2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #39754

WPA3 Personal and Enterprise Wireless Security Algorithm Labs for Undergraduate Level

Dr. Emil H. Salib, James Madison University

Professor in the College of Integrated Science & Engineering (CISE) at James Madison University (JMU). Current Teaching - Networking, Network Security, Introductory Programming, Introductory Database Systems, Introductory Web Technology Current Research - Virtualization & Cloud Computing, Blockchain Technology, Software Defined Network, Wireless Networking and Security

WPA3 Personal and Enterprise Wireless Security Algorithm Labs for Undergraduate Level

Dr. Emil H. Salib, salibeh@jmu.edu,
College of Science and Engineering (CISE),
James Madison University (JMU), Harrisonburg, VA 22807

1 Introduction

The demand for high-speed and secure wireless local area network (WLAN) continues to grow exponentially. As a result, the IEEE 802.11/Wi-Fi technology continues to evolve to meet the needs of both the enterprise and home network environments. Wireless security has been a special area of interest to users, vendors and researchers. One of the most important and recent advancements is the introduction of Wi-Fi Protected Access 3 (WPA3) personal and enterprise to address some of the serious security vulnerability issues of WPA2/WPA. Ensuring that wireless security curriculum is up to date has been a challenge due to the reluctance of the Access Point (AP) and Wireless Station (STA)/Client vendors to implement WPA3 in their currently affordable products. Also, it has been a challenge to be able to access information from the vendors products to develop and deliver practical and robust lab exercises on the security algorithms adopted by the WPA3-Personal and Enterprise standards.

In this paper, we present our solution to address these challenges where we adopted the approach of creating our own affordable, customizable and flexible wireless access point (AP) and wireless stations (STAs)/clients. These customizable wireless components are created using open source software running on the affordable Raspberry Pi 3B (RPi-3B) units. The RPi-3B units must be equipped with USB wireless adaptors with wireless chips that support WPA3 protocols and requirements. The open source software packages adopted in our solution are the Linux wpa_ supplicant (STA/wireless station/supplicant), hostapd (AP/access point/authenticator), and freeradius (radius/authentication server) latest releases.

We also share how we were able to develop and deliver a well-thought-out, thoroughly tested and exercised by our students (in the Information Technology (IT) program) lab exercises on the latest WPA3 wireless security algorithms (including Simultaneous Authentication of Equals (SAE)/Dragonfly based on National Institute of Standards and Technology (NIST) elliptical curve cryptography). In the original experimental lab exercises, our students were provided with Python script implementation of some of the security algorithms and directed to create their own for others. We also provided them with a validation methodology to enable them to extract the input ingredients required by and the target output expected from the security algorithm scripts. The students feedback and suggestions confirmed the lack of robust materials on the WPA3 security algorithms at the undergraduate level. In this paper, we present the Python implementation scripts of **all** the

lab exercises so that educators and learners can have readily access to the teaching materials we developed and tested on WPA3 security algorithms.

The paper is organized as follows. In Section 2, we provide an account of related work along with a brief overview of the project opportunity, objectives, and proposed solution. In Section 3, we offer a brief description of the solution components designed, tested and exercised by our students. Section 4 focuses on the solution component configurations in support of the lab exercises on WPA3 security algorithms and in Section 5 we detail the lab exercises developed, tested and exercised. In Section 6, we offer a discussion including our students feedback and suggestions, our conclusions, and potential next steps.

2 Related Work and Our Project

In this section, we provide an account of related work. In addition, we describe the project opportunity, goal, objectives, and proposed solution.

2.1 Related Work

In the process of tackling the challenges presented by the introduction of the WPA3 security mode, we identified three training and education categories of related work that we should explore: (1) cybersecurity , (2) WPA3 security algorithms and (3) WPA3 vulnerabilities.

In the first category, Glantz, et al paper [1] highlighted that training so far has only focused on ”traditional” cybersecurity that lightly touches on wireless in undergraduate. They encouraged curricular development that includes critical wireless security hands-on (experiential) training. They also identified a number of available training platforms but concluded that these do not yet offer cybersecurity infrastructures unique to wireless networks. Lixin Wang, et al paper [2] described four course modules on critical cybersecurity topics that can be adopted in college-level cybersecurity courses. One of the modules is on wireless networking security and includes an NDG NETLAB+ based lab [3]. The lab was created to enable the students to decrypt WPA/WPA2 traffic using the airdecap-ng tool and then analyze the decrypted 802.11 wireless packets with Wireshark. The lab lacks the opportunity for the students to gain practice on configuring wireless hardware components such as Access Points and Wireless Clients. Also, it is limited to personal WPA/WPA2 security mode.

We found the following non-academia resources to be good examples of the second category. PRANEETH’S BLOG [4] is a good and informative resource. However, the materials provided in the blogs are insufficient and incomplete to be used as training and/or educational materials. It is written from a tester perspective without presenting or describing the tools necessary to replicate the testing scenarios. Another web site that proved to be of great value is the Asecuritysite by Bill Buchanan [5]. Although it does not specifically cater to wireless security, it provides good resources on foundational cryptographic algorithms that could be utilized in the development of wireless focused training and educational materials.

Efstratios Chatzoglou, et al paper [6] represents an excellent example of research in the third category. It focused on answering the following question ”How is your Wi-Fi connection today? DoS attacks on WPA3-SAE”. Their work investigated WPA3-SAE to identify potential weaknesses or misconfigurations that could lead to successful Denial of Service (DoS) attacks. Their conclusion

was that there is a spectrum of diverse ways of abusing WPA3-SAE to inflict DoS, some of them are due to misconceptions done while implementing the standard. They advocated that their current work can be used as a reference to anyone interested in contributing to WPA3-SAE defenses. Neil Dalal, et al paper [7] presents numerous attacks performed on WPA3 Access Points (APs). Their test results showed that the WPA3 APs are vulnerable to eight out of nine attacks and existing Intrusion Detection System (IDS) (prior to the WPA3 introduction) was unable to detect any of them. They proposed a design for a signature-based IDS, which incorporates techniques to detect all nine WPA3 attacks. They made the code to perform the above attacks as well as that of their updated IDS publicly available in support of future work by the research community at large. Their contributions will also improve the quality and the opportunity of the development of training and educational materials in the wireless cybersecurity domain.

2.2 Our Project

The goal of this project is to introduce WPA3 [8] wireless security algorithms into the current Information Technology (IT) undergraduate curriculum at James Madison University. The target students are juniors and seniors in the IT major. It's expected that these students have completed the IT 215 - Introduction to Telecommunications, Networking and Security and IT 333 - Advanced Networking for Information Technology.

2.2.1 Project Opportunity

The introduction of WPA3 wireless security algorithms, on one hand, has offered significant security enhancements over WPA2 [9]. But on the other hand, it has presented us with a challenge as for the best approach of updating our curriculum and introducing our students to the WPA3 wireless security algorithms. The major changes introduced in WPA3 are (1) SAE/Dragonfly authentication algorithm [10] for WPA3-Personal in place of the Open Authentication adopted in WPA2-Personal, (2) WPA3 mandates the use of Protected Management Frame (PMF) functionality [11], and (3) offers the 192-bit optional security mode in the WPA3-Enterprise environment. The implementation of these algorithms by the vendors of APs and Wireless Adaptors varies significantly as the majority requires new hardware as well as software updates. Also, there are always the potential incompatibility issues between the vendors implementation unless they all have gone through the Wi-Fi WPA3 certification [12].

2.2.2 Project Objectives

To meet the goal of this project and realize the proposed solution, here are the objectives we defined to achieve: (1) validate that the selected components can inter-operate correctly in support of WPA3, (2) design, implement and deliver a set of lab exercises focusing on WPA3 security algorithms with a major focus on the WPA3-Personal mode, (3) assess the quality and effectiveness of the WPA3 lab exercises, and (4) recommend updates, enhancements and future work.

2.2.3 Project Proposed Solution

To solve this problem, we created our own affordable, customizable and flexible wireless access point (AP) and wireless stations (STAs)/clients. These customizable wireless components were created using open source software running on the affordable RPi-3B units [13]. These units must be equipped with USB wireless adaptors with wireless chips that support WPA3 protocols and

requirements (such as TL-WN722N [14]). The open source software packages adopted in our solution are the latest releases of the Linux wpa_supplicant (STA/wireless station/supplicant) [15], hostapd (AP/access point/authenticator) [16] and freeradius (radius/authentication server) [17]. Also, we have made use of an inexpensive access point TP-Link Archer A7 v5 flashed with the latest OpenWrt [18] alternative firmware (with hostapd-common and hostapd-openssl packages) that supports WPA3-Personal and Enterprise.

3 Solution Components

In this section, we present our solution components adopted in support of addressing the challenges described in Section 2. As you may recall, we adopted the approach of creating our own affordable, customizable, and flexible wireless AP and STAs/clients. These customizable wireless components were created using open source hardware and software. Here is a brief description of each of the four (4) key components of the solution.

3.1 TP-Link Archer A7 v5

The Archer A7 [19] has been advertised as a fast IEEE 802.11ac AP with a long range, high speed and reasonable prices. The unit as is (with its original firmware) does not support the WPA3 security mode, nor the PMF/IEEE 802.11w feature. However, thanks to the Qualcomm chip QCA9560 installed on this AP, we were able to flash the unit with OpenWrt 22.03.2 [18], [20] to enable these features on the AP. To expand the wireless security options to include support for WPA3-Enterprise (also known as WPA3- Extensible Authentication Protocol (EAP) in the openwrt definition), we had to replace the default openwrt wpad-basic-wolfssl package with the openwrt "hostapd-openssl" package.

3.2 RPi-3B

First, we prepared the RPi-3B with the Raspberry Pi OS (32-bit) Lite (Raspbian GNU/Linux 11 (bullseye)) [13] for the best performance. RPi-3B built-in wireless chip had the following driver = brcmfmac (version 7.45.98, firmware 01-8e14b897). This wireless chip did not support WPA3 nor PMF. Therefore, we had to equip the RPi-3B units with an external USB wireless adapter with WPA3 and PMF support. TP TL-WN722N v1 USB wireless adaptor [14] is built with a Qualcomm wireless chip: driver ath9k_htc (version 5.25.61-v7, firmware 1.4) and we confirmed that it supports WPA3 and PMF. The wireless adopter, however, it limited us to the use of the 2.4 GHz band. In our opinion, this was not a limitation when it comes to the wireless security algorithms as they are not dependent on the operating frequency bands.

Customizable AP using HOSTAPD on RPi-3B (ap-rasp): For transforming a RPi-3B unit into an AP, we used a bash script [21] which has the capability to install the necessary packages such as hostapd release v2.9, and dnsmasq. It also creates the hostapd.conf and dnsmaq.conf configuration files.

Customizable STA using WPA_SUPPLICANT on RPi-3B (wpa-rasp): Raspberry Pi OS (32-bit) Lite has wpa_supplicant release v2.9 [15] already installed.

Customizable RADIUS Server on RPi-3B (radius-rasp): We had to install several packages to transform a RPi-3B unit into a radius based Authentication Server. These are: freeradius (Version 3.0.21) [17], freeradius-utils (radtest), eapol_test (v2.9) and mariadb-server-10.0.

4 Solution Implementation

In this section, we describe how the components delineated in Section 3 were configured in support of delivering our IEEE 802.11 wireless network supporting WPA3-Personal and Enterprise wireless security modes.

4.1 WPA3-Personal and Enterprise Networks

Figures 1 and 2 depict the arrangements designed and deployed for implementing fully functional WPA3-Personal and Enterprise wireless networks. These networks enabled us to conduct thorough investigation of WPA3 wireless security algorithms in both security modes. Also, they were used for

(1) capturing IEEE 802.11 packets exchanged between the STAs and AP, and

(2) saving wpa_supplicant and hostapd debug message logs from which we were able to extract: the Pairwise Master Key (PMK), Pairwise Transit Key (PTK), the Temporal Key (TK), Key Encryption Key (KEK), Key Confirmation Key (KCK), Group Temporal Key (GTK) and Integrity Group Transient Key (IGTK). These are required for validating the Python implementation scripts of WPA3 security algorithms. Note that TK is required for decrypting the IEEE 802.11 traffic captured by Wireshark.

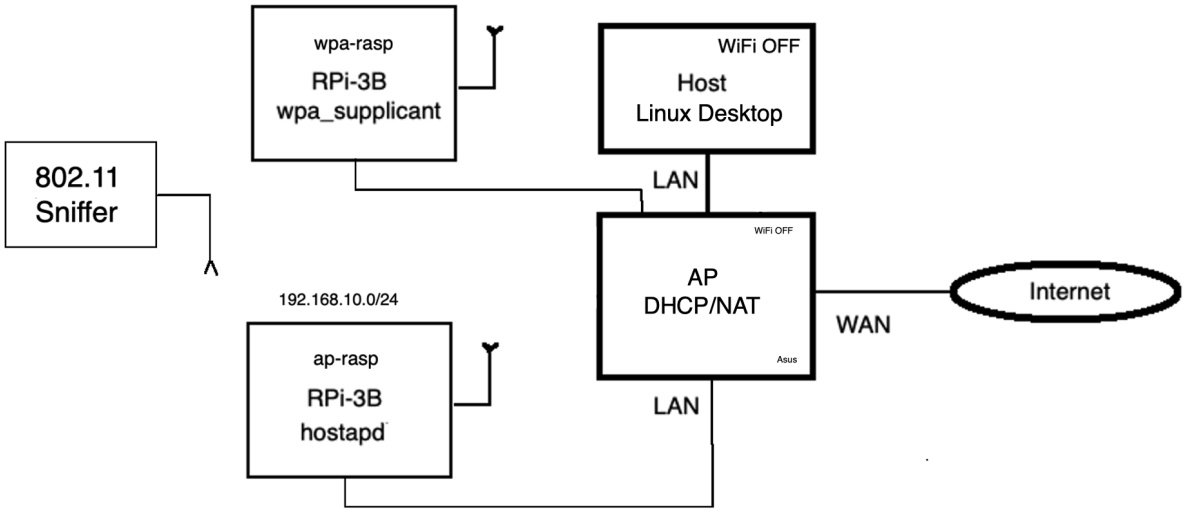

Figure 1: IEEE 802.11 WPA3-Personal Network Arrangement.

In Figures 1 and 2, the IEEE 802.11 sniffer was an iMac desktop running Wireshark and configured

in monitoring/sniffing mode [22] to capture IEEE 802.11 packets exchanged between STA/wireless client (such as, RPi-3B wpa-rasp) and AP/Wireless Access Points (such as, RPi-3B ap-rasp or TP-Link Archer A7 v5).

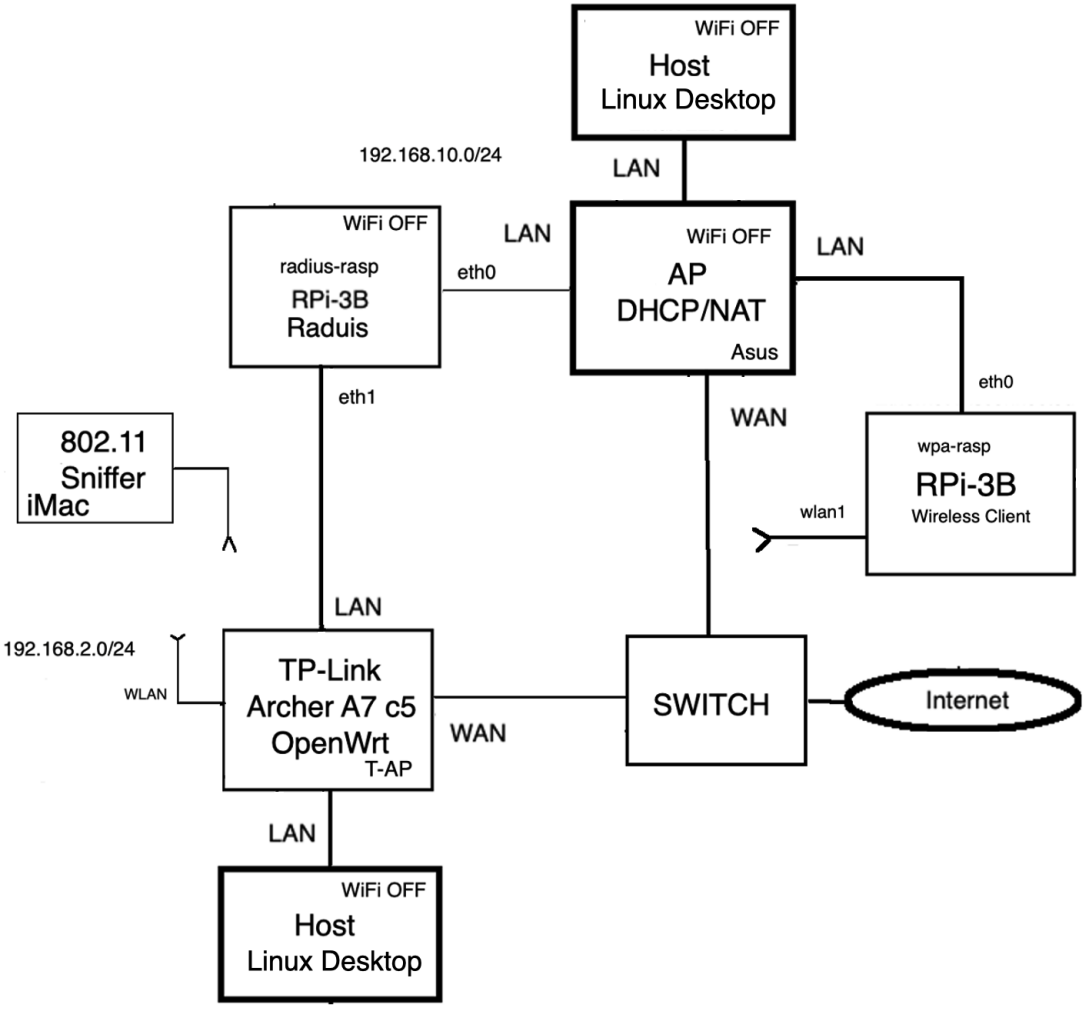

Figure 2: IEEE 802.11 WPA3-Enterprise Network Arrangement.

4.2 WPA3-Personal Security Mode Configurations

In this section, we present the RPi-3B based AP (ap-rasp) hostapd.conf and STA/Wireless Client (wpa-rasp) wpa_supplicant.conf along with the commands required to bring up both the WPA3 AP and STA with the ability to show debug messages including security algorithms related key data.

ap-rasp: In support of WPA3-Personal security mode (including SAE and PMF), the ap-rasp was configured according to the hostapd.conf configuration example given in Appendix A.1.1.

To launch the ap-rasp, we exercised the following command:

sudo hostapd hostapd.conf –dd –K

where -dd is to show even more debug messages and -K is to include key data in the debug messages.

wpa-rasp: In support of WPA3-Personal security mode (including SAE and PMF), the wpa-rasp was configured according to the wpa_supplicant.conf configuration example given in Appendix A.1.2.

To launch the wpa-rasp, we exercised the following command:

sudo wpa_supplicant –D nl80211 –i wlan1 –c wpa_supplicant.conf –dd –K

where -D is the driver, -i interface, -c configuration file, -dd is to show even more debug messages and -K is to include key data in the debug messages.

4.3 WPA3-Enterprise Security Mode Configurations

In this section, we present TP-Link Archer A7 v5 Access Point with OpenWrt configuration, the RPi-3B based radius (radius-rasp) eapol-ttls.conf, and the command required to test the freeradius users and clients. Finally, we present the RPi-3B based STA/Wireless Client (wpa-rasp) wpa_-supplicant.conf and the command required to bring up a WPA3 STA/Wireless Client with the ability to show debug messages including key data.

TP-Link Archer A7 v5 Access Point: We used the TP-Link Archer A7 v5 as the WPA3-Enterprise Access point. The configuration was performed through OpenWrt Luci interface. See Figure 3 for the different WPA3 security mode options available.

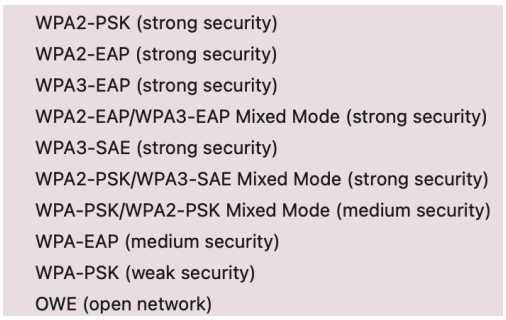

Figure 3: WPA3 Enterprise options of TP-Link Archer A7 v5 flashed with OpenWrt.

radius-rasp: For testing purposes, we created our own certificate authority (CA), and server certificates and private keys. We also added several users (STA/supplicants) and freeradius clients (AP/authenticators) to the radius-rasp server. To launch the radius server and test its configurations, we exercised the following commands in separate shells:

sudo freeradius –X
radtest testuser testcheckout localhost 1812 testsecret
eapol_test –c eapol– ttls .conf –a localhost –p 1812 –s testsecret –r1

where the eapol-ttls.conf is given in Appendix A.2.1.

wpa-rasp: In support of WPA3-Personal security mode (including SAE and PMF), the wpa-rasp was configured according to the wpa_supplicant.conf configuration example given in Appendix A.2.2.

To launch the wpa-rasp, we exercised the following command:

sudo wpa_supplicant –D nl80211 –i wlan1 –c wpa_supplicant.conf –dd –K

where -D is the driver, -i interface, -c configuration file, -dd is to show even more debug messages and -K is to include key data in debug messages.

5 Solution Results (Lab Exercises)

In this section, we present the final version of the lab exercises developed for, exercised by our students and upgraded based on their feedback. All lab exercises, with the exception of one (Exercise 6, share a common procedure presented to our students in the form of a template.

5.1 WPA3 Security Algorithm Validation Procedure Template

The Algorithm Validation Procedure Template consists of six(6) activities:

(1) **Packet Capture and Debug Message Logs:** Identify the Wireshark packet capture file and hostapd, wpa_supplicant debug message logs (to be referred to as logs) to be used for the validation of the Python implementation script of a given WPA3 security algorithm.

(2) **wTarget:** Identify a specific packet and its data (payload) as displayed by Wireshark that are needed for validation. This data is referred to as **wTarget**. Some of the displayed data may be encrypted or wrapped. In this case, the data can be decrypted or unwrapped by installing the TK key extracted from the logs into Wireshark.

(3) **Raw Ingredients:** Identify in the selected packet and/or in the logs the raw ingredients exchanged between the STA and AP that are required by the Python implementation of a given WPA3 security algorithm.

(4) **Input Ingredients:** Input ingredients required by the Python implementation of a given WPA3 security algorithm are those computed and/or extracted from the raw ingredients, such as, nonce for AES CCMP encryption algorithm.

(5) **Python Implementation Script of WPA3 Security Algorithm:** It's designed to take as an input the input ingredients and/or the raw ingredients. Its output is the computed target (to be referred to as **cTarget**.) The Python implementation script should be reviewed and explained in detail by the students.

(6) **cTarget versus wTarget:** Compare the computed target **cTarget** with the Wireshark **wTarget**. The objective is for the students to validate the prepared input ingredients and computed output **cTarget** against the **wTarget**.

5.2 WPA3-Personal Wireless Security Algorithms

In this section, we present WPA3-Personal security algorithms for (1) confidentiality using AES-128 CCMP decryption and AES-128 ECB unwrapping, (2) Message Authentication and Integrity

Code using AES-128 CMAC, (3) Authentication (SAE/Dragonfly), and (4) Key Derivation of Pairwise Transient /key (PTK) and Rekeying of GTK and IGTK.

5.2.1 Decryption and Unwrapping

Exercise 1 : Decryption of AES-128 CCMP Unicast Traffic using WPA TK
The Counter CBC-MAC Protocol (CCMP) encryption protocol is based on the Advanced Encryption Standard (AES) encryption algorithm using the Counter Mode with CBC-MAC (CCM) mode of operation. The CCM mode combines Counter (CTR) mode privacy and Cipher Block Chaining Message Authentication Code (CBC-MAC) authentication. CCM is a generic authenticate-and-encrypt block cipher mode. See Section 12.5.3 in [10] and Figure 4 for more detail. In this exercise, we focus on the decryption of a unicast packet such as an ARP or ICMP. In Exercise 3, we focus on the AES CMAC message authentication and integrity code algorithm. WPA3 uses AES with a 128-bit key, 128-bit block size and 8 Byte MIC. Note that WPA3-Enterprise supports the use of AES-128 CCM and 192-bit session key with AES GCMP-256 mode. The students are to identify the **wTarget**, that is, the message plaintext/payload (after decryption using Wireshark with the WPA Temporal Key (TK) extracted from the logs and the captured 4-way handshake messages.)

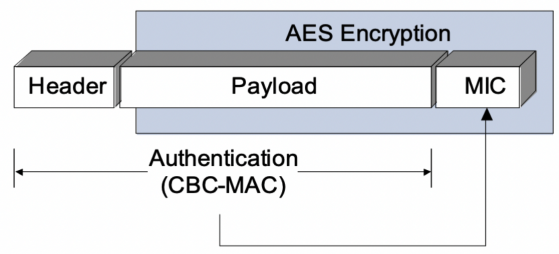

Figure 4: Encryption and Authentication with CCMP using TK or GTK in WPA3.

The students are also to be provided with the Python implementation script of the decryption algorithm given in Appendix B.1 to review and identify the raw ingredients, such as, muted Frame Control field (FC), Address 1 field (A1), Address 2 field (A2), Address 3 field (A3), Sequence Control field (SC), QoS Control field (QC), priority, and Packet Number (PN). They are to use the raw ingredients to compute the input ingredients such as the Additional Authenticated Data (AAD) (see Figure 5) and nonce (see Figure 6) which are required by the the AES-128 MODE CCM decryption algorithm as shown in Appendix B.1.

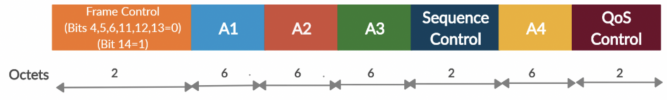

Figure 5: AES Mode CCM AAD Construction [23].

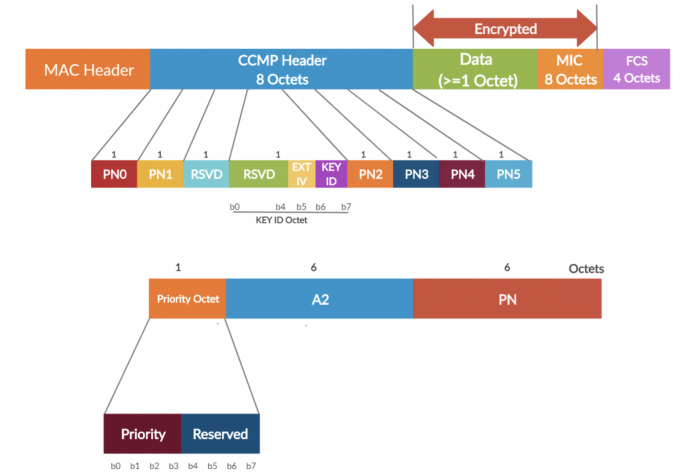

Figure 6: AES Mode CCM Nonce Construction [23].

At this point, they are to compute the **cTarget** message plaintext by executing the Python script with the objective to validate the computed **cTarget** message against that decrypted and displayed by Wireshark, that is, the **wTarget** payload.

Exercise 2: Unwrapping GTK and IGTK in EAPOL-M3/4-way Handshake using WPA KEK
In Message 3 of the WPA 4-way handshake, the group temporal key (GTK) and Integrity Group Temporal Key (IGTK) keys are wrapped using the WPA Key Encryption Key (KEK) (see Figure 7). Key wrapping [24] is the process of encrypting one key using another key, to securely store or transmit it over an untrusted channel. The IGTK is introduced as part of the IEEE 802.11w Protected Management Frame (PMF) feature. In WPA3, the wrapping algorithm is based on AES-128 MODE ECB [25].

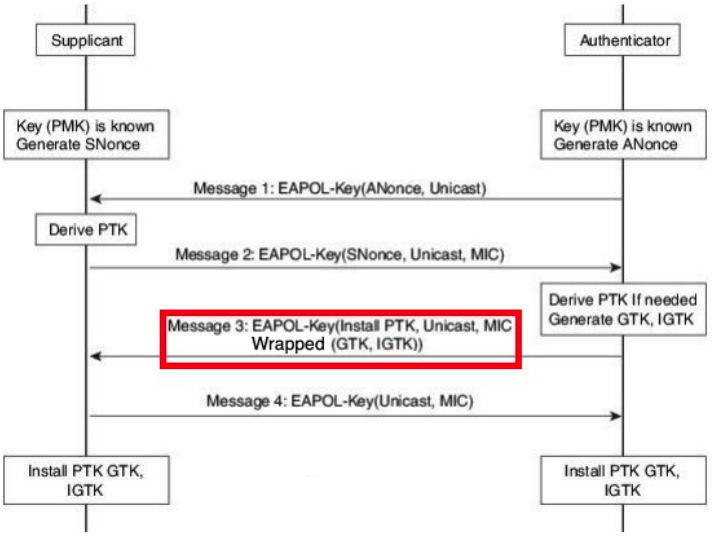

Figure 7: Wrapped and unwrapped GTK and IGTK with KEK in WPA3 EAPOL-M3 [26].

The students are to select a M3 message (WPA KEY Data) exchanged between the STA (wpa-rasp) and AP (ap-rasp) that can be unwrapped by Wireshark by installing the TK. This becomes the **wTarget** value. Once they extract the KEK from the logs, they are to compute the **cTarget** by executing the Python implementation script given in [27]. Their objective is to validate the computed **cTarget** GTK and IGTK against that computed and displayed by Wireshark, that is, **wTarget**.

Exercise 3: Decryption of broadcast/multicast packet using WPA GTK
The GTK is used for the decryption of broadcast and multicast messages. Note that the GTK is the same for all STAs on a given wireless network. In this exercise, the students are to identify the payload of an ARP or DHCP broadcast message/packet captured and decrypted (**wTarget**) by Wireshark using the TK (extracted from the logs). They are to review the AES-128 MODE CCM Python implementation script given in Appendix B.1, identify the raw, and compute the input ingredients (see Exercise 1). Once they extract the GTK from the logs, they are to compute the **cTarget** value by executing the Python script. Their objective is to validate the computed **cTarget** broadcast plaintext against that decrypted and displayed by Wireshark **wTarget**.

5.2.2 Message Integrity Code (MIC)
Unlike WPA2, which uses HMAC SHA1 MIC, WPA3 Security framework uses AES-128 MODE CMAC [28] for message integrity code computation (known also as Message Integrity Code (MIC)) and confirmation.

Exercise 4: WPA3 AES Mode CMAC MIC Verification of the EAPOL-M2 using WPA KCK
In WPA3 (with IEEE 802.11w being mandatory), the MIC is computed using the WPA Key Confirmation Key (KCK) in the case of the EAPOL-M2 unicast frame (see Figure 8).

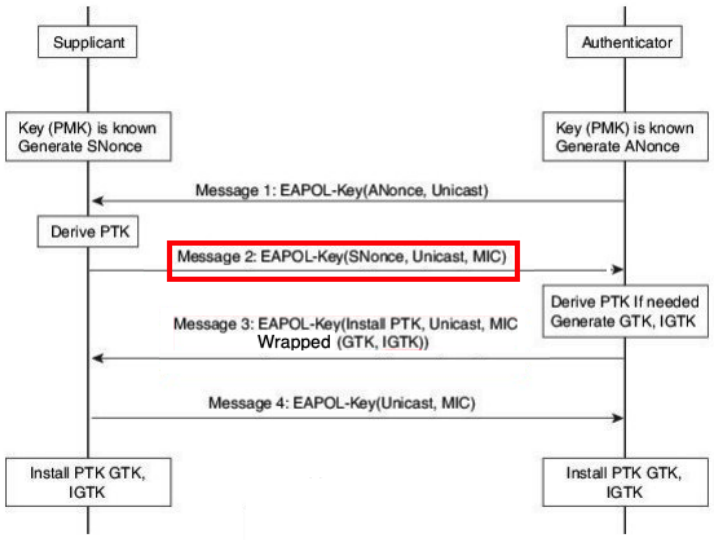

Figure 8: AES Mode CMAC MIC Verification with KCK in WPA3 EAPOL-M2 [26].

The students are to identify the EAPOL-M2 message in a 4-way handshake that is captured by Wireshark. The MIC value of the EAPOL-M2 becomes the **wTarget**. The payload to be used for computing the MIC value should have the MIC string replaced with zeros. The KCK is extracted from the logs. The students are to compute the **cTarget** using the Python script (AES-128 MODE CMAC implementation) provided in B.2.1. Their objective is to validate the computed **cTarget** EAPOL-M2 MIC against that captured and displayed by Wireshark **wTarget**.

Exercise 5: WPA3 Verification of PMF Broadcast Deauthentication Frame using WPA IGTK
In WPA3, broadcast/multicast management frames are protected against forging using a new key known as IGTK. Note that the frames are authenticated but not encrypted with that key. This key is created during the 4-way handshake (see Figure 7). Broadcast/Multicast management frames use a new algorithm known as Broadcast Integrity Protocol (BIP) (see Section 12.5.4 in [10] and Figure 10) along with new Information Element (IE): Management MIC IE (see Figure 9) with Sequence Number plus Cryptographic Hash (based on AES-128 MODE CMAC). Note that IGTK (along with GTK) are the keys that are unwrapped in Exercise 2. The BIP provides replay protection through a parameter known as an IGTK Packet Number (IPN). When the AP sends a protected broadcast/multicast frame to the STA/client, it puts an IPN in the Management MIC Element (MMIE) field. The STA/client compares the received IPN with the local IPN. If the received IPN is not larger than the local IPN, the STA/client discards the frame.

The students are to select a broadcast deauthentication management frame from a WPA3 Wireshark capture and identify the MIC value as the **wTarget**. They are to review the Python script given in Appendix B.2.2, identify the raw ingredients (such as Managament Frame Body (MFB), MMIE), and compute the input ingredient ADD (see Figure 11). Once they extract the WPA IGTK key from the logs, they are to compute the **cTarget** using the Python script. Their objective is to validate the computed **cTarget** BIP MIC against that captured and displayed by Wireshark **wTarget**.

	Element ID	Length	Key ID	IPN/BIPN	MIC
Octets:	1	1	2	6	8 or 16

Figure 9: Managament MIC Element (MMIE) Format for Management Frames in WPA3 [10].

IEEE 802.11 Header	Management Frame Body including MME	FCS

Figure 10: BIP Encapsulation of Management Frames in WPA3 [10].

	FC	A1	A2	A3
Octets:	2	6	6	6

Figure 11: BIP AAD Construction for MMIE in WPA3 [10].

5.2.3 WPA3 Authentication

In WPA2, the initial authentication phase is based on what is known as the Open System Authentication. It's vulnerable to a number of serious attacks. WPA3 uses the Simultaneous Authentication of Equals (SAE) handshake which is essentially a Password Authenticated Key Exchange (PAKE) scheme. PAKE addresses a practical security, that is, how two parties can, with a shared secret, establish secure, authenticated communication without relying on a Public Key Infrastructure (PKI) [29], [30].

SAE is based on discrete logarithm cryptography to achieve authentication and key agreement. Each party of an exchange derives ephemeral public and private keys with respect to a particular set of domain parameters that define a finite cyclic group. Groups may be based on either finite field cryptography (FFC) [31] or on elliptic curve cryptography (ECC) [32] (see Section 12.4 in [10], [33]). This algorithm results in a different Pairwise Master Key (PMK) for each STA/client and for each session. This is a significant security improvement that mitigates, for example, WPA2 dictionary-based and KRACK attacks [34].

Exercise 6: WPA3/SAE (standalone)

In this exercise, the students are to make use of a Python implementation script of the SAE algorithm which does not fully adhere to the IEEE 802.11-2020 standards [10]. However, it demonstrates the basic principles and foundation of the algorithm. The objective of this exercise is to demonstrate that the PMK value created by the SAE algorithm is always different from one run/-connection to the next between a STA/wireless client and an access point. The students are to download the following Python script: dragonfly_implementation.py from [35]. They are to review and execute the script to verify that with the same passphrase π shared among all STAs and AP, the algorithm enables a STA and the AP to generate independently the same shared secret and PMK values. However, these values are different from one run to the next on a given STA and are different on different STAs. These are significant improvements over the WPA2 Pre-Shared Key (PSK) authentication approach.

Exercise 7: WPA3/SAE Confirm Verification

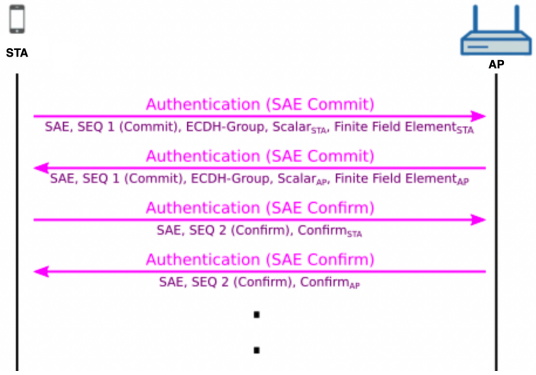

Figure 12: WPA3 SAE/Dragonfly Authentication Messages [9].

All four authentication frames in Figures 12 and 13 for WPA3-Personal contain important informa-

tion to calculate the shared secret κ and PMK based on Elliptic Curve Cryptography (ECC). The STA/wireless client and the AP need to know the calculated scalar and point (Finite Field Element (FFE)) of the other party [36] for verification. This information is shared in the first two authentication frames: the first SAE Commit frame includes the ECDH Group ID, scalar and the new FFE point on the elliptic curve from STA to AP and the second SAE Commit message contains those sent from the AP to STA.

The verification, in the form of computing a confirm token by each party (see 12.4.5.5-12.4.5.6 in [10], [37]), depends on the use of the data mentioned above in addition to the status code (2 bytes) provided in the second two (2) SAE Confirm authentication frames. The confirm token consists of a HMAC-SHA256 digest created with the SAE KCK key. Note that the SAE KCK key is not the same as the WPA KCK. Usually κ, the shared secret, is stretched into two subkeys for cryptographic hygiene. One subkey is used as the SAE KCK for the computation of the confirm token, the other subkey PMK is used as input to the 4-way handshake [30] and key derivation and confirmation.

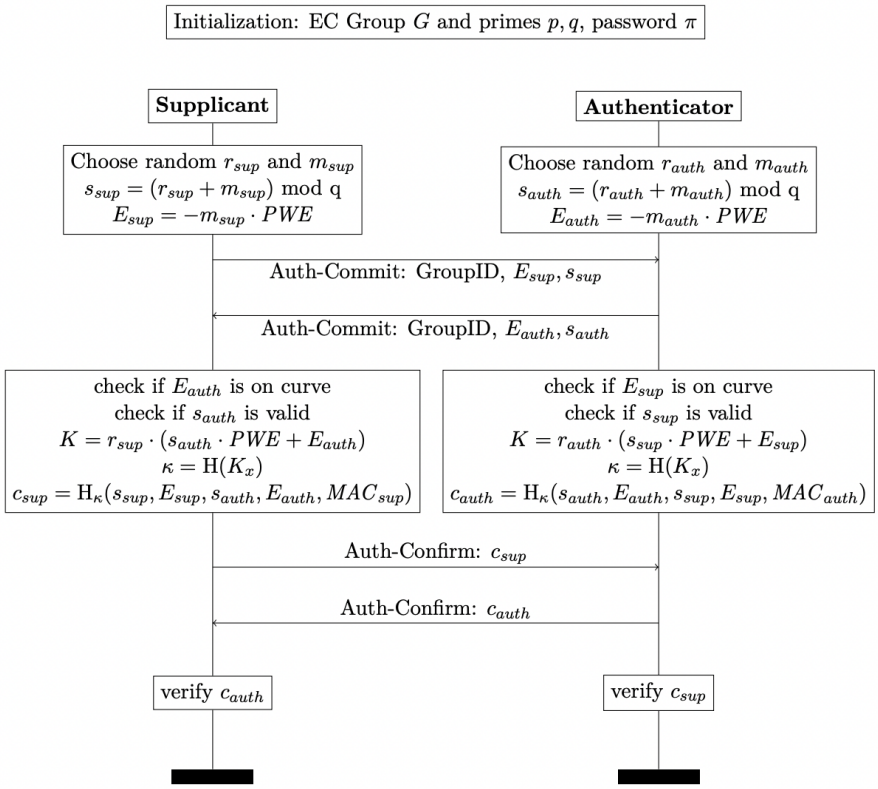

Figure 13: High-level overview of the SAE commit and confirm handshake. In this figure, the variable names scalar, Element and mask are abbreviated as s, E and m. K is the shared secret. Hκ is a cryptographic hash function with key κ and is usually implemented as SHA256-HMAC. Kx is the x-coordinate of the elliptic curve point K [30].

In the confirm exchange (see Figure 13), both parties verify that they derived the same shared secret κ and PMK and therefore possess the same passphrase π. In the beginning of the Dragonfly

key exchange, the shared passphrase π is chosen at random and given to the STA/supplicant and AP in a secure out-of-bounds process. Both the STA/supplicant and AP exchange their token and verify the other token. This verification is possible because all necessary elements were exchanged previously and are known to all participants. If the verification succeeds, the handshake completes and the subkey of the shared secretκ is used as the PMK. Otherwise, the handshake times out and the authentication fails.

The students are to select the 4 SAE authentication messages captured by Wireshark and identify the two (2) confirm (token verification) values exchanged between the STA and AP. These are considered the **wTarget**. They are to review the Python implementation script (given in Appendix B.3) and identify the raw ingredients: status code (sc), STA own_scalar, STA own_element, AP peer_scalar, and AP peer_element. Using the raw ingredients, they are to prepare the input ingredients: STA own_message and AP peer_message. Once they extract the SAE KCK from the logs, they are to compute the **cTarget**, that is, the two Confirm codes using the Python script in Appendix B.3. Their objective is to validate the computed **cTarget** token verification codes against those captured and displayed by Wireshark **wTarget**.

5.2.4 Key Derivation

Once the STA and AP have successfully authenticated each other, they move into the association phase followed by the 4-way handshake for key derivation and confirmation as shown in Figure 14. The pairwise key hierarchy takes a PMK and generates a PTK. The PTK is partitioned into WPA KCK, WPA KEK, and WPA TK. The TK is used by the STA (wireless client) and the AP (access point) to protect individually addressed communication between the two parties. It's important to note that key derivation function for WPA3 and WPA2 configured for PMF is based on HMAC-SHA256. This function is different than that specified for WPA2 which is based on SHA1 as specified in section 12.7.1.2 titled PRF in in IEEE 80211-2020 [10].

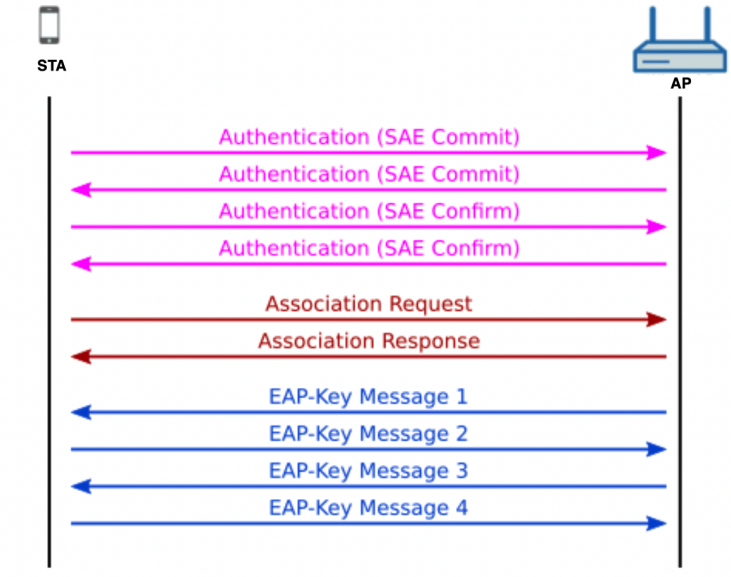

Figure 14: WPA3 Connection Flow Chart [9].

Exercise 8: WPA3 Key Management using HMAC-SHA256
In the case of the WPA3-Personal, the PMK is generated during the SAE authentication phase and is computed independedly by each of the AP and the STA/client. The SAE PMK is unique for each STA/wireless client and each session related to that STA. During the 4-handshake phase, the PMK is used for key derivation using a function known as Key derivation function (KDF). It is specified in section 12.7.1.6.2 titled Key Derivation Function (KDF)in IEEE 80211-2020 [10].

The students are to extract the WPA PTK, WPA TK, WPA KCK, and WPA KEK from the debug message logs. These are considered the **wTarget**. They are to review the Python implementation script given in Appendix B.4, identify the raw ingredients: AP_addr, STA_addr, A_nonce, and S_-nonce from the eapol messages captured in EAPOL 4-way handshake by Wireshark and compute the input ingredients A and B according to Appendix B.4. Once they extract the SAE PMK from the debug message logs, they are to compute the **cTarget** using the Python script. Their objective is to validate the computed **cTarget** derived keys against that logged in the debug message logs, captured and/or displayed by Wireshark **wTarget**.

For completeness, the students are directed to unwrap the WPA GTK, and WPA IGTK in the EAPOL M3 using the procedure presented in Exercise 2.

Exercise 9: WPA3 GTK/IGTK rekeying with 2-way Handshake using WPA TK and KEK
The 2-way handshake consists of EAPOL Group Message 1 of 2 and 2 of 2. We focus here on the Group Message 1 of 2. The message is AES-128 CCMP encrypted and therefore is not displayed by Wireshark. However, the message can be decrypted by Wireshark once the WPA TK is installed into Wireshark. The decrypted payload of the message carries wrapped updated (rekeyed) GTK and IGTK keys (WPA Key Data).

The students are to decrypt the Group Message 1 (using the procedure described in Exercise 1 and unwrap the decrypted payload of the same to extract the GTK and IGTK rekeyed keys using the procedure described in Exercise 2.

5.3 WPA3-Enterprise Wireless Security Algorithms

In this section, we provide a brief account of the WPA3-Enterprise security algorithms and the potential lab exercises based on the lab exercises described in Section 5.2. According to the Wi-Fi Alliance [38], there are two WPA3-Enterprise flavors: WPA3-Enterprise and WPA3-Enterprise with 192-bit mode.

WPA3-Enterprise builds upon the foundation of WPA2-Enterprise with the additional requirement of using Protected Management Frames on all WPA3 connections. Key derivation and confirmation is based on minimum 256-bit Hashed Message Authentication Mode (HMAC) with Secure Hash Algorithm (HMAC-SHA256). For Management frame protection, WPA3 supports minimum 128-bit Broadcast/Multicast Integrity Protocol Cipher-based Message Authentication Code (BIP-CMAC-128). The validation of these algorithms are performed according to the lab exercises in Section 5.2 excluding Exercises 6 and 7 which are not applicable to WPA3-Enterprise mode.

WPA3-Enterprise with 192-bit mode offers an optional mode using 192-bit minimum-strength security protocols and cryptographic tools to better protect sensitive data. This mode includes:

(1) Authentication: Extensible Authentication Protocol – Transport Layer Security (EAP-TLS) using Elliptic Curve Diffie-Hellman (ECDH) exchange and Elliptic Curve Digital Signature Algorithm (ECDSA) using a 384-bit elliptic curve

(2) Authenticated encryption: 256-bit Galois/Counter Mode Protocol (GCMP-256). The algorithm validation is performed according to Exercise 1 procedure with the exception of replacing AES.MODE.CCM with AES.MODE_GCM.

(3) Key derivation and confirmation: 384-bit Hashed Message Authentication Mode (HMAC) with Secure Hash Algorithm (HMAC-SHA384). The key confirmation algorithm validation is performed according to Exercises 4 and 8 procedures with the exception of replacing SHA256 with SHA384.

(4) Robust management frame protection: 256-bit Broadcast/Multicast Integrity Protocol Galois Message Authentication Code (BIP-GMAC-256). The algorithm validation is performed according to Exercise 5 procedure with the exception of replacing BIP-CMAC-128 with BIP-GMAC-256.

6 Discussion, Conclusion & Next Steps

6.1 Discussion

The original material of the lab exercises was developed by the instructor and tested by the students in the Fall 2022 semester. Section 5 of this paper represents the updated version of the lab exercises taking into account the students' observations and suggestions on the original material. Given below are the observations and suggestions offered by our students who tested the original version of the lab exercises.

Observations: Our students found (1) the original approach of having to provide them with the Python implementation scripts for some of the algorithms but not for all was very challenging, (2) the original algorithm validation template was difficult to follow and sometimes unclear leading to longer period of troubleshooting, (3) it was quite difficult to find much useful information about the subject matter on the Internet, aside from some conceptual overviews that didn't provide helpful answers, and (4) that some of the provided references were not that helpful, meaning it didn't help them in the development of neither the Python scripts nor the computation of the **cTarget**.

Suggestions: Our students suggested that it would be helpful (1) to provide the Python implementation scripts of the security algorithms that they needed to use rather than having to find some online that may or may not be reliable, (2) to provide some extra readings as well as the Python scripts to make these lab exercises less challenging, and (3) to adjust the objectives of the lab exercises from being the implementation of the security algorithms to being a vehicle to gain a deeper understanding of the same.

6.2 Conclusion & Next Steps

In conclusion, we were successfully able to design and deliver a customizable and cost-effective WPA3 wireless network environment. This enabled us to design, develop and deliver a complete

set of lab exercises on WPA3-Personal security algorithms. Our students offered valuable observations and suggestions at the end of testing these lab exercises. Their observations highlighted the lack of effective learning materials on WPA3 security algorithms, which was the initial motivation of the development of the lab exercises presented in this paper. The lab exercise material in this paper is the updated version taking into account our students' feedback.

Although the lab exercise material presented here met its initial objectives, we believe it will benefit from the following future enhancements:

(1) Expand the scope to include in-depth lab exercises for WPA3-Enterprise mode.

(2) Provide in-depth characterization of the weakness and strength of the various WPA3 security algorithms.

(3) Enhance the curriculum to include WPA3 potential vulnerability lab exercises [6].

Acknowledgment

The author would like to thank the College of Science and Engineering (CISE) and Computer Science Department at James Madison University (JMU) for their support throughout the development of the lab exercises and the creation of this paper. Special thanks are to the students in ISAT 465 class of the Fall of 2022. Also, the author thanks Mr. Paul Henriksen for his proofreading of the draft version of the paper.

References

[1] E. J. Glantz, M. R. Bartolacci, M. Nasereddin, D. J. Fusco, J. C. Peca, and D. Kachmar, "Wireless cybersecurity education: A focus on curriculum," in *2021 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, 2021. Retrieved: 2023-03-31.

[2] L. Wang, J. Yang, and P.-J. Wan, "Educational modules and research surveys on critical cybersecurity topics," *International Journal of Distributed Sensor Networks*, vol. 16, p. 155014772095467, 09 2020. Retrieved: 2023-03-31.

[3] "NDG Where Practice Leads to Success." https://www.netdevgroup.com/index.html, Retrieved: 2023-04-02.

[4] "PRANEETH'S BLOG." https://praneethwifi.in/, Retrieved: 2023-04-03.

[5] Bill Buchanan OBE - , "ASecuritySite.com." https://asecuritysite.com/, Retrieved: 2023-04-06.

[6] E. Chatzoglou, G. Kambourakis, and C. Kolias, "How is your wi-fi connection today? dos attacks on wpa3-sae (article)," in *Journal of Information Security and Applications, https://www.sciencedirect.com/science/article/pii/S221421262100243X*, vol. 64, p. 103058, 2022. Retrieved: 2023-03-31.

[7] N. Dalal, N. Akhtar, A. Gupta, N. Karamchandani, G. S. Kasbekar, and J. Parekh, "A wireless intrusion detection system for 802.11 wpa3 networks," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 384–392, 2022.

[8] "WPA3 Specifications." https://www.wi-fi.org/download.php?file=/sites/default/files/private/WPA3%20Specification%20v3.1.pdf, Retrieved: 2023-02-22.

[9] Morti, "WPA3 – Improving your WLAN security." https://wlan1nde.wordpress.com/2018/09/14/wpa3-improving-your-wlan-security/, Retrieved: 2023-02-24.

[10] "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 - Redline," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) - Redline*, pp. 1–7524, 2020. Retrieved: 2023-02-20.

[11] "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," *IEEE Std 802.11w-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, and IEEE Std 802.11y-2008)*, pp. 1–111, 2009. Retrieved: 2023-02-20.

[12] "Wi-Fi Alliance." https://www.wi-fi.org/discover-wi-fi/security, Retrieved: 2023-02-22.

[13] "Raspberry Pi OS." https://www.raspberrypi.com/software/operating-systems/#raspberry-pi-os-32-bit, Retrieved: 2023-02-13.

[14] "TP-LINK TL-WN722N v1.x." http://en.techinfodepot.shoutwiki.com/wiki/TP-LINK_TL-WN722N_v1.x, Retrieved: 2023-02-13.

[15] "Linux WPA2/WPA3/IEEE 802.1X Supplicant." https://w1.fi/wpa_supplicant/, Retrieved: 2023-02-22.

[16] "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA2/WPA3/EAP/RADIUS Authenticator." https://w1.fi/hostapd/, Retrieved: 2023-02-22.

[17] "freeradius." https://freeradius.org/, Retrieved: 2023-02-22.

[18] "TP-Link Archer A7 v5." https://openwrt.org/toh/tp-link/archer_a7_v5, Retrieved: 2023-02-13.

[19] "Archer A7 AC1750 Wireless Dual Band Gigabit Router." https://www.tp-link.com/us/home-networking/wifi-router/archer-a7/, Retrieved: 2023-02-22.

[20] "How to unbrick TP Link Archer A7 v5 ." https://milankragujevic.com/restore-a-bricked-tp-link-router-with-tftp, Retrieved: 2023-02-13.

[21] "Raspberry Pi Automated WiFi Access Point." https://github.com/arm358/Raspberry-Pi-Automated-WiFi-Access, Retrieved: 2023-02-13.

[22] "How to capture 802.11 packets using Mac OS." https://community.cambiumnetworks.com/t/how-to-capture-802-11-packets-using-mac-os/78642, Retrieved: 2023-02-22.

[23] Kemparaj Praneeth, "Understanding CTR with CBC-MAC Protocol (CCMP) AES-CCMP in depth." https://praneethwifi.in/2020/05/02/ctr-with-cbc-mac-protocol-ccmp-aes-ccmp/, Retrieved: 2023-02-24.

[24] "RFC 3394 - Advanced Encryption Standard (AES) Key Wrap Algorithm." https://www.rfc-editor.org/rfc/rfc3394, Retrieved: 2023-02-24.

[25] Kurt Rose, "Key Wrapping." https://github.com/kurtbrose/aes_keywrap/blob/master/aes_keywrap.py, Retrieved: 2023-02-20.

[26] Vivekananda Holla, "Management Frame Protection – IGTK." http://www.hitchhikersguidetolearning.com/2017/09/17/management-frame-protection-igtk/, Retrieved: 2023-02-24.

[27] Bill Buchanan OBE - ASecuritySite.com, "Key Wrapping." https://asecuritysite.com/encryption/kek, Retrieved: 2023-02-24.

[28] "RFC 4493 - The AES-CMAC Algorithm." https://www.rfc-editor.org/rfc/rfc4493, Retrieved: 2023-02-24.

[29] Feng Hao and Peter Ryan, ""J-PAKE: authenticated key exchange without PKI"," *Transactions on computational science XI. Springer, 2010*, p. 192–206, 2010. Retrieved: 2023-02-24.

[30] Nikolai Tschacher, "Model Based fuzzing of the WPA3 Dragonfly Handshake," Master's thesis, Institute for Computer Science, Humboldt University, Berlin, Germany, 2019. Retrieved: 2023-02-24.

[31] "Finite field Element (FFE)." https://en.wikipedia.org/wiki/Finite_field, Retrieved: 2023-02-22.

[32] "Elliptic Curve Cryptography (ECC)." https://cryptobook.nakov.com/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc, Retrieved: 2023-02-22.

[33] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the dragonfly handshake of wpa3 and eap-pwd," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 517–533, 2020. Retrieved: 2023-02-20.

[34] 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)*, 2017. Retrieved: 2023-02-20.

[35] "Dragonfly-SAE-Implementation - NikolaiT." https://github.com/NikolaiT/Dragonfly-SAE/blob/master/dragonfly_implementation.py, Retrieved: 2023-02-22.

[36] D. Harkins, Ed., "Dragonfly Key Exchange." https://www.rfc-editor.org/rfc/rfc7664, 2015. Retrieved: 2023-02-22.

[37] rdkcteam, "wpa_supplicant-2.9/src/common/sae.c)." https://github.com/rdkcteam/wpa_supplicant-2.9/blob/master/src/common/sae.c, Retrieved: 2023-02-22.

[38] "Discover Wi-Fi - Security." https://www.wi-fi.org/discover-wi-fi/security/, Retrieved: 2023-02-24.

Appendix A Configurations for Section 4

A.1 Personal Mode

A.1.1 Personal Mode - ap-rasp - hostapd.conf

```
interface =wlan1
ssid=”<ssid>”
hw_mode=g
ieee80211n=1
channel=2
macaddr_acl=0
auth_algs =1
ignore_broadcast_ssid =0
wpa=2
wpa_passphrase=”<psk>”
wpa_key_mgmt=SAE
rsn_pairwise =CCMP
group_cipher=CCMP
wpa_group_rekey=120
ieee80211w=2
```

A.1.2 Personal Mode - wpa-rasp - wpa_supplicant.conf

```
network={
    ssid=”<ssid>”
    scan_ssid =1
    psk=”<psk>”
    key_mgmt=SAE
    proto=RSN
    group=CCMP
    pairwise=CCMP
    ieee80211w=2
}
```

A.2 Enterprise Mode

A.2.1 Enterprise Mode - radius-rasp - eapol-ttls.conf

```
network={
    eap=TTLS
    eapol_flags =0
    key_mgmt=WPA-EAP
    identity =”<supplicant_username?”
    password=”<supplicant_password>”
    ca_cert =”<path_to>/ca.pem”
    phase2=”autheap=MSCHAPV2”
}
```

A.2.2 Enterprise Mode - wpa-rasp -wpa_supplicant.conf

```
network={
    ssid=”<SSID>”
    scan_ssid =1
```

```
    proto=RSN
    pairwise=GCMP-256
    group=GCMP-256
    group_mgmt=BIP-GMAC-256
    ieee80211w=2
    eap=TTLS
    eapol_flags =0
    key_mgmt=WPA-EAP-SUITE-B-192
    identity ="<supplicant_username>"
    password="<supplicant_password>"
    ca_cert ="<path_to>/ca.pem"
    phase2="autheap=MSCHAPV2"
}
```

Appendix B Code Snippets for Section 5

B.1 AES-128 CCM Decryption of Ciphertext

```
#!/usr/bin/python3

from scapy.all import *
from Crypto.Cipher import AES

AAD = bytes.fromhex(<muted FC in hex>) + bytes.fromhex(<A1 in hex>) + bytes.fromhex(<A2 in hex>) +
bytes.fromhex(<A3 in hex>) + bytes.fromhex(<SC in hex>) + bytes.fromhex(<QC in hex>)
nonce = bytes.fromhex(<priority in hex>) + bytes.fromhex(<A2 in hex>) + bytes.fromhex(<PN in hex>)
key = bytes.fromhex(<TK in hex>)
ciphertext = bytes.fromhex(<data in hex>)

cipher = AES.new(key, AES.MODE_CCM, nonce, mac_len=8)
cipher.update(AAD)
msg = nonce, AAD, cipher.decrypt( ciphertext )

hexdump(msg[2]) # plaintext + MIC
```

B.2 Message Integrity Code (MIC)

B.2.1 WPA3 AES MODE CMAC MIC Verification

```
#!/usr/bin/python3

from Crypto.Cipher import AES
from Crypto.Hash import CMAC
kck= bytes.fromhex(<kck in hex>)

cobj = CMAC.new(kck, ciphermod=AES)
data=bytes.fromhex(<data in hex>) # replace the MIC with all zeros
cobj.update(data)

print(cobj.hexdigest())
```

B.2.2 PMF for Broadcast/Multicast Deauthentication in WPA3

```
#!/usr/bin/python3

from Crypto.Cipher import AES
from Crypto.Hash import CMAC

AAD=bytes.fromhex(<muted FC in hex)+ bytes.fromhex(<A1 in hex>)+bytes.fromhex(<A2 in hex>)+bytes.
    fromhex(<A3 in hex>)
MFB=bytes.fromhex(<reason code in hex>)
MMIE=bytes.fromhex(<MMIE in hex>) # with MIC replaced with zeros
secret = bytes.fromhex(<IGTK in hex>)

cobj = CMAC.new(secret, ciphermod=AES,mac_len=8)
data = AAD+MFB+MMIE
cobj.update(data)

print(cobj.hexdigest())
```

B.3 Authentication - WPA3/SAE Confirm Verification

```
#!/usr/bin/env python3

import hashlib
import hmac

kck=<kck in hex>
own_scalar = <own_scalar in hex>
own_element= <own_element in hex>
peer_scalar = <peer_scalar in hex>
peer_element= <peer_element in hex>
sc ="0000" # status code

own_message = bytes.fromhex(sc)+bytes.fromhex( peer_scalar )+bytes.fromhex(peer_element)+bytes.fromhex(
    own_scalar)+bytes.fromhex(own_element)
H1=hmac.new(bytes.fromhex(kck),own_message,hashlib.sha256)
token1 = H1.hexdigest()

print(token1)

peer_message = bytes.fromhex(sc)+bytes.fromhex(own_scalar)+bytes.fromhex(own_element)+bytes.fromhex(
    peer_scalar )+bytes.fromhex(peer_element)
H2=hmac.new(bytes.fromhex(kck),peer_message,hashlib.sha256)
token2 = H2.hexdigest()

print(token2)
```

B.4 WPA3 Key Derivation (WPA2-PSK-SHA256)

```
#!/usr/bin/python3

import hashlib
import hmac

```

```
6  def custom_prf(key, a, b, l):
7      i = 1
8      r = b''
9      while i <= int((l+256) / 256):
10         hmacsha256 = hmac.new(key, i.to_bytes(2,'little') + a + b + l.to_bytes(2,'little'), hashlib.
               sha256)
11         r += hmacsha256.digest()
12         i += 1
13     return r
14
15 Label=bytes("Pairwise key expansion",'utf-8')
16 MAC_AP= bytes.fromhex(<hex>)
17 MAC_CLIENT=bytes.fromhex(<hex>)
18 PMK=bytes.fromhex(<hex>)
19 NONCE_AP=bytes.fromhex(<hex>)
20 NONCE_CLIENT=bytes.fromhex(<hex>)
21 Context = min(MAC_AP,MAC_CLIENT)+max(MAC_AP,MAC_CLIENT)+min(NONCE_AP,NONCE_CLIENT)+
       max(NONCE_AP,NONCE_CLIENT)
22
23 PTK=custom_prf(PMK, Label, Context, 384).hex() [0:96]
24
25 print("PTK:",PTK)
26 print("KCK:",PTK[0:32])
27 print("KEK:",PTK[32:64])
28 print("TK :",PTK[64:96])
```