

Aalto University
School of Science
Degree Programme in Security and Mobile Computing

Shiva Prasad Thagadur Prakash

Enhancements to Secure Bootstrapping of Smart Appliances

Master's Thesis
Espoo, July 31, 2017

Supervisors: Prof. Tuomas Aura, Aalto University
Prof. Danilo Gligoroski, NTNU
Advisor: Mohit Sethi, D.Sc. (Tech.)

Aalto University
 School of Science
 Degree Programme in Security and Mobile Computing

ABSTRACT OF
 MASTER'S THESIS

Author:	Shiva Prasad Thagadur Prakash		
Title:	Enhancements to Secure Bootstrapping of Smart Appliances		
Date:	July 31, 2017	Pages:	70
Major:	Security and Mobile Computing	Code:	T-3011
Supervisors:	Prof. Tuomas Aura, Aalto University Prof. Danilo Gligoroski, NTNU		
Advisor:	Mohit Sethi, D.Sc. (Tech.)		
<p>In recent times, there has been a proliferation of smart IoT devices that make our everyday life more convenient, both at home and at work environment. Most of these smart devices are connected to cloud-based online services, and they typically reuse the existing Wi-Fi network infrastructure for Internet connectivity. Hence, it is of paramount importance to ensure that these devices establish a robust security association with the Wi-Fi networks and cloud-based servers. The initial process by which a device establishes a robust security association with the network and servers is known as <i>secure bootstrapping</i>. The bootstrapping process results in the derivation of security keys and other connection parameters required by the security associations. Since the smart IoT devices often possess minimal user-interface, there is a need for bootstrapping methods with which the users can effortlessly connect their smart IoT devices to the networks and services. Nimble out-of-band authentication for Extensible Authentication Protocol (EAP-NOOB) is one such secure bootstrapping method. It is a new EAP authentication method for IEEE 802.1X/EAP authentication framework. The protocol does not assume or require any pre-configured authentication credentials such as symmetric keys or certificates. In lieu, the authentication credentials along with the user’s ownership of the device are established during the bootstrapping process.</p> <p>The primary goal of this thesis is to study and implement the draft specification of the EAP-NOOB protocol in order to evaluate the working of EAP-NOOB in real-world scenarios. During our implementation and testing of the initial prototype for EAP-NOOB, we discovered several issues in the protocol. In this thesis, we propose a suitable solution for each of the problems identified and also, verify the solutions through implementation and testing. The main results of this thesis work are various enhancements and clarifications to the EAP-NOOB protocol specification. The results consequently aid the standardisation of the protocol at IETF. We also design and implement several additional features for EAP-NOOB to enhance the user experience.</p>			
Keywords:	EAP, EAP-NOOB, Secure Bootstrapping, IoT, Out-of-Band Authentication		
Language:	English		

Acknowledgements

This thesis work was carried at Aalto University, Finland under the supervision of Professor Tuomas Aura (Aalto University) and Professor Danilo Gligoroski (Norwegian University of Science and Technology) as part of the Erasmus Mundus NordSecMob program.

First and foremost, I would like to sincerely thank Professor Tuomas Aura for his constant feedback and my sincere gratitude for taking the team to IETF-99, Prague and IETF-96, Berlin. I would like to sincerely thank Professor Danilo Gligoroski for his remote support and co-supervision.

I would also like to thank Dr. Mohit Sethi for all his timely advice and his continuous support.

I thank my teammate Raghavendra Mudugodu Seetarama for his efforts and contribution towards the project. Working with him had been an enjoyable experience.

Finally, I would like to express my sincere gratitude to my family and friends for their encouragement and support.

Espoo, July 31, 2017

Shiva Prasad Thagadur Prakash

Abbreviations and Acronyms

AAA	Authentication Authorization and Accounting
AES	Advance Encryption System
AMSK	Application Master Session Key
AP	Access Point
API	Application Programme Interface
CCMP	Counter-Mode-CBC-MAC
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EAP	Extensible Authentication Protocol
EAPOL	EAP over LAN
ECC	Elliptic Curve Cryptosystem
ECDH	Elliptic Curve Diffie-Hellman
EMSK	Extended Master Session Key
ESS	Extended Service Set
ESSID	Extended Service Set Identifier.
FQDN	Fully Qualified Domain Name.
IEEE	Institute of Electrical and Electronics Engineers
IETF	International Engineering Task Force
IOT	Internet of Things
IP	Internet Protocol
KDF	Key Derivation Function
LAN	Local Area Network
MAC	Message Authentication Code
MITM	Man In The Middle
MSK	Master Session Key
NAI	Network Access Identifier
NAS	Network Access Server
NFC	Near Field Communication
NIST	National Institute of Standards and Technology

NOOB	Nimble Out-Of-Band
OOB	Out-Of-Band
PAE	Port Access Entity
PE	Protocol Entity
PMK	Pairwise Master Key
PSK	Pre-Shared Key
PTK	Pairwise Transient Key
QR	Quick Response
RADIUS	Remote Authentication Dial-In User Service
RFC	Request For Comment
RSN	Robust Security Network
RSNA	Robust Security Network Association
SSID	Service Set Identifier
STA	Station
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TTLS	Tunneled Transport Layer Security
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2
WPS	Wi-Fi Protected Setup

Contents

Abbreviations and Acronyms	iv
1 Introduction	1
1.1 Problem Statement	4
1.2 Structure of the Thesis	4
2 Background	6
2.1 Lifecycle of an IoT Device	6
2.2 Objectives of Network Security	7
2.3 Secure Bootstrapping	8
2.4 Security Protocols for Wi-Fi Networks	10
2.5 WPA2 and RSN	11
2.5.1 IEEE 802.1X	13
2.5.2 EAP	15
2.5.3 EAPOL	16
2.5.4 RADIUS	17
2.5.5 EAP Authentication Process	19
2.5.6 RSNA Establishment Procedure	20
2.6 ECDH Key Exchange Procedure	24
2.7 OOB Channels for Authentication	25
3 Protocol Overview	27
3.1 A Brief Overview of the Protocol	27
3.2 Motivation for EAP-NOOB	31
3.3 A Brief Overview of the Initial Implementation	31
4 Access Control to Network Resources	34
4.1 Need for Fine-Grained Access Control	34
4.2 Proposed Solution	36
4.3 Implementation Environment	37
4.4 Access Control Based on Called-Station-ID and User-Name . .	38

4.5	Access Control Based on NAS-ID and User-Name	39
4.6	Access Control Implementation for EAP-NOOB	42
4.7	Isolation of Network Clients	43
5	Enhancements to EAP-NOOB	45
5.1	Re-keying and Algorithm Agility	45
5.1.1	No Change in the Supported Cryptosuite	46
5.1.2	New Cryptosuite Negotiated	47
5.2	Timeouts	48
5.3	Failure Recovery	49
5.4	Handling Multiple Sessions	50
5.5	Roaming	50
6	Additional Features	52
6.1	IEEE 802.1X Wired Access	52
6.2	OOB Channel with NFC	56
6.3	Web User-Interface to Configure the RADIUS Clients	58
7	Discussion	60
7.1	Thesis Contributions	60
7.2	Open Questions	62
7.3	Standards and Implementation Status	62
8	Conclusion	64

List of Figures

2.1	Lifecycle of a device [38]	7
2.2	Relationship of Wi-Fi to IEEE 802.11	11
2.3	802.1x architecture [15]	14
2.4	WPA2-Enterprise network architecture	18
2.5	RSN key hierarchy	22
2.6	RSNA establishment procedure(adapted and modified [34])	23
3.1	EAP-NOOB server-peer association state machine [51]	28
3.2	EAP-NOOB key hierarchy	30
3.3	EAP-NOOB setup	33
3.4	An example scenario depicting the relationship between various components of the EAP-NOOB protocol	33
4.1	Access control problem in home AP	35
4.2	Access control problem in enterprise environments	37
4.3	Policy file 1 based on Called-Station-ID	39
4.4	Policy file 2 based on Called-Station-ID	39
4.5	Access control based on NAS-ID attribute	40
4.6	Policy file based on NAS-ID	41
5.1	Reconnect Exchange	46
5.2	New message sequence for the Completion Exchange [51]	51
6.1	<i>wpa_suppliant</i> architecture [14]	53
6.2	802.1X-based wired access	55
6.3	OOB message transfer with NFC	57
6.4	Web UI to configure RADIUS clients	58

Chapter 1

Introduction

The Internet of Things (IoT) is a concept of an interconnected world where almost every physical object, as long as they are equipped with the necessary electronic parts, can be seamlessly connected to the Internet. IoT has evolved from the convergence of the Internet, embedded systems, wireless technology and data science. It has enabled insights and data never available before, culminating in improved decision-making and actions in the physical world. For instance, consider the problem of road traffic management. It may be inefficient to employ pre-programmed traffic signal lights for controlling the road traffic as the traffic intensity may significantly vary throughout a day. Deploying humans for this purpose could be costly and also, infeasible during adverse weather conditions. In lieu, through IoT solutions [35], the traffic can be efficiently managed by dynamically operating the traffic signal lights over the Internet.

Smart IoT devices are on the path to form a ubiquitous network around us. Examples of such smart objects include not only smartphones or tablets but also objects of everyday life such as washing machines, dishwashers, doors and so on. These smart IoT devices have made our daily lives more convenient and have begun to transform our way of living. Therefore, it is captivating for the researchers all over to address the issues that might impede the adoption of IoT in our everyday life.

In all modern communication systems, security is an important consideration. It is crucial to ensure that IoT devices from different manufacturers interoperate as well as connect to the Internet securely. In other words, the devices should communicate securely. Otherwise, they would remain vulnerable throughout the rest of their operational lifespan, attracting adversaries. If an adversary gains control over a vulnerable device, the consequences can be very severe. Since physical objects are connected to the Internet, the implications of device compromise are no longer limited to some monetary

loss but can also be to the extent of causing physical harm to humans. For instance, consider the previously mentioned IoT-based road traffic signals. If an attacker with malicious intent gains unauthorized access to these traffic signals, he can cause traffic accidents. Thus, it is crucial to ensure that IoT devices function securely. The process of making a device function securely when it is first deployed can be termed as secure bootstrapping. After an IoT device starts operating securely, managing it remotely with adequate security is equally important and challenging as well. In this thesis, we will see how secure bootstrapping of IoT devices can be performed so that they securely function and can be managed remotely with security.

Wi-Fi networks are one of the most convenient and widely deployed means to connect to the Internet. In simple words, access to the Internet is widely obtained via Wi-Fi access points. Secure bootstrapping in Wi-Fi networks can be defined as the process of establishing a security relation between an unauthenticated client and a Wi-Fi access point. The bootstrapping process also results in the derivation of security keys for both the parties to ensure data confidentiality and integrity between them. Wi-Fi Protected Access 2 (WPA2) is a Wi-Fi security standard defined for securing Wi-Fi networks. WPA2 has two modes of operation: WPA2-Personal and WPA2-Enterprise. In either mode, as part of secure bootstrapping, each client has to be authenticated. The authentication is performed with the help of credentials which are typically distributed before their use. A user will have to provide the authentication credentials to the wireless client at the time of authentication or pre-install them.

WPA2-Personal networks are deployed in homes, or small-scale environments where security is of least consideration and the number of devices to be managed is less. WPA2-Enterprise networks are deployed in scenarios where security is of paramount importance and there are many devices belonging to different users to be managed. In WPA2-Personal, pre-shared keys (PSK) are used for authentication, and the authentication is performed at the access point. All the users or devices in a given WPA2-Personal network use the same PSK. Due to the use of the same PSK, in WPA2-Personal networks, a presence of a single vulnerable device in a given network makes the entire network vulnerable. On the other hand, WPA2-Enterprise has built-in support for multiple authentication methods known as Extensible Authentication Protocol (EAP) methods, and the authentication is performed at a remote centralized authentication server. The WPA2-Enterprise mode offers a better flexibility as all the devices can be authenticated and managed at a central server. The type of credentials used for authentication depends on the EAP method chosen by the authentication server. For instance, in EAP-TLS method, digital certificates are used for authentication, and in

EAP-PSK, pre-shared keys are used. Unlike Personal networks, in Enterprise networks, each user in a given network possess different credentials.

There is an increased proliferation of smart IoT devices, for example in smart homes [54], remote health monitoring [32] and industrial automation. These IoT devices typically reuse the existing Wi-Fi network infrastructure for Internet connectivity. It is of paramount importance to ensure that these physical objects communicate securely. Also, per-device authentication and network isolation are necessary to prevent an entire Wi-Fi network from being compromised due to a single vulnerable device in the network. Therefore, WPA2-Enterprise networks may soon be preferred even in home environments because Enterprise networks offer robust security, and flexibility in managing the devices.

The enterprise and consumer IoT appliances are off-the-shelf devices. They are sold and dispensed without any prior information on ownership or a method to provision credentials. Therefore, the authentication credentials, device ownership, and device registration must all be set up at the time of their deployment. Also, many IoT appliances possess limited user-interface (UI) that could be utilized to configure them. Examples of such IoT devices are display screens, printers, speakers or cameras, which have either an output or input interface. The manual distribution and management of credentials in such scenarios demand both time and money, especially when the devices are many in number, and it is certainly not scalable. It is non-trivial to use any of the existing EAP authentication methods for secure bootstrapping such IoT devices because they need pre-configured credentials for authentication. Hence, there is a need for a user-friendly EAP authentication method to ease secure bootstrapping, so that the devices can be easily authenticated at a central server from where they can later be managed and controlled.

Nimble out-of-band authentication for EAP (EAP-NOOB) is one such EAP method for authentication, cryptographic key agreement, and registration of smart IoT devices that intend to join an existing Wi-Fi Enterprise network. EAP-NOOB was inspired by earlier work on secure bootstrapping of digital signage by Sethi et al. [39]. Unlike other EAP methods currently available, EAP-NOOB does not assume or require any pre-configured authentication credentials such as symmetric keys or certificates. In this method, a Diffie-Hellman (DH) key exchange is performed over an insecure channel, and a user-assisted out-of-band (OOB) channel is employed to authenticate the established key in order to avoid man-in-the-middle (MiM) attacks on the in-band channel. The security is based on the assumption that the OOB message is sent over a secure channel and cannot be modified or spoofed by an attacker. EAP-NOOB is applicable to relatively capable devices that can

input or output dynamically generated messages of tens of bytes in length and not to extremely resource-constrained sensors.

The NOOB protocol for EAP is specified to be a generic protocol applicable to IoT devices of any make. In order to introduce the EAP-NOOB protocol to a large community of users and also to make devices interoperable with the protocol, the protocol has been proposed for standardization. A draft version of the specification for EAP-NOOB protocol can be found at [50].

1.1 Problem Statement

The EAP-NOOB specification is currently being written. To aid the standardization of EAP-NOOB at the Internet Engineering Task Force (IETF), real-world implementation experiences are crucial.

This thesis work was executed as part of the same research group which has specified the EAP-NOOB protocol, and its main contributions are towards the practical implementation part of the research. The primary goal of this thesis is to influence the protocol specification through real-world implementation experiences of the protocol. In other words, we verify and if necessary modify the protocol specifications based on the real-world implementation experiences. As will be evident from this thesis, several issues were found in the specification during our initial prototype implementation. The identified issues needed to be addressed, and the protocol specification has to be improved accordingly. In this thesis, we also design and implement several additional features for EAP-NOOB to enhance the user experience.

1.2 Structure of the Thesis

The rest of this thesis is organized as follows. In chapter 2, we present all the topics and technologies required to understand both the importance of IoT security and the proposed solution to enhance the security of IoT devices. A brief overview of the EAP-NOOB protocol, including a brief overview of the initial prototype implementation, will be presented in chapter 3. In chapter 4, we propose a way of implementing access control policy decisions based on the Network Access Server (NAS). In chapter 5, solutions to each discovered issue of the EAP-NOOB protocol is presented along with the implementation details. Later, we explain the design and implementation of various additional features for EAP-NOOB in Chapter 6. In chapter 7, we discuss the contribution of the thesis and open questions. Finally, chapter 8

provides the concluding remarks.

Chapter 2

Background

This chapter begins by briefly describing the lifecycle of a smart object, followed by the objectives of network security and a detailed explanation of secure bootstrapping of an IoT device. After that, the security protocols for Wi-Fi networks are discussed in detail. Since EAP-NOOB employs the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol, a summary of the ECDH key exchange procedure is also provided. Finally, this chapter ends with a short description of out-of-band (OOB) channels used for authentication.

2.1 Lifecycle of an IoT Device

The following explanation is based on [37, 51]. As depicted in figure 2.1, the lifecycle of a smart object begins when it is manufactured. Depending upon the area of application, IoT devices are designed to perform various tasks such as information gathering and automation tasks. In a particular deployment of IoT devices, it is highly unlikely that all the devices are from a single manufacturer. Hence, it is of vital importance to ensure that devices from different manufacturers interoperate as well as connect to the Internet securely.

After its manufacture, an installer installs and commissions the IoT device within a network. Depending on the scenario in which the devices are deployed, the installer may be the end user, a manufacturer or some third party. All the essential configuration information such as credentials, addresses and ownership that are necessary for the device to start functioning are provided to it during this bootstrapping phase. This is the phase where EAP-NOOB comes into action; to securely bootstrap IoT devices with ease. The device enters the operational phase once it has been securely bootstrapped into the

network.

The device will be under the control of its owner during the operational phase. There might be occasional maintenance phases depending upon the lifetime of a device. During a maintenance phase, the firmware or software on the device is upgraded either locally or remotely. Re-bootstrapping of the device may be required after a maintenance phase if the device has lost the necessary state information. The device continues to loop between the operational and maintenance phases until it is decommissioned. The decommissioning of the device signifies the end of its lifespan. However, this does not imply that the device has become faulty or unusable. The detached device can be re-commissioned to start over the lifecycle under a different owner.

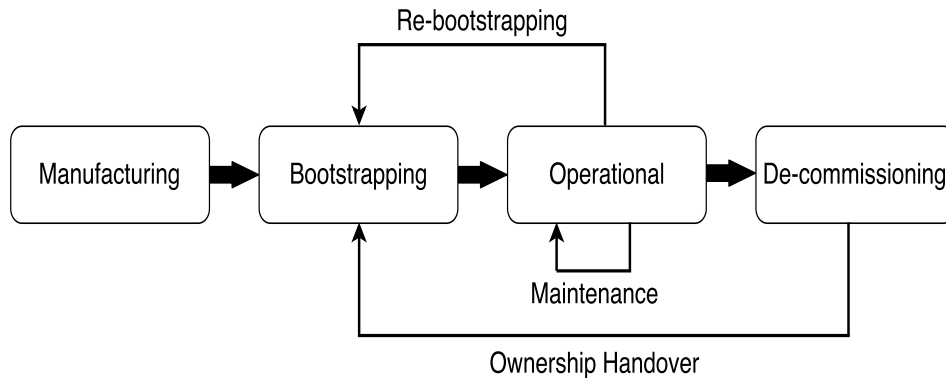


Figure 2.1: Lifecycle of a device [38]

Before diving into the details of secure bootstrapping, let us briefly look at the primary goals of network security in the next section.

2.2 Objectives of Network Security

The primary goals of network security are confidentiality, integrity, and availability, often represented as CIA triangle. So, whenever the word “security” is used in the context of communication networks, it typically means that CIA triangle is being talked about.

- *Confidentiality* means to protect data from unauthorized access. In other words, data should be made available only to authorized and intended entities. The mechanism to achieve confidentiality is obscuring or encryption of the data.

- *Integrity* means to protect the data from unauthorized modification, so as to ensure the accuracy and consistency of the data. The mechanism to achieve integrity is hashing. In addition to integrity, the authenticity of the message can be obtained through digital signatures and keyed hashes.
- *Availability* means to ensure that data and network resources are always at the use of legitimate users. Availability usually cannot be guaranteed, but the system should be designed so that denial-of-service attacks are not very easy.
- *Access Control and Authorization* means providing access only to authenticated and authorized entities. This is one way to ensure availability.

There are several security mechanisms or protocols developed to meet the goals of network security. However, these protocols are typically designed in such a way that they can be deployed at a particular layer of the TCP/IP model. Hence, to enhance network security, it may be required to use multiple security protocols. For instance, one might use the WPA2 protocol to secure layer 2 data (data link layer), IPsec to secure layer 3 data (network or internet layer), SSL/TLS to secure layer 4 data (transport layer), and so on. This is because a security protocol employed at a higher layer of the TCP/IP model cannot protect the data at lower layers and also the upper layers are not aware of the functions performed by the lower layers. EAP-NOOB is a security protocol employed to secure layer 2 data, i.e., the data link layer. As will be discussed later, EAP-NOOB can also help secure the application layer.

2.3 Secure Bootstrapping

The dictionary meaning of the term *bootstrapping* is “the technique of starting with existing resources to create something more sophisticated and efficient” [3]. The principle remains the same in the context of network security. In network security, when a device tries to establish a connection with either another device or an access network, appropriate security protocol entities deployed within the device perform the task of establishing a security association. These security protocols are called bootstrapping methods, and the process of creating a security association is called bootstrapping. Sethi [38] has categorized the various bootstrapping methods into four types:

- **Managed methods:** These methods depend on pre-established authentication credentials and trust relations. They typically make use of logically centralized servers for authentication. Examples are the Extensible Authentication Protocol (EAP) [21], Kerberos [25], Generic Bootstrapping Architecture (GBA) [1, 41] and vendor certificates [18].
- **Ad-hoc and peer-to-peer methods:** These methods do not depend on pre-established credentials. Instead, new credentials are established for subsequent secure communications by the bootstrapping protocol. Examples are various device pairing techniques [23, 36, 40].
- **Leap-of-faith and opportunistic methods:** These methods verify the continuity of the initial connection or identity rather than the original authentication. Examples are Secure Shell (SSH) [58] and Wi-Fi Protected Setup (WPS) push button [16].
- **Hybrid methods:** These methods use components from both ad-hoc and managed methods. Most of the deployed methods are hybrid methods.

The EAP-NOOB protocol can be considered as a hybrid method because it uses a centralized server for managing the devices after their ad-hoc registration with the server.

Secure bootstrapping is a crucial phase in the lifecycle of an IoT device because it should securely attach to an access network. Otherwise, if there is any breach or compromise during this phase, the device will remain vulnerable throughout the rest of its operational lifespan. In this context, bootstrapping may fulfil the following goals:

- Authentication and authorization of the devices desiring to join the network by an authentication server. Also, optionally the server may authenticate itself to the device.
- Configuration of security parameters for secure communication, i.e., deriving security keys between the server, network, and the device for secure communication.
- Registration, establishing ownership, and grouping of authenticated devices.

The types of access networks [57] relevant to this thesis are Ethernet and IEEE 802.11 Wireless LAN, i.e., Wi-Fi. In other words, these are the two technologies through which a device can obtain a connection to the Internet.

Both the technologies offer their own benefits. However, the easy deployment and mobility feature that a wireless connection provides are most useful. Also, almost all the modern day smart devices are Wi-Fi enabled. For example, smartphones and laptops are the two most popular smart devices with the capability to connect to a Wi-Fi network. In recent times, Wi-Fi networks have become ubiquitous in home and work environments. This has made it convenient for the users to connect large number of devices to the Internet without any marginal cost. Besides, users are also acquainted with Wi-Fi networks, especially in the office or home environments. This clearly signifies that Wi-Fi has become an integral part of our daily life. Nonetheless, as nicely stated by Morgan Rhodes "All magic comes with a price." This applies to Wi-Fi networks as well. Because the communication is over a wireless channel in Wi-Fi networks, anyone within the coverage area of a Wi-Fi network has access to the data transmitted in that network. Hence, compared to its wired counterpart, Wi-Fi is more vulnerable to attacks. Therefore, robust security mechanisms are essential to secure Wi-Fi networks. The subsequent sections discuss these security mechanisms in detail.

2.4 Security Protocols for Wi-Fi Networks

Wi-Fi networks can be deployed to operate in two modes, either infrastructure mode or ad-hoc mode (peer-to-peer). Hereafter, our focus will only be on Wi-Fi networks employed in the infrastructure mode, although solutions also exist for securing Wi-Fi P2P communications [16, 56]. In the infrastructure mode, all the client stations (STA) are attached to an access point (AP), such as a wireless network router, and all the communications happen through this AP. In Wi-Fi terminology, a STA is a device with the capability of connecting to the IEEE 802.11 networks. A Wi-Fi network could be either secured or open, i.e., insecure or public. In public Wi-Fi networks, the link layer data is not protected, or it is protected by a shared key that is known to all stations. Hence, the users in such public networks are susceptible to security breaches [46] and particularly to data sniffing. Therefore, it is always advisable for the users to be cautious while using public Wi-Fi networks.

A secured Wi-Fi network tries to ensure that at least the minimum goals of network security are met. As discussed in section 2.2, these goals are confidentiality, integrity, availability and access control. Wi-Fi Protected Access 2 (WPA2) is the current generation standard security protocol as well as a security certification program developed by the Wi-Fi Alliance to secure Wi-Fi networks. Wi-Fi Alliance was founded by a group of major companies that bring us Wi-Fi [13]. It defines a subset of IEEE 802.11 stan-

dards with some extensions, as depicted in figure 2.2. Any manufacturer desiring to obtain the certification must satisfy the criteria defined by the Wi-Fi Alliance. The Institute of Electrical and Electronics Engineers (IEEE) is another professional association of technical professionals with the objectives of educational and technological advancement in computer engineering, telecommunications, electrical and electronic engineering, and allied disciplines. It operates a group called the Standard Association (SA) which is responsible for the IEEE 802 family: LAN and MAN. IEEE 802 is subdivided into groups and among these groups, the “.11” working group. IEEE 802.11 is responsible for making standards for wireless LANs. The next section discusses WPA2 in more detail.

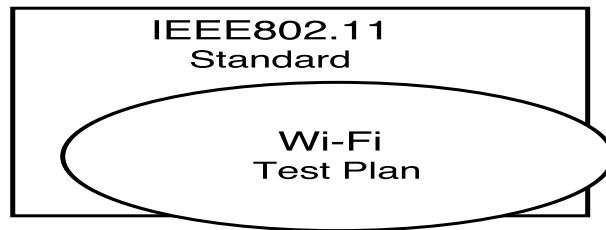


Figure 2.2: Relationship of Wi-Fi to IEEE 802.11

2.5 WPA2 and RSN

WPA2 is based on an addendum to original IEEE 802.11 standards called IEEE 802.11i. The IEEE 802.11i defines a new type of wireless network known as robust security network (RSN). Some of the primary security goals of IEEE 802.11i are:

- Access control, and mutual authentication between the client STAs and the network. As will be discussed later, this is achieved by using the IEEE 802.1X protocol.
- Data confidentiality, integrity and authenticity. The IEEE 802.11i standard provides two RSNA data integrity and confidentiality protocols: Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP) [29] and Temporal Key Integrity Protocol (TKIP) [17], with mandatory implementation of CCMP. TKIP is deprecated because it was intended for use during a transition period and is no more secure.

- Key Management or key distribution. As will be discussed later, this is also accomplished by employing the IEEE 802.1X protocol.
- Replay Detection using CCMP.

In RSN networks, the creation of robust security network associations (RSNAs) is only allowed. RSNA is a type of connection used by an AP and its client STA if 4-Way Handshake procedure is included in the process of establishing authentication or association between them [15]. It is important to note that, in a network, robust security is achieved when all the connections between the AP and its client STAs are only RSNA. Section 2.5.6 discusses the RSNA establishment procedure between an AP and its client STA in detail.

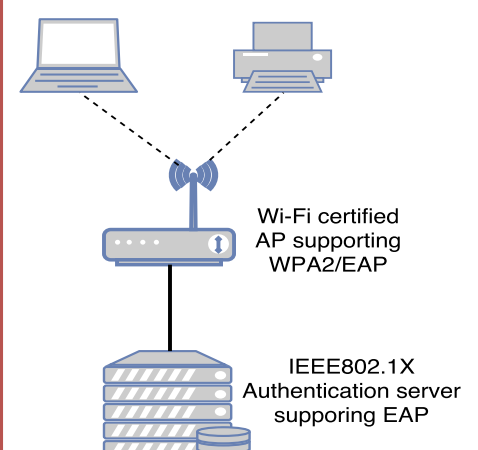
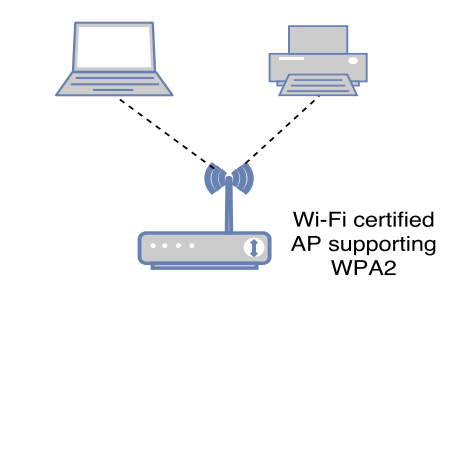
WPA2-Enterprise	WPA2-Personal
Unique credentials are assigned for each user	Unmanaged authentication using PSK. Typically, the PSK is shared by all clients of that network.
AAA server supporting EAP is required	Authentication server is not required
Unique data security keys for each session	Unique data security keys for each session
<p>Wi-Fi certified client devices with WPA2-Enterprise</p> <p>STA1 STA2</p>  <p>Wi-Fi certified AP supporting WPA2/EAP</p> <p>IEEE802.1X Authentication server supporting EAP</p>	<p>Wi-Fi certified client devices with WPA2-Personal</p> <p>STA1 STA2</p>  <p>Wi-Fi certified AP supporting WPA2</p>

Table 2.1: WPA2-Enterprise and WPA2-Personal

Depending on the requirements of the network, WPA2 has two modes of operation: Personal mode and Enterprise mode. In the Personal mode, a

pre-shared key between the AP and its client STA will be used for the authentication, whereas, in Enterprise mode, the IEEE 802.1X authentication will be employed. WPA2-Personal, also called as WPA2-PSK, is intended for small office and home networks. This is because it is easier to set-up and maintain WPA2-Personal networks in comparison to WPA2-Enterprise networks. There are few situations where WPA2-PSK is ideal to use. Firstly, when there are only trusted devices and users in the network, and this could be a small office or home. Secondly, it may be used as the means to restrain casual users from accessing the Internet in a public network, which may be reserved for the customers of a hotel or a coffee shop. Table 2.1 depicts the differences between the two modes of WPA2.

EAP-NOOB is a protocol aimed at securing the IoT devices deployed in a WPA2-Enterprise network. Before we look into the details of RSNA establishment procedure and the part EAP-NOOB plays in it, let us first go through various protocols required to understand the RSNA establishment procedure.

2.5.1 IEEE 802.1X

One of the fundamental elements of network security is access control. It follows that security cannot be achieved unless there is a mechanism to perform the segregation between authorized and unauthorized users.

IEEE 802.1X is a standard that defines a mechanism to implement Port-based Network Access Control (PNAC). Port here refers to a point at which a system gets attached to a LAN. The port can be either a physical port or a logical one. For instance, each Ethernet connector in a LAN switch can be considered as one port, or an IEEE 802.11 association between an access point and a STA can be seen as a logical port. The 802.1X involves three entities:

- *Authenticator*: An entity placed at one end of a point-to-point LAN segment to facilitate the authentication of entities on the other end of that link. In other words, an authenticator mediates the authentication process between a supplicant and an authentication server. All the non-EAP traffic through a port is blocked until the supplicant attached to that port is successfully authenticated by the authentication server. In simple words, the port remains unauthorized and disconnected until successful authentication, as depicted in figure 2.3.
- *Supplicant*: An entity attached to one end of a point-to-point LAN segment which intends to join a network to use the services offered by

the authenticator and, hence, is authenticated by the authenticator at the other end of that link. An example of services can be access to the Internet. In this thesis, the terms supplicant, peer and client are used synonymously.

- *Authentication Server (AS)*: An authentication server provides authentication service to the authenticators. In other words, the authentication server authenticates the supplicants and informs the Authenticator of the result. The authentication server can be either co-located with an authenticator or remotely accessed by the authenticator through a network to which it has the access. In this thesis, the terms authentication server and Authentication Authorization Accounting (AAA) server are used synonymously.

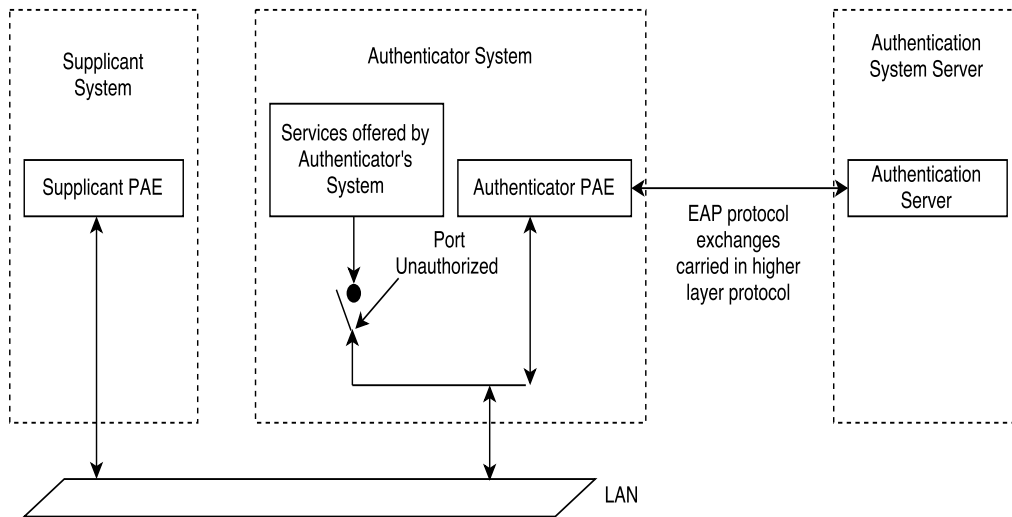


Figure 2.3: 802.1x architecture [15]

Figure 2.3 illustrates the relationships between the entities described above. In this figure, Port Access Entity (PAE) is a protocol entity associated with a port. It supports the protocol functionality related to the supplicant, the authenticator, or both.

WPA2 in enterprise mode adopts the IEEE 802.1x protocol for authentication and key management. However, the IEEE 802.1X standard does not define any particular authentication protocol itself. Instead, it relies on an upper-layer protocol ¹ called Extensible Authentication Protocol (EAP) for

¹any protocol residing in OSI layer five or above

an authentication mechanism. It must be made categorical that 802.1X is neither an authentication protocol nor does it ensure a secure authentication mechanism for wireless applications. Its standard defines encapsulation rules for the transport of EAP messages over LAN (EAPOL) and was originally designed for wired LAN. But later 802.1X was recognized to be suitable for other IEEE 802 LAN technologies such as 802.11 networks, i.e., WLANs. CISCO first incorporated the 802.1X concepts into its enterprise network products, and other vendors followed. Later, the approach was adopted in the IEEE 802.11i addendum for RSN and, subsequently, by the Wi-Fi Alliance. In figure 2.3, there is also a reference to a higher-layer protocol between the authentication server and the authenticator. This protocol is employed to transport the EAP messages over an IP network between the authenticator and the remote authentication server. There are two such transport protocols available: Remote Authentication for Dial-in User Service protocol (RADIUS) [26] and Diameter protocol [53]. The RADIUS protocol is widely used in Wi-Fi networks, and for the implementations of this thesis, we will use the RADIUS protocol.

2.5.2 EAP

Extensible Authentication Protocol (EAP) [21] is an authentication framework, and it includes several authentication methods known as EAP methods. A subset of EAP methods are used in WPA2-Enterprise networks and these methods are responsible for performing the authentication and key agreement in WPA2-Enterprise networks. EAP defines a set of messages that are used by the EAP methods to perform the authentication and key exchange. There are four types of EAP messages:

- EAP-Request: These request messages are sent or relayed by the authenticator to the supplicant. The type field in the message specifies what is being requested.
- EAP-Response: These response messages are sent by the supplicant as a reply to a corresponding request message.
- EAP-Success: This message is sent by the authentication server (AS) to the supplicant via the authenticator to indicate successful authentication.
- EAP-Failure: This message is sent or relayed by the authenticator to the supplicant to indicate an authentication failure.

Note: It is important to point out that the authenticator must not send EAP-Failure or EAP-Success messages if the given method does not explicitly specify to end the EAP authentication process at that point.

EAP defines two types of authenticator modes: Stand-alone mode and pass-through mode. In stand-alone mode, the authentication server is co-located with the authenticator, whereas in pass-through mode, the authenticator is connected to a remote authentication server and it just acts as a mediator by relaying the EAP messages between the supplicant and the authentication server during the EAP authentication process. In other words, the authenticator does not have to understand any EAP methods as the implementation of any EAP authentication method will be only on the authentication server and the supplicant. The main advantage of this is that new EAP methods can be added to the EAP framework without making any modifications to the existing access network infrastructure. Another advantage of the pass-through mode is that a centralized authentication server can be used to serve multiple authenticators. This thesis work focuses only on the pass-through mode.

2.5.3 EAPOL

The EAP specification does not specify how EAP messages should be carried. For instance, it does not determine how EAP messages are transported over the Internet using IP. Actually, EAP is not a LAN protocol. It was originally designed to be employed with dial-up authentication through a modem. Hence, IEEE 802.1X defines EAPOL protocol to encapsulate and transport EAP messages between the supplicant and the authenticator. There are five types of EAPOL frames or messages. Among these, the EAPOL-Encapsulation-ASF-Alert message is not used by WPA2/RSN. The other four messages are:

- **EAPOL-Start:** The supplicant will not know the MAC address of the authenticator when it first connects to the LAN. In fact, the supplicant will not even know whether an authenticator is present. Hence, IEEE 802.1X defined this optional message to help get things started. By sending this Start frame to a group-multicast MAC address specifically reserved for the authenticators, a supplicant can discover an authenticator and notify the authenticator that it is ready. However, in many cases, the authenticator may already be aware that a new supplicant device has connected. For instance, a switch knows that an ethernet

cable is plugged in before it receives any data from the device. In this case, the Start message may be preempted by the authenticator's EAP-Request Identity message. Nonetheless, in both the instances, an EAP-Request Identity message encapsulated within EAPOL-Packet frame will be sent by the authenticator (perhaps twice, if both simultaneously send the initial message).

- **EAPOL-Key:** This frame is used during the 4-way handshake procedure to establish both data encryption and integrity keys, and also to confirm that both the parties have the right keys before granting network access. As will be described later, this frame is also used to send the encrypted Group Transient Key(GTK) to the supplicant by the authenticator.
- **EAPOL-Packet:** This is just a container frame for transporting the actual EAP messages across the LAN, which is the original intention of the EAPOL protocol.
- **EAPOL-Logoff:** This frame is sent by the supplicant to indicate that it wants to disconnect from the network.

2.5.4 RADIUS

RADIUS, an acronym for Remote Authentication Dial-In User Service, is defined in RFC 2865 [26]. RADIUS is based on the client/server model, and it runs at the application layer. It describes two things: firstly, a set of capabilities that should be common across authentication servers, secondly, a protocol for allowing other devices to access those capabilities. The set of capabilities is Authentication, Authorization, and Accounting, popularly known as AAA. The AAA capabilities mean authentication of devices or users before granting them access to the network, authorization of those devices or users for specific services offered by the network, and accounting of the usage of those services.

A Network access server (NAS), a gateway that controls the access to a network, i.e., the IEEE 802.1X authenticator, generally contains a RADIUS client that communicates with the RADIUS server to interact with the AAA capabilities. In the WPA2/RSN context, the AP is the equivalent of the NAS. As illustrated in figure 2.4, when we talk about a RADIUS server, it refers to a part of the authentication server with AAA capabilities; and when we talk about RADIUS, it relates to the protocol used to communicate with the RADIUS server.

RADIUS can carry the messages of a variety of authentication methods such as Point-to-Point Protocol Password Authentication Protocol (PPP PAP), Challenge-Handshake Authentication Protocol (CHAP), and EAP. However, our interest is only on the EAP, and RFC2869 [27] specifies how to use EAP over RADIUS. RADIUS has four types of relevant messages to carry EAP messages. They are:

- Access-Request: These messages are sent by NAS to AS and are used to encapsulate EAP response messages from the supplicant.
- Access-Challenge: These messages are sent by AS to NAS and are used to encapsulate EAP request messages to the supplicant.
- Access-Accept: This message is sent by AS to NAS as a result of EAP authentication process. This message encapsulates the EAP-Success message and a cryptographic session key.
- Access-Reject: This message is sent by AS to NAS as a result of the EAP authentication process. This message encapsulates the EAP-Failure message, and no session key is transmitted.

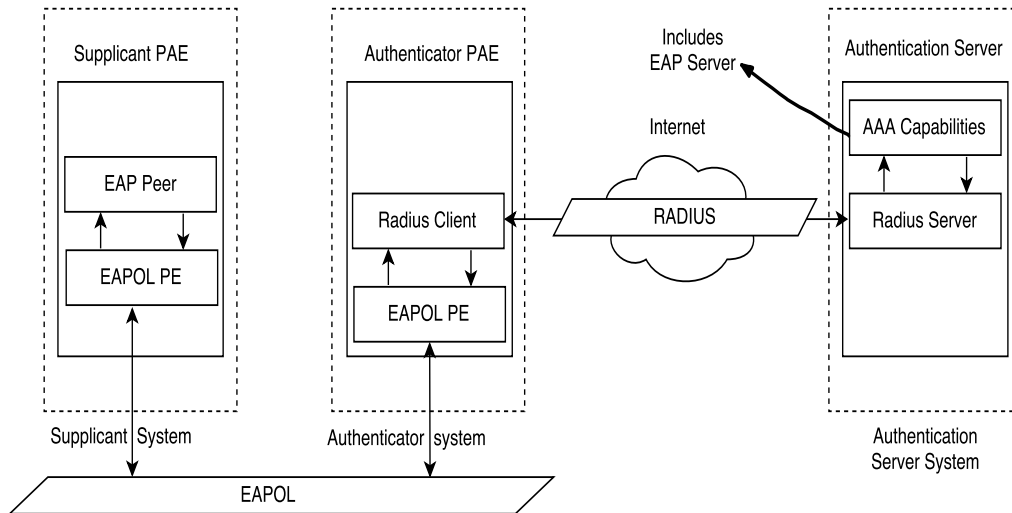


Figure 2.4: WPA2-Enterprise network architecture

In addition to EAP messages, RADIUS messages also carry specific additional information in the form of attribute-value pairs (AVPs) pertaining to authentication, authorization, and accounting. In fact, an EAP message

is also carried as an AVP. Though RADIUS can either use UDP or TCP for transport, for the reasons stated in section 2.4 of RFC 2865 [26], UDP is used. A RADIUS server can also function as a proxy client to other kind of ASs or other RADIUS servers. There are several ways through which the link between the RADIUS client and the RADIUS server can be secured: a shared secret can be used between them or RADIUS over TLS [49] or RADIUS over DTLS [20] can be employed.

2.5.5 EAP Authentication Process

As discussed in the earlier subsections, the EAPOL protocol carries the EAP messages between the supplicant and the authenticator, and the RADIUS protocol carries the EAP messages between the authenticator and the AS. The EAP authentication process triggers when the authenticator sends an EAP-Request/Identity message to the supplicant. After receiving the request message, the supplicant responds with an EAP-Response/Identity message. This identity message contains a Network Access Identifier (NAI). As defined in RFC 4282 [22], the NAI consists of two parts: a username part that may uniquely identify the supplicant at least within its home domain and a realm part that identifies the supplicant's home realm. The identity message is particularly used for two purposes. First, it is used to select an EAP method for authentication, and second, it may also be used for routing purposes, i.e., domain part of the NAI is used to decide the server to which the messages are supposed to be sent. The supplicant may opt to disclose its username part of the NAI only to its home server via a secured connection. Hence, section 5.1 of RFC3748 [21] recommends the EAP methods to incorporate a method-specific mechanism to obtain the proper NAI, therefore avoiding sending it in the EAP-Response/Identity message. After receiving the identity response message, the AS chooses an appropriate EAP method and the method specific authentication begins. The authentication process may include the exchange of multiple EAP-Request/Response messages. It is imperative to note that EAP is a “lock-step” protocol, which means that a new EAP-Request message cannot be sent before receiving a valid EAP-Response to the previous Request. After the authentication is complete, the AS conveys the result to the authenticator.

In the case of successful authentication, a Master Session Key (MSK) will be derived between the supplicant and the AS. The MSK along with the EAP-Success message will be delivered to the authenticator by the AS. The authenticator keeps the MSK for itself and forwards only the EAP-Success message to the supplicant. This marks the end of EAP authentication process.

On the other hand, if authentication fails, the AS notifies the authenticator via an EAP-Failure message which will also be forwarded to the supplicant, and this marks the end of EAP authentication process. The supplicant will remain blocked from accessing any services offered by the authenticator until a successful authentication happens.

Now that the concepts required to understand the RSNA establishment procedure have been covered, the RSNA establishment process will be discussed in the next subsection.

2.5.6 RSNA Establishment Procedure

Figure 2.6 depicts the 802.11i Robust Security Network Association (RSNA) establishment procedure. The establishment procedure can be divided into 6 stages [34]. This section describes each stage in detail.

Stage 1, the network and security capability discovery phase, involves the exchange of messages numbered (1) to (3) between the AP and the supplicant. There are two ways through which a supplicant may discover the available APs and their security capabilities. First, it may actively probe every wireless channel for a Probe Response frame from an AP. Second, it may passively monitor the Beacon frames which are periodically broadcasted by the APs. The Beacon or Probe Response frame sent by an AP contains the Robust Security Network Information Element (RSN IE) which indicates its security capabilities: all enabled authentication suites, all enabled cipher suites for unicast, and cipher suite for broadcast.

Stage 2, the 802.11 authentication and association phase, involves the exchange of messages numbered (4) to (7) between the AP and the supplicant. The supplicant selects one access point from the list of available access points. Then, the supplicant tries to authenticate and associate with the selected access point. Initial 802.11 standards had defined two types of authentication methods: Open System Authentication and WEP Shared Key authentication. These two authentication mechanisms are known to be insecure [42, 43]. However, open system authentication, which is just two empty messages, has been retained for backward compatibility and is supplemented by further steps. Also, pre-shared key (PSK) or 802.1x/EAP authentication should not be confused with the open system authentication. After the open system authentication, which is always successful and does not provide any security, the supplicant indicates its security capabilities in the Association request. At the end of this stage, the supplicant and the AP will be in authenticated and associated state. However, no real authentication has yet taken place, the 802.1X port will remain blocked, and no data packets are allowed.

Stage 3, authentication process based on the IEEE 802.1X protocol, consists of the messages numbered (4) to (7). The authentication server and the supplicant execute an EAP method, for example, EAP-TLS [28], for mutual authentication, with the AP usually acting as a relay agent. The EAP authentication process is carried out as described under subsection 2.5.2. At the end of this stage, the AS and the supplicant have authenticated each other and derived a common secret, called the MSK. The MSK will be securely delivered to the NAS or AP by the AS, indicated by message (15). Another common secret, called the Pairwise Master Key (PMK), will be generated by the supplicant and the AP from the MSK. This phase might be skipped in two scenarios. First, during a re-association, if a cached PMK is used. Second, in WPA2-Personal mode, where the supplicant and the AP are configured to use a pre-shared key as the PMK. These two scenarios can also be considered as other two means to get the PMK.

Stage 4, the 4-Way Handshake procedure, involves the exchange of messages numbered (16) to (19) between the AP and the supplicant. The 4-Way Handshake procedure is executed regardless of the means by which the PMK was obtained. The three different ways to get the PMK are as mentioned above. The 4-way handshake procedure between the supplicant and the AP serves three purposes: to confirm that both have the same PMK, to verify the cipher suite selection, and to derive a new Pairwise Transient Key (PTK) for protecting the data exchanged during the following data session. At the same time, the AP might also issue a Group Transient Key (GTK), which is encrypted using the PTK, via message (18). All the supplicants within a network are given the same GTK by the AP, and the GTK is used to protect the broadcast messages. Figure 2.5 depicts the RSN key hierarchy. After this phase, the supplicant and the AP share the new PTK (and maybe GTK) and the IEEE 802.1X port is unblocked for data packets. Through the 4-Way handshake, the AP and the supplicant mutually authenticate each other and begin a secure session for data communication. Also, any tampering of RSN IE exchanged between the supplicant and AP will be detected during the 4-Way handshake as both the parties retransmit their respective security capabilities with integrity protection.

Stage 5, the Group key handshake, involves the exchange of messages numbered (20) to (21) between the AP and the supplicant. To protect the broadcast traffic, the AP generates a new GTK from the Group Master Key (GMK), a random key generated by the AP, and distributes the GTK to all its supplicants. This stage might be skipped if the GTK was issued to the supplicant during the 4-Way Handshake. However, the Group key handshake may be rerun multiple times using the same PMK in order to update the group key. Though both the data and broadcast messages are protected in a

given WPA2 network, there is no message origin authenticity for broadcast messages as all the supplicants use a same GTK to protect the broadcast messages.

Stage 6, secure data communication between the AP and its supplicants, is indicated by (22). The supplicant and the AP can protect the data packets between them by employing the negotiated cipher suites from the above stages along with the PTK (or GTK). Data protection means confidentiality and integrity protection of the data using CCMP.

In a WPA2-Enterprise network, secure bootstrapping occurs during stage 3 of the RSNA establishment procedure. Hence, Stage 3 can be considered a crucial stage, and this is when EAP-NOOB comes into action, assisting a user to bootstrap new IoT devices to the network with ease.

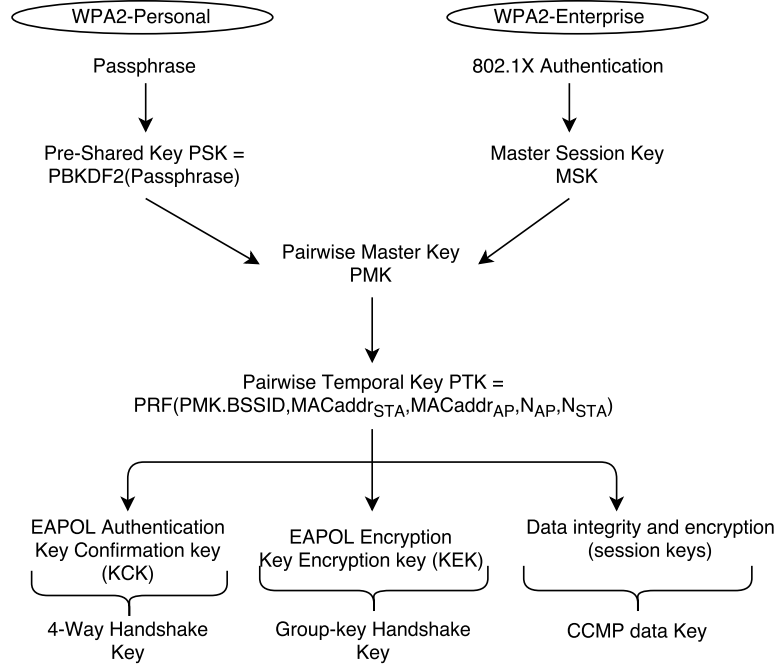


Figure 2.5: RSN key hierarchy

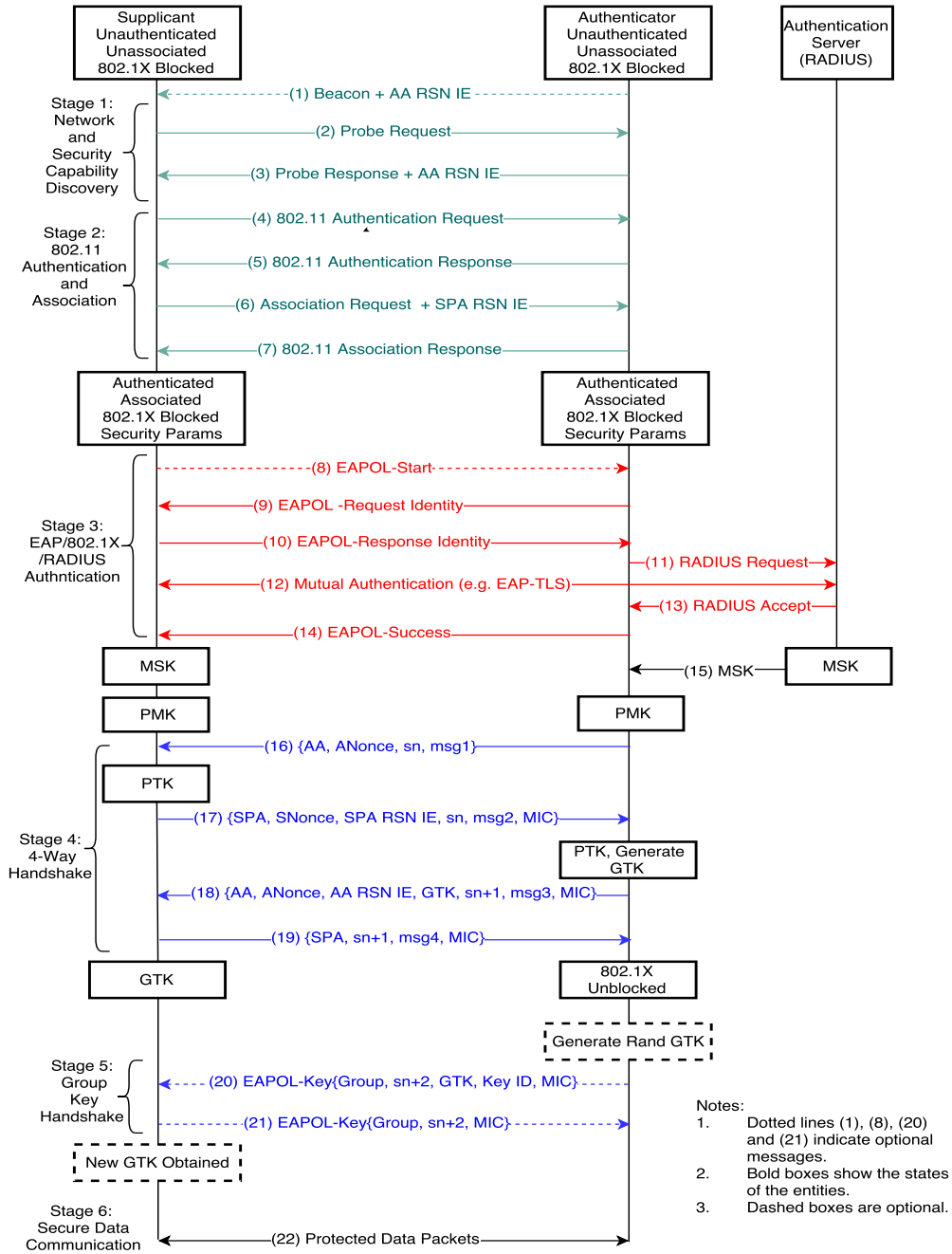


Figure 2.6: RSNA establishment procedure(adapted and modified [34])

2.6 ECDH Key Exchange Procedure

The cornerstone of the EAP-NOOB protocol's capability to securely bootstrap IoT devices without any pre-configured credentials is the Diffie-Hellman (DH) key exchange method, a public-key cryptography protocol defined by Whitfield Diffie and Martin Hellman [55]. The DH key exchange procedure enables two entities that do not have prior knowledge of each other to derive a common secret key over an insecure communication channel mutually. This key can then be utilized to either secure the subsequent communications or derive more secure keys between those two entities. The security offered by DH key exchange procedure relies on the discrete logarithm problem.

Elliptic Curve DH (ECDH) is a variant of the DH algorithm, including the benefits of Elliptic Curve Cryptography (ECC) [24]. In other words, ECDH uses points on elliptic curves to perform key agreement rather than traditional integers as employed in DH. The primary advantage of using ECDH is reduced private key size in comparison to other public key cryptosystems to get the same level of security [4]. The shorter key implies relatively less computation. Hence, ECDH key exchange mechanism is employed in our EAP-NOOB protocol for key distribution. A detailed explanation of ECDH key exchange procedure is as follows,

Let us consider Alice and Bob as communicating parties.

- Initially, the involved parties share or agree upon the system parameters, p : a finite field, a and b : curve constants and G : a base point.
- Alice chooses a random number r and computes $l = Gr$. Here, l is the public key and r is the private key. Alice then sends her public key to Bob.
- Likewise, Bob chooses a random number s and computes $m =Gs$. Bob then sends his public key to Alice.
- Shared secret key, $Z = mr$ is computed by Alice. And, Bob calculates $Z = ls$.
- Both Alice and Bob have derived same secret key $Z = Grs$.

Similar to DH key exchange mechanism, the security offered by ECDH key exchange procedure relies on the elliptic curve discrete logarithm problem. In simple words, the security depends on the difficulty in computing the secret 'r' given the public parameters 'p', 'G', 'l' and 'm'.

The major downside of the ECDH or DH key exchange mechanism is the anonymity of the process, i.e., the established common secret is not authenticated. Therefore, ECDH or DH algorithms are vulnerable to man-in-the-middle (MiM) attacks. In general, ECDH or DH key exchange algorithms are complemented by other methods to authenticate the established shared secret key. For example, in TLS [52], an extended version of the Diffie-Hellman algorithm is used in conjunction with TLS communication protocol to establish an authenticated common secret key between two parties. As used in the EAP-NOOB protocol, another method to authenticate the common secret established via ECDH or DH key exchange is through out-of-band (OOB) channels. The next section provides a brief discussion of OOB channels.

2.7 OOB Channels for Authentication

Out-of-band (OOB) channel refers to a physically or logically separate communication channel other than the main communication channel, also called the in-band channel, between two communicating parties. Other synonyms for OOB channels include human-assisted channels, auxiliary channels, and manual channel [48]. OOB channels are generally employed to exchange specific or confidential information. For instance, online banking systems utilize email services or SMS services as an OOB channel either to alert a user about his account transactions or deliver one-time passwords (OTPs) required to complete a particular transaction on the in-band channel, which in that case is the Internet.

In the case of network security, OOB channels are extensively employed for bootstrapping the communication security for the in-band channel by performing authentication using an OOB communication channel. In other words, to either authenticate the information sent over the in-band channel or else establish an authenticated shared secret key to be used for subsequent secure communications. This is typically due to the reason that OOB channels are considered to be robust against Man-in-The-Middle (MITM) attacks. Different OOB channels may offer different security properties. For example, the security properties of OOB channels, as stated by Shahab et al. [10], may include: authentication based on a proven identity, data integrity, data origin authenticity and data confidentiality. Examples of various types of OOB channels are NFC, Quick Response (QR) Code, Bluetooth and Audio waves.

As discussed in the previous section, an OOB channel may be employed between two entities to authenticate a common secret established between them using ECDH or DH algorithms via in-band. Since EAP-NOOB also

utilizes the ECDH algorithm for key agreement, the established key needs to be authenticated via an appropriate OOB channel. The implementation of the OOB channel is not a part of the EAP-NOOB protocol specification and it is left as an implementation choice. However, the protocol assumes that the OOB channel is secure, and the message sent via the OOB channel cannot be both sniffed and modified by an attacker. The next chapter provides an overview of the EAP-NOOB protocol.

Chapter 3

Protocol Overview

The objective of this chapter is to give an overview of the EAP-NOOB protocol and its initial prototype implementation as it is imperative to know these to understand the subsequent chapters. This chapter begins by giving a brief overview of the EAP-NOOB protocol. After that, the motivation for EAP-NOOB along with security goals and a brief description of the implementation details of the initial prototype are provided.

3.1 A Brief Overview of the Protocol

One execution of the EAP-NOOB protocol spans multiple EAP sessions. As explained below, this is required to provide time for the delivery of the OOB message. Figure 3.1 depicts the association state machine for this protocol. The state machine remains the same for both the server and the supplicant.

Initially, as shown in figure 3.1, both the supplicant (also called EAP peer) and the authentication server will be in the **Unregistered state**. The supplicant starts off by searching for an AP that supports WPA2-Enterprise and tries to associate with it. Once the AP and the supplicant have completed the 802.11 open authentication successfully and are associated, the AP requests the supplicant to send its identity or Network Access Identifier (NAI). The supplicant then sends the generic NAI string ‘noob@eap-noob.net’ for the first time because there will be no pre-configured identities for the supplicant or any information between the supplicant and the server. The AP will then forward the received NAI to the server to trigger an EAP-NOOB conversation between the server and the supplicant. From this point onwards, the only task of the AP is to relay the EAP messages between the supplicant and the server.

After the server receives the generic NAI from the supplicant, it detects

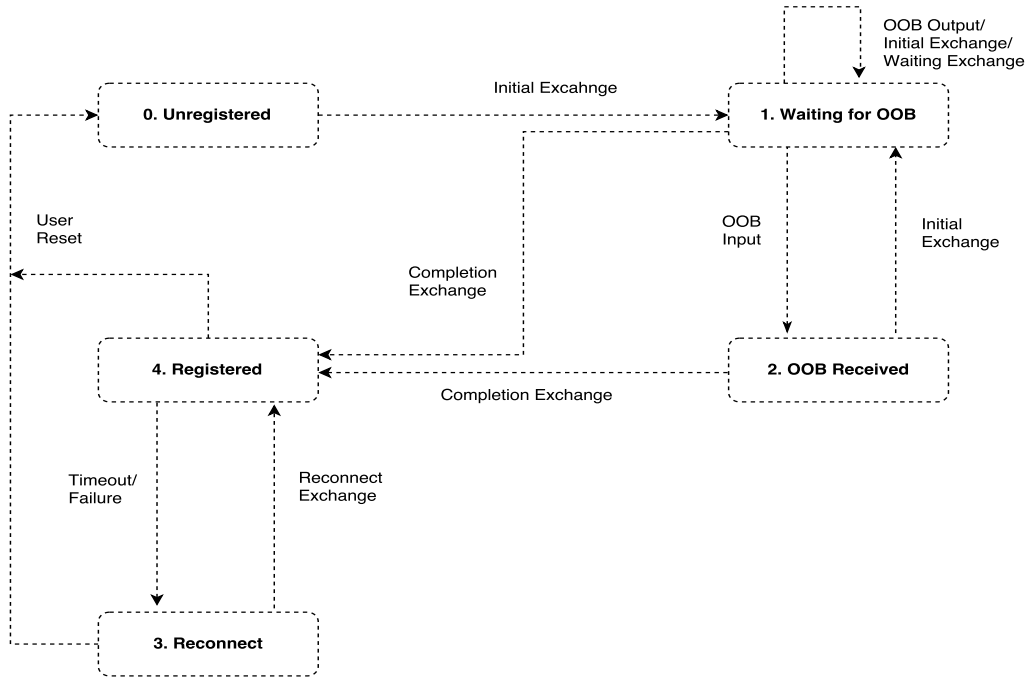


Figure 3.1: EAP-NOOB server-peer association state machine [51]

that the supplicant is new from the generic NAI and both the parties enter the **Initial Exchange phase**. During the Initial Exchange, the server assigns a unique identifier called PeerId to the supplicant, and they negotiate the EAP-NOOB protocol version and the cryptographic algorithm suite. The supplicant uses the assigned PeerId to construct the NAI for sending it in the subsequent EAP sessions. In addition, nonces are exchanged, and an Elliptic Curve Diffie-Hellman (ECDH) key exchange is performed between the server and the supplicant. The Initial exchange always ends in EAP-Failure, and both the parties move to the **Waiting for OOB state**.

The user-assisted **OOB step** is then performed. A single OOB message consisting of the PeerId assigned by the server to the supplicant, a nonce called ‘Noob’ and a hash value called ‘Hoob’ is delivered either from the server to the supplicant or from the supplicant to the server with the aid of the user. The direction of this OOB message is negotiated during the Initial Exchange. The nonce ‘Noob’ will be used in the derivation of cryptographic keys during the following Completion Exchange. The hash value ‘Hoob’ (i.e. cryptographic fingerprint) is calculated using all the parameters exchanged during the Initial Exchange and the OOB step. The delivery of this hash value along with a nonce over the OOB channel ensures the integrity pro-

tection of the parameter negotiation and authenticates to the recipient the ECDH key exchange performed during the Initial Exchange phase. The party that receives the OOB message moves to the **OOB Received state**, and the other party remains in the Waiting for OOB state. However, the implementation of the OOB channel is not a part of the protocol specification, and it is left as an implementation choice. For instance, if we consider the university example stated at the beginning, the displays can render their OOB messages as a QR code, and the user can scan and deliver it to the Authentication, Authorization and Accounting (AAA) server via a secure university website after authentication, i.e. after logging in to his or her account.

The supplicant might probe the server while waiting for the completion of the OOB step. The probe may result in the **Completion Exchange phase** on the satisfaction of two conditions. First, the OOB message should have been delivered successfully to the intended party, i.e., between the server and the supplicant. Second, the hash value ‘Hoob’ received in the OOB message should match the one computed locally by the recipient. During the completion exchange, the key confirmation and mutual authentication happen between the supplicant and the server. The secret nonce Noob from the OOB message is used as the shared authentication key.

If the key confirmation or the mutual authentication fails, it leads to an error. A mismatch or an error at any stage of the protocol leads to the delivery of an error message from one party to the other and is followed by an EAP-Failure message from the server to the supplicant. Based on the type of error, an appropriate action as specified in the draft [51] will be performed by the parties.

On the other hand, if the OOB message has not yet been delivered by the user, the probe results in a **Waiting Exchange**. There could be several Waiting Exchanges before the delivery of the OOB message. The time span between each Waiting Exchange and the number of times the Waiting Exchange can occur are specific to the server implementation. The Waiting Exchange always results in EAP-Failure, and both the parties remain in their respective current state.

At the beginning of the **Completion Exchange phase**, both the parties derive cryptographic keys using a Concatenation Key Derivation Function (KDF). The KDF is defined in the National Institute of Standards and Technology (NIST) specification [30]. The keys derived by both the parties are:

- Their respective and other party’s key confirmation keys, i.e., the server key ‘Kms’, and the supplicant key ‘Kmp’.
- A common key ‘Kz’.

- Application-layer session key: Application MSK (AMSK).
- The temporal keys: MSK and EMSK.

All the nonces exchanged via the OOB and in-band channels along with the common secret ‘Z’ derived through the ECDH key exchange will be used as the inputs to the KDF. Then, both the parties calculate and exchange their respective keyed-hash message authentication codes (HMAC): MACs and MACp. These HMACs are calculated using all the parameters exchanged both in the OOB and in-band channels along with the respective party’s key confirmation key. HMACs mutually authenticate both the parties and help in key confirmation along with ensuring the integrity protection of the exchanged information [31]. The received HMACs will be compared with the ones computed locally on both the sides. If the comparison is successful, it results in EAP-Success and both the parties move to the **Registered state**, or else it results in EAP-Failure, and both the parties remain in their current states assuming that the failed Completion Exchange did not happen.

Also, both the parties store the common key ‘Kz’, negotiated cryptosuite and the assigned NAI in persistent storage in order to avoid the repetition of the OOB step. This persistent data will be used in the future during the **Reconnect Exchange**. The section 5.1 will discuss the Reconnect Exchange in more detail. Figure 3.2 shows the EAP-NOOB key hierarchy.

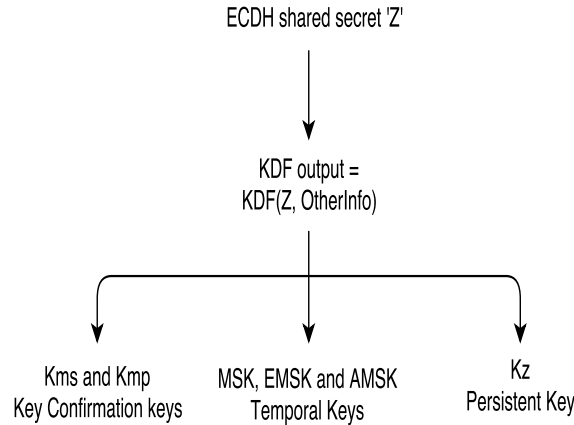


Figure 3.2: EAP-NOOB key hierarchy

The detailed explanation of the protocol along with the message exchanges between the authentication server and the supplicant can be found in the IETF draft [51].

3.2 Motivation for EAP-NOOB

The main motivation for EAP-NOOB is that it eases security bootstrapping of smart IoT devices with minimal UI, especially when they are many in number. Consider, for instance, an enterprise that has hundreds of new smart displays to be connected to their existing Wi-Fi network. It will be a tedious task to configure each of the displays with manual input manually. But employing EAP-NOOB solves this problem because a user just has to hang all the displays and transfer an OOB message for each of the displays to securely connect them to the existing Wi-Fi network. For example, each display can display its OOB message as a QR code, and the user can scan and deliver it to the server via a secure university website after authentication, i.e. after log-in.

In EAP-NOOB, the secret nonce ‘Noob’ and the hash value ‘Hoob’ are used in combination to provide protection. Both the nonce Noob and the hash value Hoob are sent in the OOB message. The delivery of the nonce Noob authenticates the ECDH key exchange performed during the Initial Exchange. The hash value ‘Hoob’ (i.e. cryptographic fingerprint) is calculated using the parameters exchanged during the Initial Exchange and the OOB step. The delivery of this hash value ensures the integrity protection of the parameter negotiation. In other words, if an attacker has performed a man-in-the-middle attack on the in-band channel, the mismatching cryptographic fingerprint will notify the OOB receiver, which will discard the OOB message. The user can easily detect this as the peer device does not seem to be registered on the network or server.

To break the protocol, an attacker has to perform both man-in-the-middle attack on the in-band channel and modification of the OOB message. It is more strenuous to spoof or modify messages on a human-assisted OOB channel, such as a QR code or sound burst, than it is to spy on them. The EAP-NOOB establishes a different master secret for each of the peer devices. This is clearly more resilient to device compromise than a shared common secret and also, makes device revocation easier.

3.3 A Brief Overview of the Initial Implementation

The EAP-NOOB implementation can be split into two fundamental components: EAP-NOOB method implementation and the OOB channel implementation. The EAP-NOOB method implementation comprises EAP-

NOOB peer and server implementations. We implemented and integrated the EAP-NOOB method into an open-source project [7] which has focused on UNIX-like operating systems. We choose this open-source project based on considerations such as the number of users, size of the community that supports the project and the frequency of updates. The two components of the open-source project that we are interested are *hostapd* and *wpa_supplicant*. The Host access point demon (*hostapd*) component is a user space daemon for authentication servers and APs. In other words, *hostapd* can be independently deployed to turn any regular network interface cards into authentication servers and APs. The authentication server part implements EAP server and RADIUS authentication server. The EAP-NOOB server was implemented within the authentication server part of *hostapd*. The *wpa_supplicant* component is an implementation of the IEEE 802.1X/WPA supplicant for UNIX-like operating systems. Like *hostapd*, *wpa_supplicant* also runs as a daemon program, and it controls the wireless connection. The EAP-NOOB peer was implemented within the *wpa_supplicant*.

As part of the OOB channel implementation, we developed a web server using Node.js. The users create their account with the web server, and the web server serves two purposes. First, it acts as one end of the OOB channel and runs alongside the EAP server. In other words, this web server is used to deliver the OOB messages to and from the EAP server, and it communicates with the EAP server via a local database. Second, it tracks the EAP-NOOB authentication process and also manages the successfully authenticated supplicant devices. Another important component involved in the implementation of the OOB channel is an OOB device. An OOB device contains the applications and tools required for the user to convey the OOB messages either in the server-to-peer direction or peer-to-server direction. In this implementation, we extensively used smartphones as they are commonly available and possess the necessary applications and tools such as camera, QR code scanner and NFC for automating the OOB message transfers. As part of the initial implementation, two OOB channels were implemented:

- Transfer of OOB messages in either of the directions with QR codes. In other words, scan and deliver an OOB message displayed as a QR code on the smart display to the server, or fetch and show the OOB message from the server as a QR code to the smart camera.
- Transfer of OOB messages in the peer to server direction with NFC. As part of this thesis work, we have implemented an OOB channel based on NFC for transferring OOB messages in the other direction, i.e., server to peer direction.

Figure 3.3 depicts the EAP-NOOB setup where a user registers or creates an account with the web server. Figure 3.4 depicts the relationship between all the components that were discussed above. In other words, the figure illustrates a scenario in which EAP-NOOB is employed for secure bootstrapping a Display device. All the implementations of this thesis work were done on top of this initial implementation. Mudugodu Seetarama [47] describes the initial prototype implementation in detail.

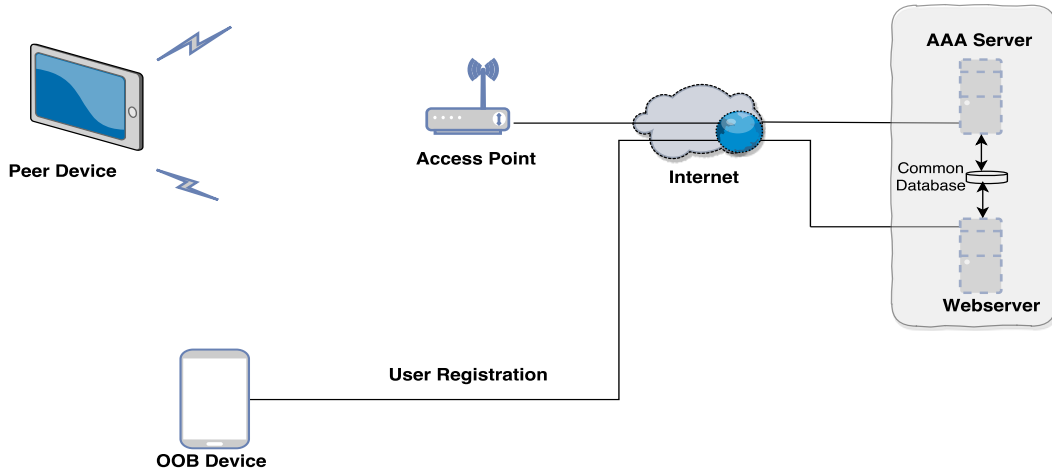


Figure 3.3: EAP-NOOB setup

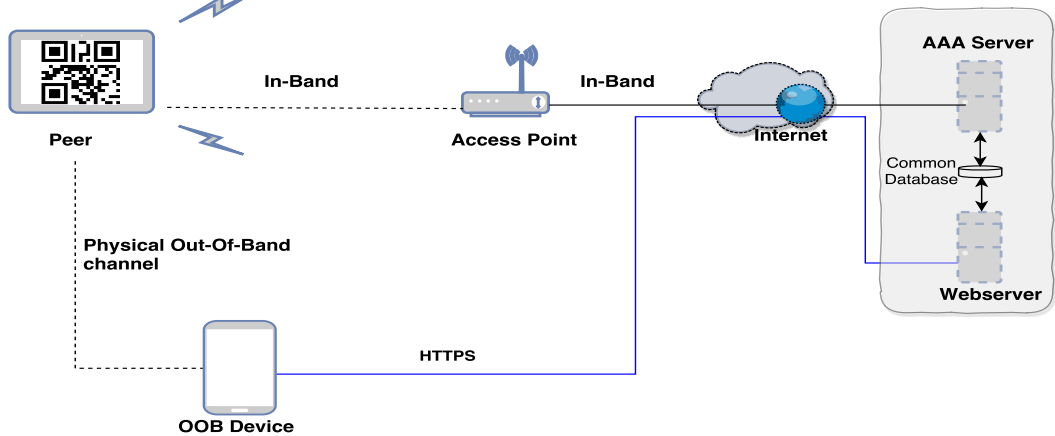


Figure 3.4: An example scenario depicting the relationship between various components of the EAP-NOOB protocol

Chapter 4

Access Control to Network Resources

This chapter begins by describing the problem that arises due to the lack of fine-grained access control on the Network Access Server (NAS), also known as the Access Point (AP), in IEEE 802.1X authenticated wireless access. Then, an appropriate solution to the problem is presented along with the implementation details.

4.1 Need for Fine-Grained Access Control

Let us understand this problem with a real life scenario. As discussed in the introduction, one assumption behind our current project is that WPA2-Enterprise networks may soon be preferred even for home environments. However, it is unlikely that the users in home environments would want to set up local Authentication Authorization Accounting (AAA) servers. Instead, they could opt to buy it as an on-demand software, i.e., Software as a Service (SaaS), from any of the cloud computing vendors such as Amazon, Google, Microsoft, etc. The two most obvious reasons to choose so would be that all the users may not be aware of technical aspects of AAA servers, and the second reason would be the cost and maintenance overhead. Some device or cloud vendors might even offer the AAA server services for free or part of some larger service packages.

Now, let us consider two neighbours: Alice and Bob. In order to securely connect their new smart home IoT appliances to the Internet and to manage them with adequate security over the Internet, they both decide to set up a Wi-Fi Enterprise network. Somehow, unknowing to each other, they both choose to purchase the AAA server service from a same cloud computing ven-

dor. Let the name of the vendor be ‘Example-Cloud’. It is implausible that the cloud computing vendors would provide individual AAA servers for each customer as the load on each server would be very low and resources would be reserved unnecessarily. Hence, rather than maintaining per-user AAA server, a typical implementation would use a single AAA server for serving several or all the users. In other words, let us assume that the remote AAA server is same both for Alice and Bob. Because of the various advantages offered by the EAP-NOOB protocol, they both decide to employ it for secure bootstrapping and then, manage their IoT devices at a central server. Consequently, they both create their respective web accounts at the same cloud vendor’s website ‘www.example-cloud.com’. These accounts would be used during the OOB steps to deliver OOB messages and also to manage the successfully bootstrapped IoT devices.

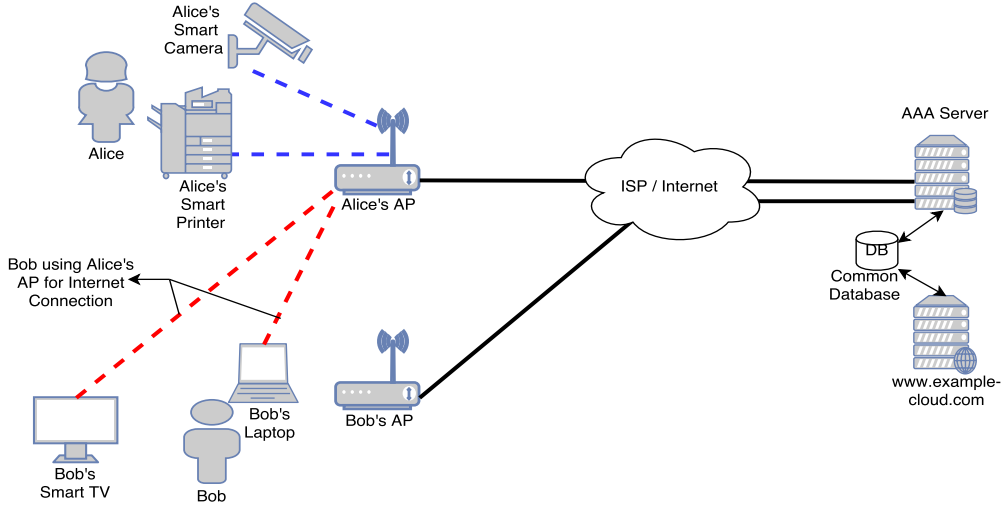


Figure 4.1: Access control problem in home AP

In this scenario, it is possible for Bob to advertently or inadvertently utilize Alice’s access point (AP) to connect his IoT devices to the Internet as he also holds a valid account in the same AAA server. In other words, Bob can use Alice’s Internet connection service for free, meaning that Bob takes a free ride for which Alice may have to pay if her Internet connection is not free. Similarly, the reverse also holds, i.e., Alice can utilize Bob’s AP to connect her IoT devices to the Internet. This problem arises due to the following reason. The server side comprising the AAA and web server that authenticates Alice and Bob is the same, and it only validates two things. First, the AAA server checks whether the NAS through which the EAP-NOOB authentication is

happening is trusted. Second, the web server checks whether the user has a valid web account. But, the server side does not implement any policies to check whether a user is authorized to attach his or her IoT devices to a given NAS or AP. In other words, the AAA server does not distinguish between the authentication requests coming via different NAS devices. Figure 4.1 depicts this problem scenario.

The above-described problem is not unique to the EAP-NOOB method. It is a common problem with any EAP authentication method when the AAA server serving multiple NASs does not implement policies to check whether an entity authenticating through a NAS is authorized to access the NAS. For instance, consider an enterprise with employees having different levels of access. Perhaps it is desirable to allow the full-time permanent employees to connect to any NAS, whereas the visiting and temporary employees may only be allowed to connect to certain NASs. This would prevent any visiting or temporary employee using the NASs intended for the full-time permanent employees, even though they have valid authentication credentials at the AAA server. Figure 4.2 illustrates the described scenario. The figure depicts two sites of an enterprise with their respective local AAA servers connected to the central AAA server of the enterprise. In site A of the enterprise, it is desirable to restrict the visiting and temporary employees to NAS-1 and NAS-2. Similarly, in site B, the visiting and temporary employees must be allowed to only connect to NAS-4 and NAS-5. However, the full-time permanent employees may have access to all the NASs at their respective site.

4.2 Proposed Solution

In order to overcome the problem, fine-grained access control based on the NAS from which the RADIUS Access-Requests originate is required. The access control can be achieved at the AAA server by using some identity information that uniquely identifies the NAS from which the Access-Requests arrives. There are two RADIUS attributes that are appropriate for obtaining the required identification information about the NAS: Called-Station-ID and NAS-ID. As defined by RFC 3580 [44], the Called-Station-ID attribute carries the MAC address of the NAS originating the Access-Request in ASCII format (upper case only), with octet values isolated by a “-” and the NAS-ID attribute carries a string identifying the NAS originating the Access-Request. Either using Called-Station-ID or NAS-ID or both, the authentication server can implement access control policies set by the administrator to decide whether a user requesting to authenticate via a NAS is authorized to use

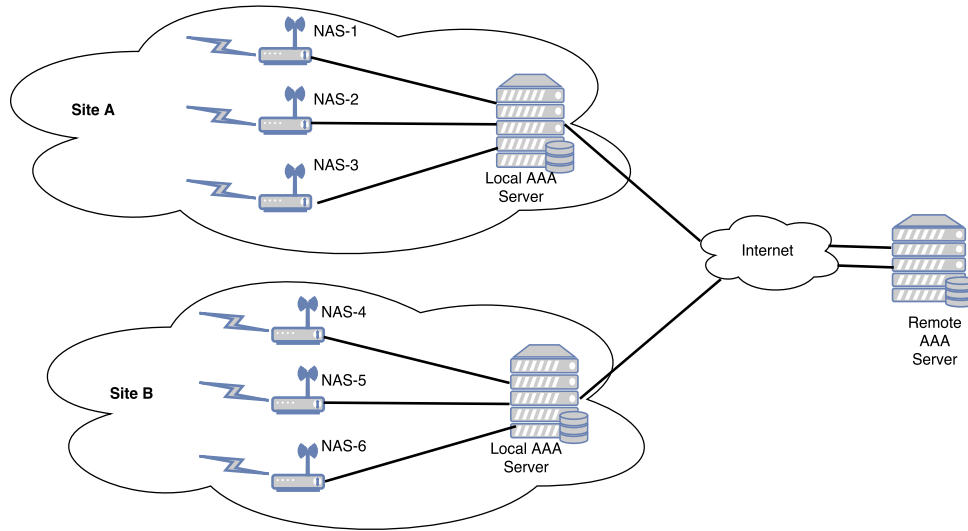


Figure 4.2: Access control problem in enterprise environments

the NAS. In other words, for each incoming authentication request from the users, the user identity is matched against the Called-Station-ID or NAS-ID attribute to decide whether the user is authorized to use the NAS based on the specified policies.

In the case of 802.1X/EAP wired authentication, another RADIUS attribute named NAS-Port-ID, which uniquely identifies each Ethernet connector in a LAN switch, can be used in conjunction with NAS-ID or Called-Station-ID or both to define similar access control policies for each of the available Ethernet ports.

In the case of 802.1X/EAP wireless authentication, the NAS-Port-ID attribute may uniquely identify each of the associations between the supplicant and the NAS. Nonetheless, most of the currently available access points populate this attribute with either a constant value or a dynamic random unique value for each of the associations. Hence, in the case of wireless authentication, the NAS-Port-ID cannot be used to define the access control policies as done in the case of wired authentication.

4.3 Implementation Environment

First, the general implementation of access control policies based on NAS is described. Then, the EAP-NOOB particular details follow, as the implemen-

tation is a bit tricky in the case of EAP-NOOB in contrast to other EAP authentication methods.

For implementing the general solution, we used FreeRADIUS software [5]. FreeRADIUS is an open source implementation of the RADIUS server with support for EAP authentication. In other words, the software is an open source AAA server implementation. The RADIUS server implemented in FreeRADIUS software supports a simple application-specific language called “unlanguage” [10], also known as Unlang in short. The objective of Unlang is to let RADIUS administrators define simple policies with minimum effort. So, based on the results of conditional checks performed by the established policies, the RADIUS request or response attributes can be updated. In other words, an administrator can write a script to perform editing of RADIUS attributes and attribute lists based on simple conditional checks. Unlang is intended only for performing specific actions on RADIUS responses and requests.

4.4 Access Control Based on Called-Station-ID and User-Name

The Called-Station-ID attribute contains the MAC address of a NAS originating the Access-Request, and it uniquely identifies the NAS device. This attribute is included in every Access-Request message sent to the AAA server by the NAS. As already described in the background section, any supplicant intending to connect to a NAS sends an EAP-Response/Identity message containing its Network Access Identifier (NAI) to the NAS. The NAS copies the NAI to a RADIUS attribute called User-Name before forwarding the received request, encapsulated in a RADIUS packet, to the AAA server. The User-Name attribute typically identifies the supplicant uniquely. So, a set of simple access control policies can be defined at the AAA server, mapping each NAS device, i.e., the Called-Station-ID attribute to their authorized supplicants, i.e., the User-Names. A simple policy file can look as shown in figure 4.3. According to the policy file 1, the only users authorized to access the NAS with Called-Station-ID ‘AB:CD:EF:GH:11:12’ are Alice and Bob. Similarly, Mallory and Alex are the only users allowed to access the NAS with Called-Station-ID ‘12:AB:34:DA:11:12’. The AAA server proceeds with the authentication process if and only if the validation against the defined policies returns true, i.e., the AAA server checks whether a mapping exists between the received User-Name and Called-Station-ID attributes. The validation against the established policies prevents supplicants from using a NAS

Called-Station-ID	Allowed-Users
AB:CD:EF:GH:11:12	alice@example.org bob@example.org
12:AB:34:DA:11:12	alex@example.org mallory@example.org

Figure 4.3: Policy file 1 based on Called-Station-ID

Called-Station-ID	Allowed-Users
AB:CD:EF:GH:11:12	*@stud.example.org *@prof.example.org
12:AB:34:DA:11:12	*@prof.example.org

Figure 4.4: Policy file 2 based on Called-Station-ID

to which they are unauthorized, even though they have valid authentication credentials at the AAA server.

As discussed in section 2.5.2, the NAI of a supplicant consists of two parts: a username part and a realm part. The realm part may identify the domains or roles to which the users belong. So, using the realm part, role-based or domain-based access control can be employed, typically in large enterprises such as universities and companies, instead of defining access control policies for individual users. In other words, instead of mapping the Called-Station-ID to a set of User-Names, it can be mapped to the various roles or domains to which the users belong. For example, consider the policy file 2 as shown in figure 4.4. All the students at the University, i.e., users with realm as ‘stud.example.org’, can only access the NAS with Called-Station-ID ‘AB:CD:EF:GH:11:12’. On the other hand, all the professors at the University, i.e., users with realm as ‘prof.example.org’, can access both the NAS devices.

4.5 Access Control Based on NAS-ID and User-Name

Similar to the above approach, the NAS-ID attribute can also be used to define the access control policies. The NAS-ID attribute contains a string identifying the NAS originating the Access-Request. For example, in our test deployment, this attribute is populated with a fully qualified domain name (FQDN) such as “fi.aalto.iot.guest.nas1” to uniquely identify each NAS. However, unlike Called-Station-ID, not all the NAS devices currently support this attribute. In other words, not all the NAS devices add this attribute in the RADIUS Access-Request messages which they send to the AAA server. Hence, in our implementation, a local RADIUS server was setup to include

this attribute in the RADIUS Access-Accept messages. At the local RADIUS server, we stored the mappings that map a Called-Station-ID to its corresponding NAS-ID in the MySQL database table. An Unlang script was written at the local RADIUS server to fetch and append the NAS-ID attribute to each Access-Request message based on the Called-Station-ID attribute present in the message. At the final remote authentication server, the validation against the policies happens in the same way as performed in the case of Called-Station-ID based access control, but the NAS-ID is used instead of Called-Station-ID to define and apply the policies. Figure 4.5 illustrates the solution. The figure is for illustrative purpose only, and it does not depict the actual message and database structures.

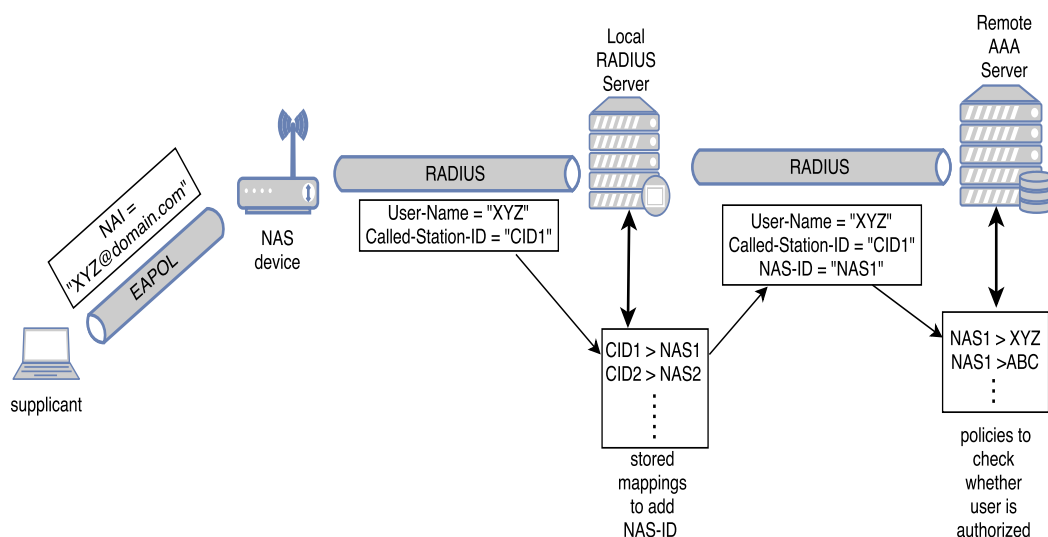


Figure 4.5: Access control based on NAS-ID attribute

One might wonder the necessity of NAS-ID based access control when Called-Station-ID is already readily available. In fact, the NAS-ID attribute offers a great benefit of flexibility in defining the access control policies. Since the NAS-ID attribute takes a string value, it can be given an FQDN value. In other words, various NAS devices in a particular deployment, such as in a large enterprise, can be grouped. The flexibility in defining access control policies is primarily due to the grouping of NAS devices. By grouping, access control policies can be set based on the various domains or groups of the NAS devices rather than the individual NAS devices. For example, a policy can be defined as shown in figure 4.6. According to the policy, all the users with realm “staff.example.org” are allowed to access all the NAS devices belonging

to the domain “nas.researchlab.example.org”.

NAS-ID	Allowed Users
*.nas.researchlab.example.org	*@staff.example.org

Figure 4.6: Policy file based on NAS-ID

Another advantage of assigning an FQDN is, in the presence of supporting infrastructure like a Dynamic Host Configuration Protocol (DHCP) and name servers, each domain and its members can be provided services based on their role or type.

An important question to be answered before moving to the next section is: *can the AAA server trust the RADIUS attributes sent by the RADIUS client?* In a home office, small Office or isolated enterprise deployments, the network administrator will be having the authority over the entire network setup including the NAS devices (RADIUS client) and AAA servers (RADIUS server). Since administrator only can configure the NAS devices and AAA servers, the RADIUS attributes received at the RADIUS server can be trusted. In other words, the NAS devices always send authorised RADIUS attributes.

On the other hand, in roaming scenarios or scenarios described in figure 4.1, the AAA server cannot trust the RADIUS client to have sent authorised RADIUS attributes. This is because the administrator at the server side is not in control of the client. It is important to note that authenticating the RADIUS clients does not guarantee that they send legitimate attributes. Hence, the AAA server could maintain mappings that associate the identity of the client to its authorized attributes. The server can use this mapping to verify whether the client has sent legitimate attributes. The client identity is its IP address or TLS identifier [49]. In RADIUS over TLS, the RADIUS attributes that are required to be authorised could be included in the client certificate.

4.6 Access Control Implementation for EAP-NOOB

Before we look into the implementation details, it is important to clarify that, for EAP-NOOB, the User-Name attribute in the RADIUS Access-Request messages contain the PeerId of a device rather than the actual username of a user. The implementation of access control based on NAS for EAP-NOOB remains the same as described in the above subsections, but with a few tweaks. In EAP-NOOB, the policy decisions are made during the OOB step at the application-layer device registration server rather than the RADIUS server. This is because the user authentication happens over the OOB channel when he or she logs-in to deliver the OOB message in either direction. Consequently, the identity of the user, such as the username, required to make access-control decisions, is available only during the OOB step. In addition to the user identity, an association between the identity of the NAS, i.e., Called-Station-ID or NAS-ID, to which the user is trying to attach his or her device and the PeerId of the device must be available to the registration server. So, when a user logs-in to the registration server account to deliver the OOB message in either direction, a check can be performed to verify whether the user is allowed to attach the devices to the given NAS. If the check fails, the user is not allowed to deliver the OOB message to the authentication server in the peer-to-server direction, and in the server-to-peer direction, the user is denied from fetching the OOB message from the authentication server. Then, the EAP-NOOB authentication process does not proceed further because the OOB step has not been completed. On the other hand, if the check succeeds, the OOB message is allowed to be delivered in either direction, and the EAP-NOOB authentication process proceeds further. This implies that any device can perform the Initial Exchange via NAS, but only the authorized users can attach the devices to the NASs.

As mentioned above, we need to associate the NAI of the supplicant device with the NAS identity. This may be done by the RADIUS server or EAP server. For EAP-NOOB, we choose to do this inside the EAP code because the RADIUS server initially only sees the generic NAI ‘noob@eap-noob.net’. However, in the current hostapd software implementation, the EAP server has access only to the EAP messages and not the RADIUS attributes. Therefore, the Called-Station-ID or NAS-ID attribute required to create the association or mapping with the PeerId is not available to the EAP server. To overcome this obstacle, we made a small change to the hostapd code. The change was to send, along with the EAP message, the RADIUS attributes: Called-Station-ID and NAS-ID from the RADIUS server to the

EAP server. Now, since the EAP-NOOB server has access to the required RADIUS attributes, it can store the mapping between the PeerId and NAS identity.

In this paragraph, a detailed explanation for not choosing the RADIUS server to store the required mappings is provided. Throughout any particular EAP session, the NASs use the NAI they received at the beginning of the EAP session via the EAP-Response/Identity message to populate the User-Name attribute in the RADIUS Access-Request messages sent to the RADIUS server. The EAP-NOOB always uses a generic NAI to start the Initial Exchange and a unique PeerId is assigned only during the Initial Exchange. In the current hostapd software implementation, the RADIUS server can only access the RADIUS attributes and not the EAP message contents. Hence, during the Initial Exchange, the RADIUS server does not have access to the unique PeerId assigned to the device. Consequently, the mapping between the PeerId and NAS identity to be used by the registration server to make access-control decisions cannot be stored during the Initial Exchange by the RADIUS server. However, the RADIUS server can store the required mappings during the next EAP session, i.e., during the Waiting Exchange phase as the NAS would have received the latest PeerId from the device via the EAP-Response/Identity message at the beginning of the EAP session. However, this forces the OOB step to happen only after at least one waiting exchange is performed, causing a significant delay in completing the EAP-NOOB authentication. To circumvent a mandatory Waiting Exchange, we chose the EAP server to store the required mappings.

The necessity for sending the RADIUS attributes from the RADIUS server to the EAP server is not unique to the EAP-NOOB method. This would also be necessary for any tunneled EAP methods such as EAP-TTLS [45] or EAP-PEAP since the RADIUS server will only see the outer anonymous identity, and the real NAI of the user is available only to the EAP server. Hence, to make the access-control decisions inside the EAP code, the RADIUS attributes must be made available to the EAP code.

4.7 Isolation of Network Clients

Before moving onto the next section, let us very briefly see how isolation or grouping of devices or users within a network can be achieved. It can be accomplished by creating a Virtual LAN (VLAN) for each group of users or devices and then, after a successful authentication, the user or device can be assigned to the appropriate VLAN group. The grouping of users or devices can be done in several ways. For example, in the case of users, grouping can

be based on the user role and in the case of devices, grouping can be based on device type. More details about the VLANs can be found at [19].

Chapter 5

Enhancements to EAP-NOOB

This chapter describes the various issues discovered during the prototype implementation and deployment of EAP-NOOB. We propose a suitable solution for each of the problems found. Then, we verify the proposed solutions by redeploying the EAP-NOOB prototype with necessary changes.

5.1 Re-keying and Algorithm Agility

In EAP-NOOB, re-keying is performed during the Reconnect Exchange phase. Figure 5.1 depicts the message sequence in the Reconnect Exchange.

The fields within square brackets are optional. Several reasons that can lead to the Reconnect Exchange are:

- A timeout has occurred for the temporal keys: Master Session Key (MSK) and Extended Master Session Key (EMSK).
- A hardware or software failure has occurred.
- The non-persistent temporal keys MSK and EMSK have been lost due to the user action such as resetting the non-persistent state of the peer.
- The supported cryptosuites at the EAP peer or server have changed.

During the implementation of the initial prototype for EAP-NOOB, we discovered the re-keying specification to be unclear. According to the initial specification, the key confirmation keys were part of the EAP-NOOB persistent association, and if the ECDH public keys were exchanged between the server and peer during the reconnect exchange, both the parties obtained their respective new key confirmation key by concatenating the previously

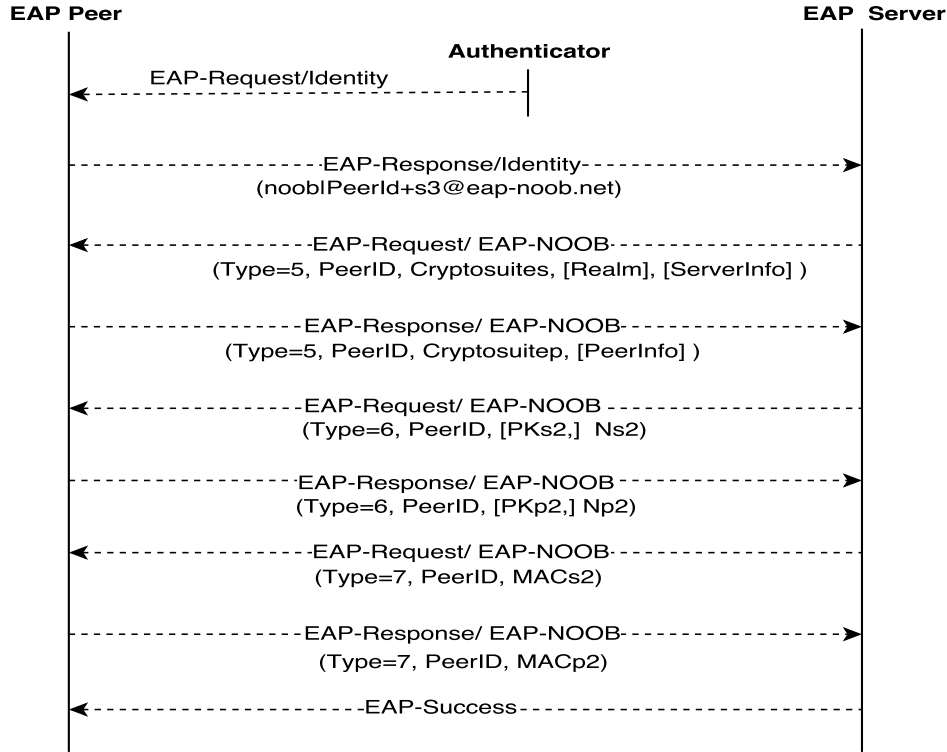


Figure 5.1: Reconnect Exchange

stored key confirmation key with certain bytes of the Key Derivation Function (KDF) output. This concatenation would increase the length of key confirmation keys after every successful Reconnect Exchange. Hence, we re-worked the re-keying, and subsequently, verified the changes by deploying the prototype after modifying it according to the new re-keying procedure. According to the new re-keying specification, the key confirmation keys are no longer part of the persistent association and certain bytes of the KDF output during the reconnect exchange form the new key confirmation keys. There are two ways of re-keying. They are:

- Re-keying with no change in the supported cryptosuite.
- Re-keying with a new cryptosuite negotiated.

5.1.1 No Change in the Supported Cryptosuite

When there is no change in the supported cryptosuite, the two parties perform the re-keying only to refresh or recalculate the temporal keys: MSK,

	KDF input field	Value
Re-keying with ECDH	Z Algorithm Id PartyUInfo PartyVInfo SuppPubInfo SuppPrivInfo	ECDH shared secret "EAP-NOOB" Np2 Ns2 (not allowed) Kz
Re-keying without ECDH	Z Algorithm Id PartyUInfo PartyVInfo SuppPubInfo SuppPrivInfo	Kz "EAP-NOOB" Np2 Ns2 (not allowed) (null)

Table 5.1: Re-keying Input [51]

EMSK and Application MSK (AMSK). The AMSK can be used for application-layer security. This re-keying can happen as a result of expired or lost temporal keys, or hardware or software failure at the peer or authenticator, due to user action. Table 5.2 shows how the Key Derivation Function (KDF) output bytes are used.

During the Reconnect Exchange phase, if the ECDH public keys were not exchanged (or if ECDH public key was sent by only one party), a special case of the re-keying happens where the computational cost of the ECDH key exchange is avoided. However, in this case, the trade-off for avoiding the computational cost is the loss of forward secrecy. When no ECDH key exchange was performed, the long-term common key Kz from the persistent association replaces the input Z. This enables re-keying without the computational cost. Table 5.1 shows the various inputs to the KDF for re-keying with and without ECDH key exchange.

5.1.2 New Cryptosuite Negotiated

When there is a change in the supported cryptosuites, the two parties re-derive their long term common key Kz. There could be several reasons for a change in the supported cryptosuites such as deprecation of cryptosuites or standardization of more efficient or more secure cryptosuites. The derivation of a new long-term common key also results in refreshing or re-calculating of the temporal keys. The usage of the output bytes is shown in table 5.2. For this type of re-keying, the ECDH public keys are always exchanged, and the

	KDF output bytes	Used as
Re-keying, no change in cryptosuite	0....63 64....127 128....191 192....223 224....255	MSK EMSK AMSK Kms2 Kmp2
Re-keying, new cryptosuite negotiated	0....63 64....127 128....191 192....223 224....255 256....287	MSK EMSK AMSK Kms2 Kmp2 Kz

Table 5.2: Re-keying output [51]

various inputs to the KDF are the same as in the case of re-keying with the ECDH key exchange, as shown in table 5.1.

5.2 Timeouts

As discussed in section 3.1, depending upon the negotiated OOB direction, either the peer or the server sends an OOB message to the other party during the user-assisted OOB step. The OOB message consists of the PeerId of the peer, a nonce value Noob and the hash value Hoob.

The initial specification of the EAP-NOOB protocol did not define any timeout for the nonce value Noob, after which the Noob expires or becomes invalid. This may open doors for attackers. An attacker may misuse an undelivered and misplaced OOB message even after a long period. For example, an OOB message printed as a QR code on a paper by a smart printer could be left lying around, and the attacker may later pick it up to connect the printer to his user account provided that the printer remains unregistered. So, to avoid the misuse of undelivered OOB messages by an adversary and thereby enhance the security, an application-dependent timeout called NoobTimeout, after which the nonce value Noob expires, was introduced. The Noob timer is configured at the sender of the OOB message. The sender may remember multiple previous OOB messages until the NoobTimeout timer for them expires. Nonetheless, the receiver only accepts the first valid OOB message delivered by the user. It will not accept another one until the following Com-

pletion Exchange has either failed or succeeded. The OOB messages sent by the server expire immediately after the NoobTimeout period, and the server will not accept the OOB messages older than that in the Completion Exchange. The peer, on the other hand, may remember the OOB messages until it has probed the server at least once even if the OOB message has expired. The peer will discard all the OOB messages that have expired after it has probed the server at least once. The peer discards the expired OOB messages irrespective of whether the probe to the server succeeded or failed. The probe may fail due to various reasons such as MAC failure during the Completion Exchange or because the peer is unable to connect to the EAP server.

Since the OOB messages expire after the NoobTimeout, the sender may output multiple OOB messages with different Noob values periodically at an implementation-dependent time interval NoobInterval. The recommended value for the NoobInterval is NoobTimeout/2 so that the sender accepts two latest OOB messages. The sender may also output different OOB messages to different user-assisted out-of-band channels or on demand when prompted by the user. For example, in the case of the server-to-peer OOB direction, multiple authenticated users may fetch multiple different OOB messages to be delivered to the same peer device.

5.3 Failure Recovery

In the initial specification of the EAP-NOOB protocol, it was specified that when the EAP peer or server detects an incorrect cryptographic fingerprint value Hoob, the recipient remains in the Waiting for OOB state as if no out-of-band message was received. This may lead to a persistent denial of service (DoS) attack if an attacker modifies any of the in-band message parameters that are used in the calculation of the Hoob. It is important to note here that all the in-band messages are communicated in plain text without any protection. In this scenario, the peer can never complete the authentication with the server until it is reset. This is because the Hoob value comparison always fails at the recipient. So, in order to recover from the failure and to thwart this attack, the recipient should limit the number of OOB messages delivered with wrong Hoob value. After an application-dependent number of invalid OOB messages have been received, the recipient should go back to the Unregistered state. This application-specific number is called OobRetries in the latest specification of EAP-NOOB.

5.4 Handling Multiple Sessions

As described in the previous section the sender can send multiple OOB messages with different Noob values. Naturally, the hash value ‘Hoob’ in each of the OOB messages also differs as the Noob value is one of the inputs to calculate the hash value. So, to help the sender choose the appropriate Noob value for deriving the keys during the Completion Exchange, an identifier called ‘NoodIb’, which uniquely identifies an OOB message, is conveyed to the sender by the receiver during the Completion Exchange. In other words, NoodIb helps the sender to handle multiple sessions with the same recipient by uniquely identifying each session. The NoobId is a hash value obtained by hashing the Noob value concatenated with a constant string “NoobId”. Figure 5.2 shows the new message sequence during the Completion Exchange.

As shown in the message sequence, the Completion Exchange now consists of one or two EAP-NOOB request-response message pairs. If the out-of-band message was sent in the peer-to-server direction, the Completion Exchange has only one request-response message pair. In the Request message, the server sends the NoobId with which the peer can identify the exact out-of-band message that the server received. On the other hand, if the OOB message was delivered in the server-to-peer direction, the Completion Exchange has two request-response message pairs. With the first request message, the EAP-NOOB server discovers the NoobId value, which determines the exact out-of-band message that the peer received.

5.5 Roaming

Roaming is a scenario where a supplicant is not in the vicinity or the coverage area of the home network. In the case of roaming, the peer device may try to connect to any of the available local networks if there is a service agreement between the home network service provider and the visited network service provider.

Authentication Authorization Accounting (AAA) architectures [33] support roaming of network-connected devices that are authenticated over EAP. While in a visited network, the authentication still happens between the peer or supplicant and an authentication server in its home network. To support such roaming in EAP-NOOB, the peer device is assigned a Realm. The server may assign the Realm to the peer during the Initial Exchange phase, and the Realm can also be assigned or re-assigned during the Reconnect Exchange phase. Once the peer has been assigned with a Realm, the visited network can route the EAP session to the home AAA server of the peer using the

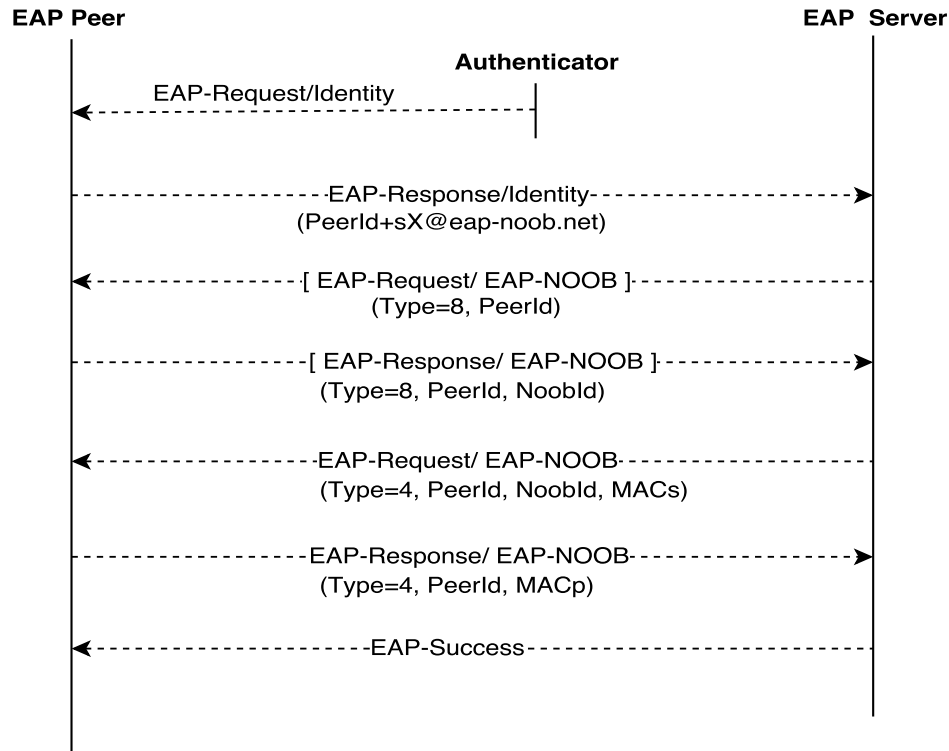


Figure 5.2: New message sequence for the Completion Exchange [51]

realm i.e. domain name part of the peer's Network Access Identifier (NAI).

Before roaming, a peer device should have established an EAP-NOOB association with its home server. After creating an EAP-NOOB association with the home server, the peer device can perform the Reconnect Exchange from the visited network. As an alternative, the peer device may also provide the user with some method to configure the home network's Realm. In that case, the peer device could establish an EAP-NOOB association with its home AAA server while roaming. For example, a camera could read a QR code shown by the user which contains the home network's Realm. The user-assigned Realm is used by the peer device in the Initial Exchange, which enables the visited network to route the EAP messages to the user's home AAA server.

Chapter 6

Additional Features

This chapter begins by describing how EAP-NOOB specification was verified to suit wired Ethernet networks. After that, the design and implementation of additional features to the EAP-NOOB prototype are explained.

6.1 IEEE 802.1X Wired Access

The 802.1X authentication framework can be employed both in wired and wireless networks. In fact, it was originally defined for IEEE 802.3 Ethernet LAN, i.e., wired LAN, and later was clarified to be suitable for other IEEE 802 LAN technologies such as 802.11 networks, i.e., wireless LAN. EAP is an application-layer protocol and is defined independently of the underlying network access technologies: Ethernet or wired access and Wi-Fi or wireless access. Naturally, the EAP-NOOB authentication method also was defined independently of the underlying network access technologies. The EAP-NOOB specification was verified to suit Wi-Fi networks through the initial prototype implementation and subsequently, testing by deploying in a testbed. However, the specification was not tested on Ethernet networks. Therefore, as part of this thesis work, we verified that the EAP-NOOB specification also works for wired devices. To test, we made necessary changes to the prototype and then, deployed it in an Ethernet network. Changes were required only at the supplicant side, i.e., the *wpa_supplicant* code, as the EAP server operations are independent of the network access technology used by the supplicants. A detail explanation of the changes made to the *wpa_supplicant* code follows.

Before we look into the details of the modifications done to the prototype to support wired authentication, it is imperative to understand the *wpa_supplicant* architecture. Since *wpa_supplicant* is built to support both

secure and open wireless connection establishment, we have only considered the part that involves 802.1X components. Figure 6.1 depicts the architecture. As shown in the figure, each layer maintains its state machine during the 802.1X-based authentication process to keep track of the message transfers and also the authentication results. Each EAP authentication method is an individual module implemented over the EAP state machine. The EAP state machine, in turn, operates on top of the EAPOL layer state machine. Sections 2.5.2 and 2.5.3 in the background chapter discuss the functionality of the EAP and EAPOL layers respectively.

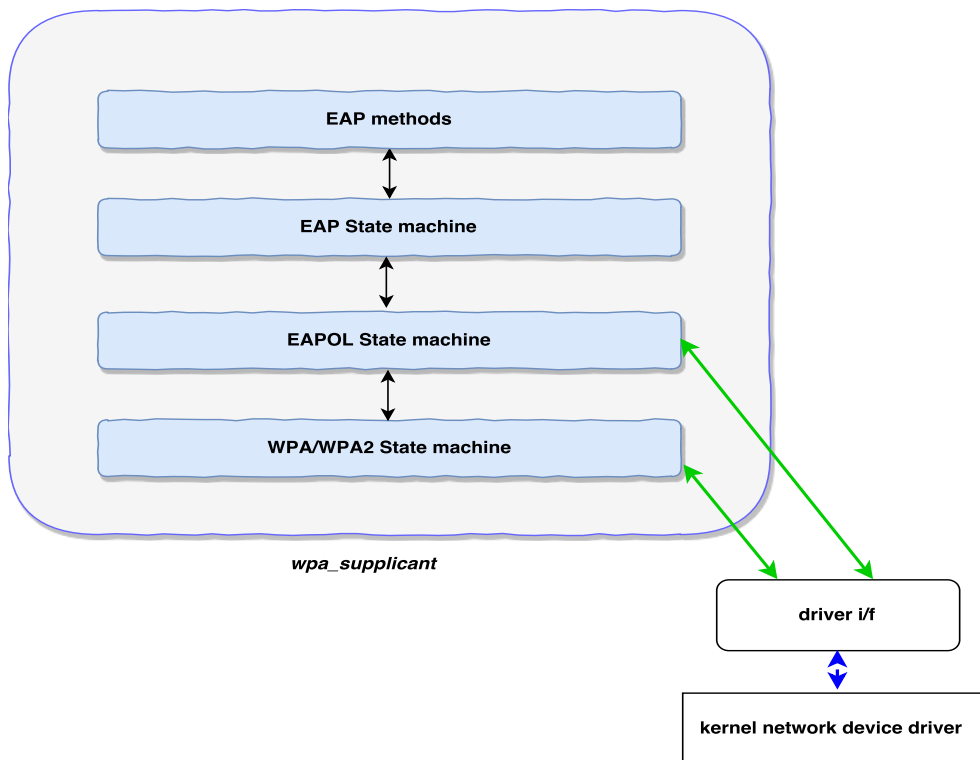


Figure 6.1: *wpa_supplicant* architecture [14]

In the case of Wi-Fi networks, the EAP-NOOB peer tries to authenticate with all the reachable SSIDs that support the EAP-NOOB method before successfully authenticating with the AP that the user chooses and authorizes. Hence, the EAP-NOOB peer code keeps track of the various ongoing authentication processes and associated contexts on a per-SSID basis. Here, the intermediate EAP-NOOB authentication parameters such as nonces, ECDH keys, etc. are collectively called the context. On the other hand, in the case

of IEEE 802.1x wired network authentication, the supplicant only authenticates with the NAS to which it is connected via the Ethernet cable. In other words, the EAP-NOOB peer needs to maintain only one context for the network interface. This implies that the EAP-NOOB peer implementation must distinguish between wired and wireless access. Since the initial implementation of the EAP-NOOB supplicant was aimed at Wi-Fi networks, there was no distinction between the wireless and wired authentication. So, to support wired authentication, the first change we made to the prototype was to distinguish between the wired and wireless access and handle the authentication process accordingly. The distinction was made based on the driver interface specified in the configuration file for *wpa_supplicant*, i.e., wired or wireless driver, with which the *wpa_supplicant* was started. This is depicted in figure 6.1. In *wpa_supplicant*, the EAPOL layer stores the driver name as a string in a variable, and the variable is also accessible to the EAP methods. Hence, we used this variable to make the distinction between wired and wireless access.

In EAP-NOOB, it may take several minutes for the OOB message to be delivered. The peers may unnecessarily probe the server several times before the OOB message is delivered. Hence, the server may assign a wait time to the peers during the Initial and Waiting Exchanges to limit the rate at which they probe the server. If a wait time was assigned, the peer should not probe the server until it elapses. As explained by Mudugodu Seetarama [47], the waiting functionality for 802.1X-based wireless access was implemented using a provision in *wpa_supplicant* to disable association tries to an SSID temporarily. In other words, the *wpa_Supplicant* temporarily abstains from sending the probe requests, the message two of Stage 1 in RSNA establishment procedure 2.5.6, to an SSID. *wpa_supplicant* provides this functionality to avoid contacting SSIDs which recently returned an EAP-Failure. Instead, the supplicant can attempt to associate with other reachable SSIDs. On the other hand, for 802.1X-based wired access, *wpa_supplicant* currently has no such provisions. It could be because an Ethernet port on the supplicant device can be connected to only one NAS at a time. The requirement for the supplicant to wait for a random period, assigned by the EAP server, before again probing the server is a new requirement introduced by the EAP-NOOB method. So, we had to implement the waiting feature from scratch for 802.1X-based wired access.

In the case of wired access, the peer probes only one NAS. Hence, it is important for the peer to wait for the wait time to elapse before it probes the server again. Otherwise, if the peer neglects to wait and keeps probing the server repeatedly before the OOB message is delivered, it might quickly reach the maximum number of Waiting Exchanges allowed by the server. Once the maximum number of Waiting Exchanges are reached, the peer will

receive the Unwanted-Peer error from the server.

As explained in section 2.5.3, the EAP authentication is triggered either by the supplicant via an EAPOL-start message or the NAS can send an EAP-Request Identity message before it receives the EAPOL-Start message. To introduce wait time before triggering the EAP authentication, we added two features in the EAPOL layer. One was to prevent the EAPOL layer from sending the EAPOL-Start message until the wait time elapsed. Another was to silently discard the EAP-Request Identity message if received from the NAS before the wait time elapsed. With these features we ensured that the peer device waits until the wait time assigned by the server has elapsed. To communicate the waiting time to the EAPOL layer from the EAP-NOOB peer code, we added a new integer variable to the EAP state machine context which is accessible to the EAPOL layer. The EAP-NOOB peer code populates this variable with the epoch ¹time until which the server should not be probed.

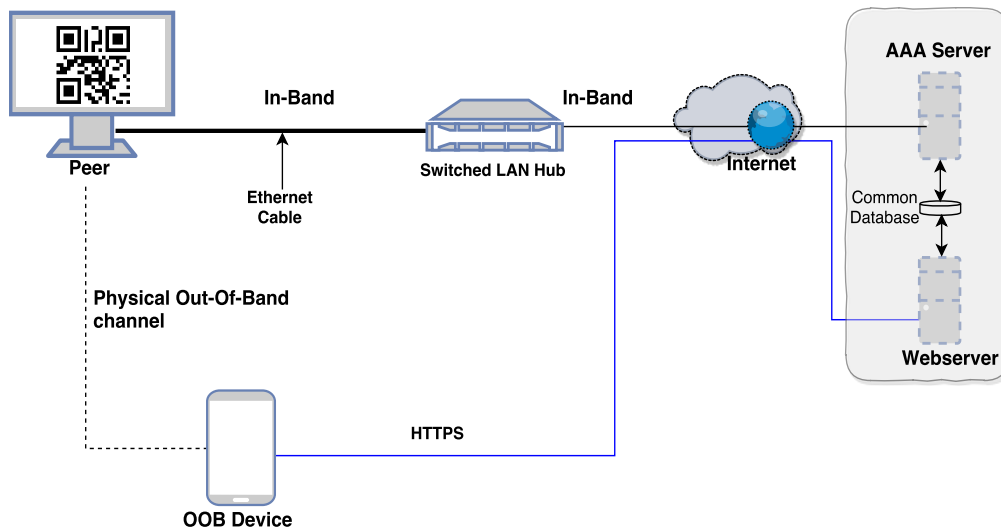


Figure 6.2: 802.1X-based wired access

With the changes mentioned above, the EAP-NOOB prototype was deployed for authenticating supplicant devices in 802.1X-based wired access network. The deployment scenario is shown in figure 6.2. The only difference, when compared to 802.1X-based wireless access, is that a LAN switch is used instead of a wireless access point. We were able to successfully authenticate and attach the supplicant devices to the wired network, and hence, it is

¹the number of seconds that have elapsed since January 1, 1970

verified that EAP-NOOB specification also suits IEEE 802.1X wired access.

6.2 OOB Channel with NFC

As discussed in section 3.3, in our implementation, smartphones were extensively used for automating the OOB message transfers between the server and peer. In this thesis work, we implemented another new method for transferring the OOB messages in the server-to-peer direction with NFC. Unlike the cameras, the browsers on smartphones currently do not have support for accessing NFC from a web application. Hence, it was not possible to build an entirely platform-independent application to transfer OOB messages in the server to peer direction with NFC. Instead, we used Android WebView, an Android system component powered by Chrome, with which applications can display web content [12] while also having access to native Android APIs. This way, the web application can interface with the NFC feature in the Android phone. Before looking at the implementation details, let us briefly review some important concepts of Android application development related to our implementation.

- Shared Preferences [11] - It is one of the ways provided by Android through which an application can save and retrieve its data in the form of a key-value pair.
- JavaScript Interface [2] - It is a functionality with which an interface can be created between the JavaScript code and the client-side Android code. In other words, using JavaScript Interface, JavaScript code on a web page can call a method in the client-side Android code.
- Host-based card emulation [6] - Host Card Emulation (HCE) is an on-device technology that allows a smartphone to emulate an NFC card on an NFC-enabled device. Host-based card emulation is an implementation of HCE technology available on Android phones for the developers to emulate an NFC card.

Figure 6.3 depicts the architecture of the Android application that we developed using the Android WebView. First, the user logs in to the EAP-NOOB web application via this application. Once the OOB message to be delivered to a peer device is ready, the user clicks ‘NFC send’ button. When the ‘NFC send’ button is clicked, the EAP-NOOB web application calls a native Android function ‘sendOOB’ with the OOB message as the function parameter through the Android JavaScript Interface. The Android function,

in turn, stores the received OOB message in the sharedpreferences in the form of a key-value pair. Once the OOB message is successfully stored in the key-value store, the user is prompted to hold the phone close to the NFC reader of the peer device for which the OOB message has to be delivered. An NFC card with the OOB message stored in the key-value store is emulated by the host-based card emulation service running on the Android smartphone. The card emulation service retrieves the OOB message stored in the key-value store via the same key with which it was stored. The NFC reader of the peer device reads the emulated NFC card to get the OOB message and then continues the EAP-NOOB authentication process.

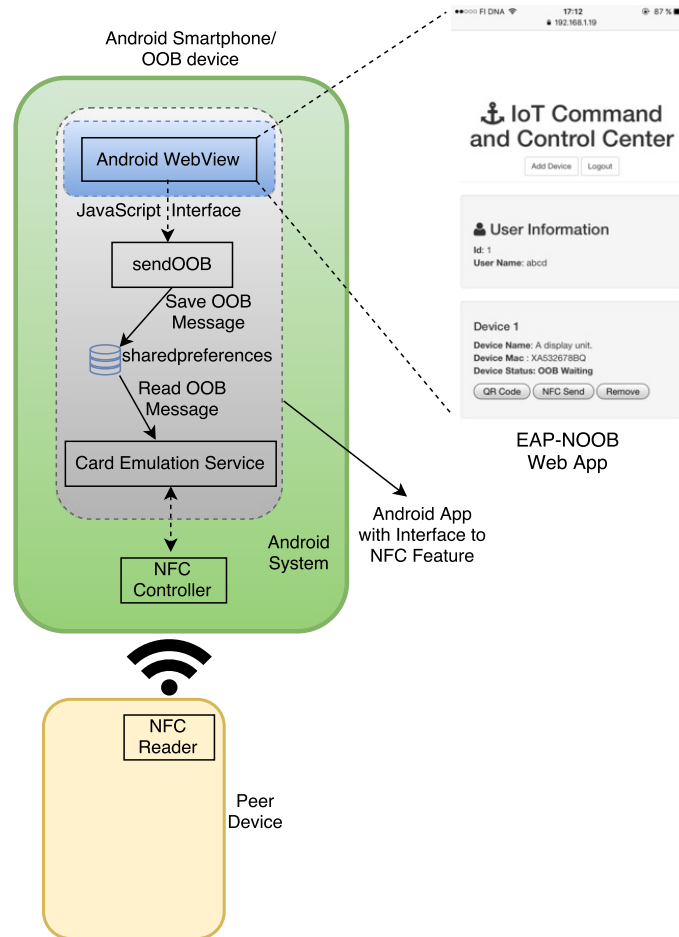


Figure 6.3: OOB message transfer with NFC

6.3 Web User-Interface to Configure the RADIUS Clients

Currently, the RADIUS server code inside the *hostapd* reads the RADIUS client configuration file 'hostapd.radius_clients' only once, at the beginning when *hostapd* is started. After starting *hostapd*, if the RADIUS clients configuration file is modified, the *hostapd* does not re-read the configuration file itself. The *hostapd* needs to be restarted to trigger the re-read. Restarting *hostapd* just to re-read the RADIUS client configuration file is not efficient. Hence, to make *hostapd* re-read the configuration file on the fly without restarting, and also to allow remote modification of the file, we designed and implemented a web functionality as part of the EAP-NOOB web application. The proposed model can be easily extended to any AAA server implementations deployed in real world scenarios. Figure 6.4 illustrates the implementation setup.

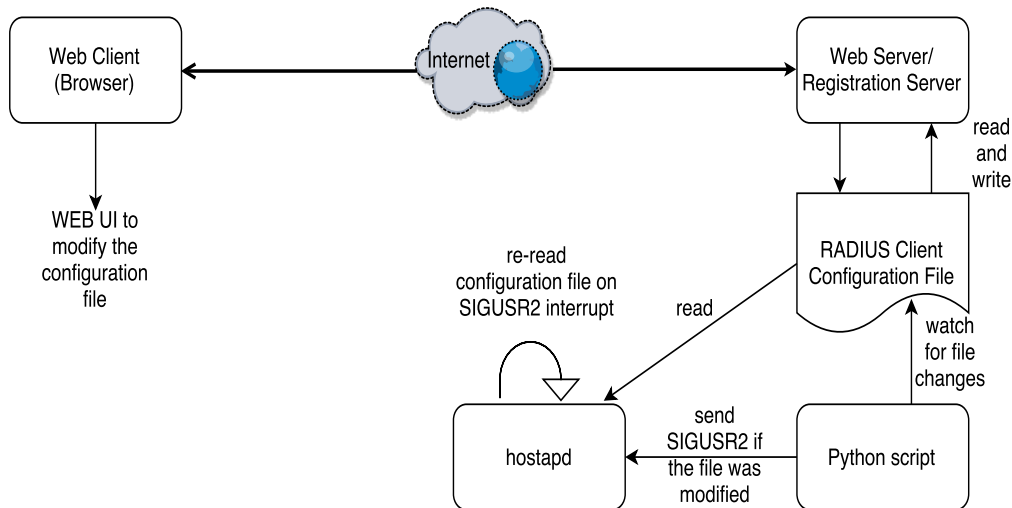


Figure 6.4: Web UI to configure RADIUS clients

To make *hostapd* re-read the configuration file on modifications without restarting, we made use of Linux Interrupts and inotify. Inotify or inode notify is part of the Linux kernel subsystem with which applications can watch certain files and get notified when the files are deleted, modified, renamed or opened [9]. Firstly, we added code to register for a Linux user-space interrupt signal 'SIGUSR2' inside the source code of the *hostapd* RADIUS server. Secondly, as a response to the SIGUSR2 interrupt signal,

an interrupt service routine (ISR) was written to re-read the RADIUS client configuration file. A Python script was written to watch for modifications to the configuration file with the inotify API. As soon as the script gets notified of any modifications to the configuration file, it sends the SIGUSR2 interrupt signal to *hostapd*, and the *hostapd* re-reads the configuration file upon receiving the interrupt signal.

Chapter 7

Discussion

This chapter begins by presenting a summary and contributions of this thesis work. After that, few open questions on EAP-NOOB are discussed.

7.1 Thesis Contributions

Firstly, the issue of fine-grained access control to network resources was addressed in chapter 4. The access control was implemented at the AAA server by associating the user identity with the identification information that uniquely identifies the Network Access Server (NAS) from which the Access-Requests are generated. We decided to address this issue first because it was a common problem associated with all EAP authentication methods including the EAP-NOOB method. Through implementing a model of the proposed solution, we have clarified that the proposed solution can be employed to achieve access control based on the NAS irrespective of the EAP method used. Also, while implementing the solution, we identified that there might be a necessity of sending certain RADIUS attributes from the RADIUS server to the EAP server, which is a different layer of software in the server, to help the EAP server in making access control policy decisions based on the NAS identity. This necessity arises when the proper user identity is only available to the EAP server as in the case of EAP-NOOB and tunneled EAP methods such as EAP-TTLS or EAP-PEAP. We have also explained how to send the RADIUS attributes from the RADIUS server code to the EAP server code.

Secondly, several issues that are specific to EAP-NOOB were addressed in chapter 5. At first, we took up the issue of re-keying as it was a flaw in the specification. Following the original specification, the size of key confirmation keys would have grown after every Reconnect Exchange phase. The issue was

primarily because the key confirmation keys were part of the EAP-NOOB persistent association, and during the Reconnect Exchange phase, new key confirmation keys were derived by concatenating the previously stored key confirmation keys with certain bytes of the KDF output. To fix this issue, we no longer include the key confirmation keys as part of persistent EAP-NOOB association. The new re-keying procedure is as described in section 5.1. As a result of the new re-keying procedure, the memory size required to store EAP-NOOB persistent association is also reduced. Then, as discussed in section 5.2, a potential bug in the EAP-NOOB protocol where an attacker could cause a persistent denial of service attack by just modifying any of the in-band messages was fixed by limiting the number of tries to deliver invalid OOB messages at the receiver side. Also, an application-dependent timeout value after which the nonce Noob expires was introduced to prevent the misuse of undelivered OOB messages and, consequently, to enhance the protocol security. Since the Noob values expire, the sender may generate multiple OOB messages with different Noob before the start of the Completion Exchange. Hence, as described in section 5.4, to help the sender choose the appropriate Noob value for deriving the keys during the Completion Exchange, an identity field called NoodIb, which uniquely identifies an OOB message, has been introduced in the Completion Exchange. Finally, as discussed in section 5.5, to support roaming in EAP-NOOB, the peer device is now assigned with a Realm by the server. Once the peer has been assigned with a Realm, the visited network can route the EAP session to the home AAA server of the peer using the domain name part of the peer's Network Access Identifier (NAI).

Apart from fixing the issues in EAP-NOOB protocol, as described in chapter 6, we have also designed and implemented additional features for EAP-NOOB to enhance the user-experience. Firstly, we have added support for IEEE 802.1X-based wired access in the EAP-NOOB implementation. Through this, we also confirmed that the EAP-NOOB specification also works for wired devices. Then, we have designed and implemented a new OOB channel based on NFC to transfer the OOB messages in the server-to-peer direction. Finally, as discussed in section 6.3, we have designed and implemented a web application to modify the RADIUS clients configuration file. As part of this web application design and implementation, we have shown how a RADIUS server can re-read the configuration file, if modified, on the fly without having to restart.

7.2 Open Questions

From all the discussions we had, it is evident that EAP-NOOB is a secure open standard and generic protocol for secure bootstrapping of smart IoT devices in IEEE 802.1X-based access networks. However, a remaining challenge for EAP-NOOB in the case of IEEE 802.1X wireless access is the authentication latency [47]. In other words, in Wi-Fi networks, the EAP-NOOB authentication process may be time-consuming. This is mainly because a peer device has to sequentially try to associate with all the available SSIDs that support WPA2-Enterprise and attempt to perform the Initial Exchange. This process adds a significant delay to the authentication process. To reduce the authentication latency, we propose to parallelize this initial process, i.e., a peer device can attempt to perform the Initial Exchange via various available SSIDs in parallel and only complete the authentication with the SSID for which the user delivers the OOB message. This might require some modifications to the current implementations of the WPA/WPA2 supplicant, but no changes to the EAP-NOOB protocol are required. Alternatively, the EAP-NOOB peer device may provide some method for the user to configure the SSID to which the device has to connect. In that case, the peer device can use the user-configured SSID to perform the EAP-NOOB authentication. This would significantly reduce the authentication latency as the peer device has to deal with only one SSID rather than several of them.

Another improvement that could be made is in the software architecture proposed in figure 6.4. Instead of watching for changes to the RADIUS client configuration file from the Python script, the source code of the *hostapd* RADIUS server could use C APIs to watch for changes to the configuration file. This approach would simplify the architecture as watching and signalling from the Python script would be avoided.

7.3 Standards and Implementation Status

Based on the experiences gained from this thesis work, the EAP-NOOB protocol specification has been improved. The latest draft specification has been published on the Internet and can be found at [51]. The protocol has been proposed for standardisation at the IETF. As there is always scope for the improvement, it cannot be claimed that the specification of the protocol is complete. We will continue to analyse and improve the protocol to make it more secure and user-friendly.

A working prototype of EAP-NOOB and the instructions for installing and using it can be found at [8]. The implementation of the prototype is

according to the latest draft specification of the EAP-NOOB protocol. We are working to improve its performance and user experience.

Chapter 8

Conclusion

As the prices of electronic components such as memory and processors are getting cheaper, devices and appliances of our everyday life are becoming more intelligent and productive. Smart home devices from major vendors are already available in the market. Soon these smart IoT devices are expected to proliferate both in our work and household environments, and consequently, influence our daily life. Since physical objects are connected to the Internet, the ramifications of device compromise are no longer limited to some monetary loss but can also be to the extent of causing physical harm to humans. Moreover, the methods for bootstrapping and managing the devices securely have to be usable and scalable enough to be repeatedly applied to thousands or more devices. Hence, it will be wise to investigate and be prepared with robust security solutions to make our work and home environments safe and secure. In this thesis work, we have implemented, tested and enhanced one such solution, the Nimble out-of-band authentication for EAP (EAP-NOOB), which performs the secure bootstrapping of IoT appliances intending to connect to the Internet via IEEE 802.1X-based access networks.

More specifically, we have produced an updated prototype implementation of EAP-NOOB and successfully addressed the various issues of the EAP-NOOB protocol that were discovered during the implementation of its initial prototype. An appropriate solution to each of the problems identified was provided along with the implementation details. Also, new additional features were designed and implemented to enhance the user-experience of EAP-NOOB. This thesis work has contributed in numerous ways to enhancing the EAP-NOOB protocol specification, and the latest version of the EAP-NOOB Internet-Draft can be found at [51]. The EAP-NOOB protocol is currently proposed for standardization with the Internet Engineering Task Force (IETF), and gathering implementation experiences is a crucial part of the specification and standards process.

Bibliography

- [1] 3GPP TS 33.220 version 13.0.0 Release 13. <http://www.3gpp.org/DynaReport/33220.htm>. Generic authentication architecture (GAA); generic bootstrapping architecture (GBA).
- [2] Building Web Apps in WebView — Android Developers. <https://developer.android.com/guide/webapps/webview.html>. (Accessed on 05/22/2017).
- [3] Definition of bootstrap in English, "bootstrap". <https://en.oxforddictionaries.com/definition/bootstrap>. (Accessed on 04/26/2017).
- [4] Elliptic Curve Cryptography - OpenSSLWiki. https://wiki.openssl.org/index.php/Elliptic_Curve_Cryptography. (Accessed on 04/22/2017).
- [5] FreeRADIUS: The world's most popular RADIUS Server. <http://freeradius.org/>. (Accessed on 05/05/2017).
- [6] Host-based Card Emulation — Android Developers. <https://developer.android.com/guide/topics/connectivity/nfc/hce.html#HCE>. (Accessed on 05/22/2017).
- [7] hostapd and wpa_supplicant. <https://w1.fi/>. (Accessed on 05/08/2017).
- [8] Implementation of Nimble out-of-band authentication for EAP (EAP-NOOB). <https://github.com/tuomaura/eap-noob>. (Accessed on 07/30/2017).
- [9] inotify(7) - Linux manual page. <http://man7.org/linux/man-pages/man7/inotify.7.html>. (Accessed on 05/22/2017).
- [10] Manpage of unlang. <http://freeradius.org/radiusd/man/unlang.html>. (Accessed on 05/05/2017).

- [11] SharedPreferences — Android Developers. <https://developer.android.com/reference/android/content/SharedPreferences.html>. (Accessed on 05/22/2017).
- [12] WebView — Android Developers. <https://developer.android.com/reference/android/webkit/WebView.html>. (Accessed on 05/22/2017).
- [13] Who We Are — Wi-Fi Alliance. <http://www.wi-fi.org/who-we-are>. (Accessed on 04/24/2017).
- [14] wpa_supplicant / hostapd: Developers' documentation for wpa_supplicant and hostapd. https://w1.fi/wpa_supplicant/devel/. (Accessed on 05/19/2017).
- [15] IEEE SA - 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. Specification, 2004.
- [16] Wi-Fi Alliance. Wi-Fi protected setup. Specification, 2007.
- [17] IEEE Std 802.11-2012 (Rev. of IEEE Std 802.11-2007). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Specification, 2012.
- [18] IEEE std. 802.1ar-2009, Standard for local and metropolitan area networks - secure device identity, December, 2009.
- [19] Institute of Electrical and Electronics Engineers, Virtual Bridged Local Area Networks. IEEE Standard 802.1Q, May 2006.
- [20] ALAN DEKOK. DTLS as a Transport Layer for RADIUS. Internet-Draft draft-dekok-radext-dtls-03, Internet Engineering Task Force, 2010. Work in Progress.
- [21] B. ABOBA, L. J. BLUNK, J. R. VOLLBRECHT, J. CARLSON AND H. LEVKOWETZ. Extensible authentication protocol (EAP). IETF. RFC 3748, June, 2004.
- [22] B. ABOBA, M. BEADLES, J. ARKKO AND P. ERONEN. The Network Access Identifier. IETF. RFC 4282, December, 2005.

- [23] B.GROZA AND R.MAYRHOFFER. Saphe: Simple accelerometer based wireless pairing with heuristic trees. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia* (2012), ACM.
- [24] CERTICOM RESEARCH. Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography. Version 2.0, May 21, 2009.
- [25] C.NEUMAN, T.YU, S. HARTMAN AND K. RAEBURN. The Kerberos network authentication service (V5). IETF. RFC 4120, June, 2004.
- [26] C.RIGNEY, S.WILLENS, A.RUBENS AND W.SIMPSON. Remote dial-in user authentication service (RADIUS). IETF. RFC 2865, June, 2000.
- [27] C.RIGNEY, W.WILLATS AND P.CALHOUN. RADIUS Extensions. IETF. RFC 2869, June, 2003.
- [28] D.SIMON, B.ABOBA AND R.HURST. The EAP-TLS authentication protocol. IETF. RFC 5216, March, 2008.
- [29] D.WHITING, R.HOUSLEY AND N.FERGUSON. Counter with CBC-MAC (CCM). IETF. RFC 3610, September, 2003.
- [30] E. BARKER, L. CHEN, A. ROGINSKY AND M. SMID. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. In *Technical Report; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2006. 2012* (2006), Cite-seer.
- [31] H. KRAWCZYK, M. BELLARE AND R. CANETTI. HMAC: Keyed-hashing for message authentication. IETF. RFC 2104, February, 1997.
- [32] HASSANALIERAGH, M.PAGE, ET AL. Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges. In *2015 IEEE International Conference on Services Computing (SCC)*.
- [33] J. VOLLBRECHT, P. CALHOUN, S. FARRELL, L. GOMMANS, G. GROSS, B. DE BRUIJN, C. DE LAAT, M. HOLDREGE AND D. SPENCE. AAA Authorization Framework. IETF. RFC 2904, August, 2000.
- [34] J.C.MITCHELL AND C.HE . Security Analysis and Improvements for IEEE 802.11 i. In *The 12th Annual Network and Distributed System*

- Security Symposium (NDSS'05) Stanford University, Stanford, Citeseer, 2005.*
- [35] L.FOSCHINI, T.TALEB, A.CORRADI AND D.BOTTAZZI. M2M-based metropolitan platform for IMS-enabled road traffic management in IoT. *IEEE Communications Magazine* (2011).
 - [36] M.ANTIKAINEN, M.SETHI, S.MATETIC AND T.AURA. Commitment-based device-pairing protocol with synchronized drawings and comparison metrics. *Pervasive and Mobile Computing* (2015).
 - [37] M.SETHI. Security in smart object networks. <http://urn.fi/URN:NBN:fi:aalto-201210313327>, 2012. Master's Thesis, Aalto University.
 - [38] M.SETHI. Security for Ubiquitous Internet-Connected Smart Objects. <https://aaltodoc.aalto.fi/bitstream/handle/123456789/23874/isbn9789526072241.pdf>, 2016. Doctoral Thesis, Aalto University.
 - [39] M.SETHI, E.OAT, M.D. FRANCESCO AND T.AURA. Secure bootstrapping of cloud-managed ubiquitous displays. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2014), ACM.
 - [40] M.SETHI, M.ANTIKAINEN AND T.AURA. Commitment-based device pairing with synchronized drawing. In *International Conference on Pervasive Computing and Communications (PerCom)* (2014), IEEE.
 - [41] M.SETHI, P.KORTOCI, M.D FRANCESCO AND T.AURA. Secure and low-power authentication for resource-constrained devices. In *5th International Conference on the Internet of Things (IOT)* (2015), IEEE.
 - [42] N.BORISOV, I.GOLDBERG AND D.WAGNER. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the 7th Annual International Conference on Mobile computing and Networking, ACM* (2001).
 - [43] N.SHANKAR, J.WANG AND W.A.ARBAUGH. Your 802.11 Network has no Clothes. In *Proceedings of the First IEEE International Conference on Wireless LANs and Home Networks* (2001).
 - [44] P. CONGDON, B. ABOBA, A. SMITH, G. ZORN AND J. ROESE. IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines. IETF. RFC 3580, September, 2003.

- [45] P. FUNK AND S. BLAKE-WILSON. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). IETF. RFC 5281, August, 2008.
- [46] P.S.HENRY AND H.LUO. WiFi: what's next? *IEEE Communications Magazine*, December 2002.
- [47] R M SEETARAMA. Secure Device Bootstrapping with the Nimble Out of Band Authentication Protocol. <http://urn.fi/URN:NBN:fi:aalto-201706135412>, 2017. Master's Thesis, Aalto University.
- [48] S. MIRZADEH, H. CRUICKSHANK AND R. TAFAZOLLI. Secure Device Pairing: A Survey. *IEEE Communications Surveys and Tutorials* (2014).
- [49] S. WINTER, M. MCCAULEY, S. VENAAS AND K. WIERENGA. Transport Layer Security (TLS) Encryption for RADIUS, May, 2012.
- [50] T.AURA AND M.SETHI. Nimble out-of-band authentication for EAP (EAP-NOOB). Internet-Draft draft-aura-eap-noob-01, Internet Engineering Task Force, 2016. Work in Progress.
- [51] T.AURA AND M.SETHI. Nimble out-of-band authentication for EAP (EAP-NOOB). Internet-Draft draft-aura-eap-noob-02, Internet Engineering Task Force, May 2017. Work in Progress.
- [52] T.DIERKS AND E.RESCORLA. The Transport Layer Security (TLS) Protocol Version 1.2. IETF. RFC 5246, August, 2008.
- [53] V. FAJARDO, J. ARKKO, J. LOUGHNEY AND G. ZORN . DIAMETER Base Protocol, October, 2012.
- [54] WANG, M.ZHANG, ET AL. An IoT-based appliance control system for smart homes. In *2013 IEEE International Conference on Intelligent Control and Information Processing (ICICIP)*.
- [55] W.DIFFIE AND M.HELLMAN. New directions in cryptography. *IEEE transactions on Information Theory* (1976).
- [56] W.SHEN, B.YIN, X.CAO, L.CAI X AND Y.CHENG. Secure device-to-device communications over WiFi direct. *IEEE Network* (2016).
- [57] W.WEI, B.WANG, C.ZHANG, J.KUROSE AND D.TOWSLEY. Classification of access network types: Ethernet, wireless LAN, ADSL, cable modem or dialup. *Computer Networks*. Elsevier, 2008.

- [58] YLONEN, T., AND LONVICK, C. The secure shell (SSH) transport layer protocol. IETF. RFC 4253, January 2006.