
    -19.07.2-ipq40xx-generic-avm_fritzbox-7530-squashfs-sysupgrade.bin -P $HOME
wget https://downloads.openwrt.org/releases/19.07.2/targets/ipq40xx/generic/openwrt
    -19.07.2-ipq40xx-generic-avm_fritzbox-7530-initramfs-fit-uImage.itb -P $HOME

# Download bootloader
wget https://downloads.openwrt.org/releases/19.07.2/targets/ipq40xx/generic/u-boot-
    fritz7530/uboot-fritz7530.bin -P $HOME

# Install tfdp server
sudo apt-get install atftpd

# inetd controls atftp. We want to change this behavior
sudo /etc/init.d/inetutils-inetd stop
vim /etc/default/atftpd # Change USE_INETD=true to USE_INETD=false
sudo /etc/init.d/inetutils-inetd start

# Copy the initramfs image to the tftp server location and rename it
sudo cp openwrt-19.07.2-ipq40xx-generic-avm_fritzbox-7530-initramfs-fit-uImage.itb
    /srv/tftp/FRITZ7530.bin

# Start the tftp server
sudo service atftpd start

# Assign a static IP on the physical LAN interface
nmcli con add type ethernet ifname enp0s8 con-name wired ipv4.addresses "
    192.168.178.10/24" ipv4.gateway "192.168.178.1" ipv4.method manual

# Connect one LAN port of the FRITZ!Box (not the WAN port) to the LAN
# port of the computer with an ethernet cable. Important: The LAN
# interface of the virtual machines host OS must be disabled!
```

```
# Power on the box and run immediately the following command. Try it
# several times if a "connection refused" occurs, the operation must
# be exactly in time
./eva_ramboot.py --offset 0x85000000 192.168.178.1 uboot-fritz7530.bin

# Now the bootloader should be up and running. You can confirm it by
# executing the following command in a new terminal window. You
# should see a tftp request asking the client with the IP
# "192.168.1.70" for the file "FRITZ7530.bin"
sudo tcpdump -i enp0s8 port 69 -v

# Assign yourself the IP address 192.168.1.70/24 and adjust the
# default gateway
nmcli con add type ethernet ifname enp0s8 con-name wired ipv4.addresses "
    192.168.1.70/24" ipv4.gateway "192.168.1.1" ipv4.method manual

# With the correct IP address set, our tftp server will answer the
# request and transfer our initramfs image. The FRITZ!Box is then
# rebooted

# Login onto the router
ssh root@192.168.1.1

# Set a root password and logout again
passwd
exit

# Copy the persistent squashfs image and the bootload to the box
scp openwrt-19.07.2-ipq40xx-generic-avm_fritzbox-7530-squashfs-sysupgrade.bin uboot
    -fritz7530.bin root@192.168.1.1:/tmp/

# Login to the box again. All following commands are executed via ssh
# on the FRITZ!Box
ssh root@192.168.1.1

# Install uboot persistently
mtd write /tmp/uboot-fritz7530.bin uboot0
mtd write /tmp/uboot-fritz7530.bin uboot1
ubirmvol /dev/ubi0 --name=avm_filesys_0
ubirmvol /dev/ubi0 --name=avm_filesys_1

# Flash the persistent firmware
sysupgrade -n /tmp/openwrt-19.07.2-ipq40xx-generic-avm_fritzbox-7530-squashfs-
    sysupgrade.bin
```

B. Extroot configuration

The following describes how an external USB flash drive can be used to extend the root filesystem of the FRITZ!Box 7530. With an extended file system it is possible to install additional packages without having to rely on the limited internal disk space of the router. In most supported devices OpenWrt splits the internal storage into `rootfs` and `rootfs_data` partitions which are merged together into a single writable overlay filesystem. *Extroot* works by setting another overlay partition in the external storage device, and during boot this new overlay partition will be mounted over the internal storage's overlay partition. Note: All following command are executed on the FRITZ!Box 7530 via SSH.

```
# Install required packages
opkg update && opkg install block-mount kmod-fs-ext4 kmod-usb-storage e2fsprogs
    kmod-usb-ohci kmod-usb-uhci fdisk

# Configure /etc/config/fstab to mount the rootfs_data in another
# directory in case you need to access the original root overlay to
# change your extroot settings
DEVICE="$(sed -n -e "/\s\/overlay\s.*$/s///p" /etc/mtab)"
uci -q delete fstab.rwm
uci set fstab.rwm="mount"
uci set fstab.rwm.device="${DEVICE}"
uci set fstab.rwm.target="/rwm"
uci commit fstab

# Search for the partition name of the USB drive. In my case "sda1"
block info

# Format the partition as ext4
mkfs.ext4 /dev/sda1

# Configure the USB as new overlay via fstab uci subsystem
DEVICE="/dev/sda1"
eval $(block info "${DEVICE}" | grep -o -e "UUID=\S*")
uci -q delete fstab.overlay
uci set fstab.overlay="mount"
uci set fstab.overlay.uuid="${UUID}"
uci set fstab.overlay.target="/overlay"
uci commit fstab

# Transfer the content of the current overlay inside the external
# drive
mount /dev/sda1 /mnt
cp -a -f /overlay/. /mnt
umount /mnt

# Reboot the device
reboot
```

C. Cross-Compilation for the FRITZ!Box 7530

The following steps were made to setup the build system on a Ubuntu 18.04.4 LTS running inside a virtual environment and cross-compile `hostapd`.

```
# Install all essential packages
sudo apt update
sudo apt install subversion build-essential libncurses5-dev zlib1g-dev gawk git
    ccache gettext libssl-dev xsltproc zip git-core unzip subversion mercurial

# Download build system
git clone https://github.com/openwrt/openwrt.git

# Configure build system
cd openwrt
./scripts/feeds update -a
./scripts/feeds install -a
make menuconfig

# The last command opens an interactive configuration menu. Select
# the following:
# --> Target system --> Qualcomm Atheros IPQ40XX
# --> Subtarget --> Generic
# --> Target profile --> AVM FRITZ!Box 7530
# --> Libraries --> SSL --> libopenssl
# --> Libraries --> libnl

# Build the toolchain. This takes approx. 2 hours
make -j 4

# Add the following lines to the hostapd Makefile
ifdef CONFIG_FRITZBOX_7530
CC := arm-openwrt-linux-gcc
STAGING_DIR := /home/jonas/openwrt/staging_dir
TOOLCHAIN_DIR := ${STAGING_DIR}/toolchain-arm_cortex-a7+neon-vfpv4_gcc-8.4.0
    _musl_eabi
TARGET_DIR := ${STAGING_DIR}/target-arm_cortex-a7+neon-vfpv4_musl_eabi
CFLAGS += -I${TARGET_DIR}/usr/include
LIBS += -L${TARGET_DIR}/usr/lib
LDCFLAGS := ${TOOLCHAIN_DIR}/usr/lib
LD_LIBRARY_PATH := ${TOOLCHAIN_DIR}/usr/lib
PATH := ${TOOLCHAIN_DIR}/bin:${PATH}
export PATH
export STAGING_DIR
endif

# Add the following line to the hostapd.conf. When this statement is
# omitted, hostapd is compiled natively for the Ubuntu system.
CONFIG_FRITZBOX_7530=y

# Now hostapd can be cross-compiled as usual with make.
```

D. Configuration file for identity-based WPA2/WPA3-Personal mixed mode

```
# Change wlan0 to your wireless device
interface=wlan0

# Driver interface nl80211 is used with all Linux mac80211 drivers
driver=nl80211

# Define a network name
ssid=Example

# Sets the operating mode of the interface and the allowed channels
hw_mode=g

# Use 802.11n to increase data rates
ieee80211n=1

# Set a channel. Channels 1-13 are allowed with 802.11n
channel=1

# Setting this variable configures the AP to require WPA (either
# WPA-PSK or WPA-RADIUS/EAP based on other configuration)
wpa=2

# WPA pre-shared key for WPA-PSK. This can be either a PSK, entered
# as a 256-bit secret in hex format (64 hex digits), or an ASCII
# passphrase (8..63 characters) that will be converted to PSK.
# When iPSK mode is enabled, the wpa_passphrase acts as master-secret
wpa_passphrase=mastersecret

# Set of accepted key management algorithms. The entries are
# separated with a space.
wpa_key_mgmt=WPA-PSK SAE

# Set of accepted cipher suites
rsn_pairwise=CCMP

# Specify whether protected management frames (PMF) is enabled
# 0 = disabled
# 1 = optional
# 2 = required
ieee80211w=1

# Enable iPSK policy
ipsk=1

# Enable internal EAP server for EAP-WSC (part of WPS)
eap_server=1
```

```
# Do not allow external WPS registrars
wps_state=2

# Enable control interface for WPS PBC/PIN entry
ctrl_interface=/var/run/hostapd
```

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den August 7, 2020 .....