Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Kirsten Lunde Skaug
Elise Breivik Smebye

Keeping Connected in Internet-Isolated Locations

Master's thesis in Communication Technology
Supervisor: Yuming Jiang
Co-supervisor: Besmir Tola

June 2021

NTNU
Norwegian University of Science and Technology

Kirsten Lunde Skaug
Elise Breivik Smebye

Keeping Connected in Internet-Isolated Locations

Master's thesis in Communication Technology
Supervisor: Yuming Jiang
Co-supervisor: Besmir Tola
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title:	Keeping Connected in Internet-Isolated Locations
Students:	Kirsten Lunde Skaug
	Elise Breivik Smebye

Problem description:

Smartphones are essential tools for everyday tasks and are widely used for a multitude of communication services. Services such as video streaming, content sharing or instant messaging require users to have an internet connection in order to connect to a central server. However, in out-of-coverage areas such as the mountains or in areas where the infrastructure is damaged, phones are unable to establish a connection to the server. As a consequence, these services become unavailable. This can be critical during natural disasters and rescue operations where user's connectivity can make a significant difference.

In out-of-coverage areas, smartphones equipped with Wi-Fi radio may make use of the radio to establish a Peer-to-Peer (P2P) network with other users in the vicinity. This would then enable users to communicate through the established network. However, unlike ordinary Wi-Fi settings that use a dedicated Wi-Fi base station, in a P2P setting every device has the opportunity to assume this master role and the communication has to be adjusted accordingly. To achieve this, other Wi-Fi technologies, such as Wi-Fi Direct and Wi-Fi Aware, are needed.

The P2P approach also presents new challenges in several aspects of security due to a lack of connection to a central server. In particular, the authentication of users is essential to ensure the identity of the communicating parties. Besides, users want to make sure they are not talking to an imposter and that their messages are not read or tampered with in the process. Thus, the main challenges are how to provide authentication of users and how to ensure the integrity and confidentiality of messages in internet-isolated P2P network.

The goal of the master thesis is to design, implement and validate a solution for instant messaging when there is no internet connection. The proposed solution will enable secure and mutually authenticated communication among a group of devices in the vicinity. While Wi-Fi Aware will be used as the communication technology, an investigation that compares our solution with other related solutions will also be performed. In order to validate the solution experimentally, a proof-of-concept instant messaging application will be developed.

Date approved:	2021-01-13
Supervisor:	Yuming Jiang, IIK
Cosupervisor:	Besmir Tola, IIK

Abstract

In areas affected by natural disasters, at crowded concerts, and in rural areas, mobile social networks requiring an Internet connection might not be available. Without connectivity, users become isolated and unable to communicate. Open-source solutions are available that provide connectivity and enable instant messaging between devices. However, the solutions are mainly limited to Wi-Fi Direct and Bluetooth technologies. Wi-Fi Aware promises interesting properties that address the shortcomings of the centralized architecture of Wi-Fi Direct and the short range and of Bluetooth. However, it has received little research and the full potential of Wi-Fi Aware is still to be explored. The aim of this master thesis is to design, propose, and validate a solution that enables connectivity between mobile devices in the vicinity and provides authenticated and secure instant messaging between users in internet isolated locations.

The proposed solution uses Wi-Fi Aware for establishing a Peer-to-Peer (P2P) network and the mutual Transport Layer Security (mTLS) protocol to provide mutually authenticated and encrypted message transfer. Key contributions provided by this project comprises research into Wi-Fi Aware as a connectivity technology for instant messaging together with the Transport Layer Security (TLS) protocol for authentication. Additional contribution is a decentralized peer authentication scheme compatible with Wi-Fi Aware and TLS that enables peers to verify other peers when offline.

The proposed solution is as follows: A device in an internet isolated location can discover a group of devices participating in a Wi-Fi Aware network. That same device can connect to the network and establish a data path to each device in that group in order to exchange data. Instant messaging can be performed between peers once an mTLS or peer authentication connection is established. A connection requires both communicating parties to verify the other user's identity by exchanging authentication credentials. Digital certificates are provided to users who sign up for the service through an authentication server. Users who have not signed up for the service can receive authentication credentials by being verified by another peer.

The created proof-of-concept application makes it possible to evaluate the performance of Wi-Fi Aware technology experimentally and verify the

proposed TLS offline and online authentication scheme. Testing includes measuring connectivity and messaging time.

The last part of the thesis presents findings, limitations and a comparison of results with other related solutions. The performance of Wi-Fi Aware compares well with a similar solution using Wi-Fi Direct. The proposed Peer Authentication scheme works as a backup authentication solution that enables users to use the application without having signed up beforehand. However, a limitation of this scheme is that peer-authenticated users are not able to communicate with each other.

Sammendrag

I områder rammet av naturkatastrofer, på folksomme konserter og i rurale områder, kan sosiale medier og chatte applikasjoner som krever internettforbindelse bli utilgjengelige. Uten tilkobling til internett blir brukerne isolert og ute av stand til å kommunisere. Det eksisterer åpne kildekode-løsninger som gjør det mulig å sende direktemeldinger til enheter i nærheten. Disse løsningene er imidlertid hovedsakelig begrenset til de trådløse kommunikasjonsteknologiene Wi-Fi Direct og Bluetooth. Wi-Fi Aware lover interessante egenskaper som adresserer manglene av den sentralisert arkitekturen til Wi-Fi Direct og den korte rekkevidden til Bluetooth. Det er imidlertid gjort lite forskning på Wi-Fi Aware og få forsøk på å utnytte dens potensiale. Formålet til denne masteroppgaven er å designe, foreslå og validere en løsning som muliggjør sammenkobling mellom mobile enheter i nærheten av hverandre og gir autentisert og sikker direktemelding mellom brukere i internett isolerte områder.

Den foreslåtte løsningen bruker Wi-Fi Aware for å etablere et peer-to-peer nettverk og mTLS for å gi gjensidig autentisering og kryptert overføring av meldinger. Hovedbidragene som denne oppgaven gir, er forskning på Wi-Fi Aware som en kommunikasjonsteknologi for direktemelding sammen med TLS protokollen for autentisering. Tilleggsbidrag er en desentralisert autentiseringsprosess som gjør det mulig for autentiserte brukere å verifisere andre brukere.

Den foreslåtte løsningen er som følger: En enhet som befinner seg i et internett isolert område kan oppdage en gruppe enheter som deltar i et Wi-Fi Aware nettverk. Den samme enheten kan dermed koble seg på nettverket for å utveksle meldinger. mTLS protokollen brukes for å muliggjøre kryptert og verifisert kommunikasjon over det opprettede nettverket. En sammenkobling mellom to enheter krever at begge parter kan verifisere den andres identitet ved å utveksle autentiseringslegitimasjon. Brukere som registrerer seg når de er tilkoblet internett får utstedt et digitalt sertifikat av en autentiseringsserver. Brukere som ikke har registrert seg til tjenesten kan bli verifisert av en annen autentisert enhet i nettverket, noe som gjør det mulig for brukeren å sender meldinger.

En enkel meldingsapplikasjon basert på den foreslåtte løsningen ble laget for å evaluere ytelsen og sikkerheten til Wi-Fi Aware teknologien samt for å kunne verifisere den foreslåtte autentiseringsprosessen. Eksperimenter utført måler tiden det tar for en sammenkobling og tiden det tar å sende meldinger.

Den siste delen av oppgaven presenterer funn, begrensninger og en sammenligning av resultater med andre relaterte løsninger. Ytelsen til Wi-Fi Aware sammenligner godt med lignende løsninger som bruker Wi-Fi Direct. Den foreslåtte autentiseringsprosessen muliggjør fungerer som en reserveløsning for brukere som ikke har registrert seg til tjenesten på forhånd. En begrensning ved denne løsningen er imidlertid at brukere som er verifisert andre bruker i nettverket ikke kan kommunisere med hverandre.

Preface

This thesis was submitted to finalize our masters degree in Communication Technology at the Norwegian University of Science and Technology. It was completed during the spring of 2021 and based on a pre-project conducted in the fall of 2020.

We would like to express our gratitude to our supervisors Yuming Jiang and Besmir Tola for their guidance and advice during this master thesis.

We would like to thank our parents for the love and support, and especially within the last few months. Additionally we want thank our classmates for making our five years in Trondheim memorable.

Kirsten Lunde Skaug and Elise Breivik Smebye

Contents

List of Figures

List of Tables

List of Acronyms

ADB Android Developer Bridge.

AEAD Authenticated Encryption with Associated Data.

AES Advanced Encryption Standard.

AM Anchor Master.

AP Access Point.

API Application Programming Interface.

BSSID Basic Service Set Identifier.

CA Certificate Authority.

CCMP Counter Mode Cipher Block Chaining Message Authentication Code Protocol.

CDF Cumulative Distribution Function.

CG Cluster Grade.

CRL Certificate Revocation List.

CSR Certificate Signing Request.

DoS Denial of Service.

DSA Digital Signature Algorithm.

DW Discovery Window.

ECC Elliptic Curve Cryptography.

ECDHE Elliptic Curve Diffie–Hellman Ephemeral.

ECDSA Elliptic Curve Digital Signature Algorithm.

GCM Galois/Counter Mode.

GO Group Owner.

IP Internet Protocol.

LE Low Energy.

MAC Media Access Control.

MP Master Preference.

MR Master Rank.

mTLS mutual Transport Layer Security.

NAN Neighbor Awareness Networking.

OSI Open Systems Interconnection.

P2P Peer-to-Peer.

PA Peer Authentication.

PAN Personal Area Network.

PKI Public Key Infrastructure.

PMK Pairwise Master Key.

PSK Pre-Shared Key.

QoS Quality of Service.

RF Random Factor.

SD Standard Deviation.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TSF Time Synchronization Function.

TTP Trusted Third Party.

WPA2 Wi-Fi Protected Access II.

WPS Wi-Fi Protected Setup.

Chapter 1
Introduction

When people find themselves in areas without cellular or network infrastructure, popular messaging applications are unavailable. In such situations, a Peer-to-Peer (P2P) network can be established with the people in the vicinity. However, this raises questions such as how to ensure secure and authentication communication between peers. This thesis proposes a solution that ensures secure communication between peers in the vicinity utilizing Wi-Fi technology.

The following chapter describes a secure instant messaging application when people find themselves in internet-isolated locations. It also presents the scope of the thesis and the research questions that will be answered based on testing the proof-of-concept application that was developed.

1.1 Motivation

These days, most people use cell phones to communicate and perform everyday tasks. Popular messaging and calling applications such as WhatsApp and Facebook require an internet connection in order to access a central server providing authentication and forwarding of messages. However, in out-of-coverage areas such as rural areas that lack a network infrastructure or areas in which natural disasters have damaged the infrastructure, users become isolated from the Internet. Messaging and other communication services on cell phones become unavailable. This can be an issue in scenarios such as mountain rescue operations or in areas prone to natural disasters in which communication is vital.

At concerts, sports events and protests, large crowds of people gather, sharing photos and messages with friends through mobile applications. These applications are services that require an internet connection in order to connect to a central server. When a high volume of network requests are made simultaneously, network congestion might occur. A high traffic load can lead to lower throughput, which

slows down services. In some edge cases, when network traffic is extensive, Denial of Service (DoS) can occur, and network services become unavailable.

In scenarios in which the network is congested due to heavy traffic, device-to-device communication can be a helpful tool for offloading the cellular network. Using Wi-Fi for connectivity is less power consuming for a phone, compared to using a 3G or 4G cellular network [1]. Device-to-device communication in crowded urban environments can offload traffic, thereby improving network performance [2]. By utilizing the radio antennas integrated into smartphones, P2P connections can be established using technologies such as Wi-Fi Direct and Bluetooth. Files, messages and images can be shared directly between devices without a connection to a router or cellular tower. However, Wi-Fi Direct requires one device in a group to serve as a network Access Point (AP), which entails issues such as unfair battery consumption and a single point of failure [3]. Further, Bluetooth is limited by its range and data transfer rates, making it less suitable for large file transfers compared to Wi-Fi technology.

Wi-Fi Aware is a similar technology to Wi-Fi Direct and utilizes Wi-Fi and Bluetooth antennas to discover and create decentralized P2P networks between devices in the vicinity [4]. The network facilitates the sending of encrypted messages, images and files between devices using the widely adopted IP addressing protocol. Wi-Fi Aware runs in the background and notifies users when a match for a service is found, making it energy efficient. There is little documentation beyond the Wi-Fi Alliance specification [5] and few projects have explored the technology. Wi-Fi Aware has untapped potential and properties that could make it suitable as a connectivity technology for instant messaging.

Since Wi-Fi Aware establishes connectivity directly between devices, no authentication server is responsible for verifying the identity of users. How can peers using the service be confident that their messages have been delivered to the intended recipient and not an imposter posing as the recipient? Upper layer measures are required to ensure that there is no man in the middle listening in or modifying the transmitted messages.

1.2 Scope

The scope of this master thesis is to design, implement and validate a solution that enables mobile devices to create a network using Wi-Fi Aware and ensures secure and authenticated communication when no internet connection is available.

The aim if the thesis is to provide a secure and authenticated solution to a Wi-Fi Aware established network. This includes providing authentication credentials, such

as a certificate, to users who have signed up for the service through an authentication server. A certificate serves as proof that a mutually trusted entity has verified the identity of a user. Additional focus has been put into searching for a scheme that can provide authentication credentials to users who wish to join the network when they are offline.

The proof-of-concept application is limited to device-to-device instant messaging between devices in a Wi-Fi Aware network. This enables simple validation of the proposed technology and authentication solutions. However, this can be extended to sending images, files and audio in group chats. Providing the means to communicate using infrastructure Wi-Fi or a cellular network is outside the scope of this thesis.

1.3 Challenges

Wi-Fi Aware uses Wi-Fi Protected Access II (WPA2) Personal in order to ensure confidentiality and integrity protected data transfer during a P2P connection [4]. However, unlike WPA2 Enterprise, WPA2 Personal does not require an authentication server. Without any additional authentication measures implemented by the developer, anyone participating in a Wi-Fi Aware network can send messages to other peers without proving their identity. A user could be talking to an imposter.

A challenge presented in papers and related research is how to authenticate new users when they are already in an out-of-coverage area. Available and well-known authentication schemes such as Transport Layer Security (TLS), explained in more detail in Section 3.2.6, facilitate encrypted and authenticated communication between web applications and servers [6]. Digital certificates downloaded from an authentication server are used in TLS to verify the identity of the communicating parties. Applying this protocol to an out-of-coverage scenario would require users to sign up to an authentication server before going offline. How can new users be authenticated in out-of-coverage areas when they have not received authentication credentials such as a certificate from a central server? These challenges give rise to research questions.

1.4 Research Questions

- Is Wi-Fi Aware a suitable connectivity technology in terms of ensuring confidentiality, integrity and availability for an instant messaging application in areas in which internet connectivity is unavailable?

- How can users be verified and participate in authenticated communication, for both online and offline scenarios?

– How will Wi-Fi Aware perform as a connectivity technology in a secure communication application, compared to other related solutions regarding connectivity time and resource distribution across devices?

1.5 Methodology

The project has been divided into four sections: Preparation, Design, Implementation, and Validation.

1.5.1 Preparation

The preparation phase involved researching connectivity technologies and understanding related solutions for establishing secure P2P networks. The aim of this phase was to learn about unresolved issues in existing P2P communication solutions, technology with untapped potential and areas in which additional contributions could be made.

The solution context, selected during the preparation phase, comprises secure and authenticated communication between mobile devices in internet-isolated locations. Communication during hiking and rescue operations in the mountains was selected as intended use-cases.

During the preparation phase, Wi-Fi Aware was researched and further selected as a connectivity technology. Security schemes provided by Wi-Fi Aware do not include a way to authenticate users. Additional methods and schemes in the application layer are necessary to verify the users participating in the communication. Therefore, different authentication schemes that could work in a decentralized P2P architecture were studied.

1.5.2 Design

The architecture of the proposed messaging solution was designed in this phase. More specifically, a design for a client-server model suitable for Wi-Fi Aware communication was created. The focus of the design phase was to select an authentication scheme that would provide users with authentication credentials in both offline and online scenarios.

An online authentication server was designed to provide users with a signed certificate. In client-server communication, signed certificates are exchanged to prove that a mutually trusted entity has verified the identity of each user. The mutual Transport Layer Security (mTLS) protocol was selected to provide mutually authenticated and end-to-end encrypted communication using certificates issued by the authentication server. The authentication server is only accessible using

the Internet and therefore cannot be used to receive authentication credentials in out-of-coverage areas.

The process of designing an offline authentication scheme involved searching for proposed and existing decentralized authentication schemes that could be used in the solution. The requirement for the scheme was that it could be incorporated into the existing TLS solution.

Figure 1.1 illustrates the high-level schematic view of the proposed decentralized Peer Authentication (PA) scheme. The first step shows communication between two authenticated users, A and B, using mTLS. Since A and B can establish an mTLS connection, they can trust each other. The second step illustrates user A verifying the identity of C and providing C with authentication credentials. C and B can communicate in step 3 because A has verified the identity of C, and because B trusts A.

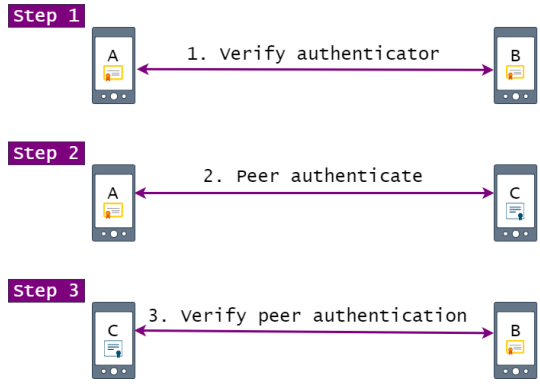


Figure 1.1: High level design of the decentralized PA scheme.

1.5.3 Implementation

The proof-of-concept application verified the designed solution and enabled testing according to the research questions. Android Studio [1] version 4.12 was the development platform and the programming language was Java[2]. The development platform Application Programming Interface (API)s contained the necessary Wi-Fi Aware tools and support to implement the application. API is software that allows two devices to communicate. The phones used throughout the project were a Samsung Galaxy A71 and a Samsung Galaxy S9, both with Android Version 10. Two models were used during testing in order to observe Wi-Fi Aware using different hardware.

[1] Android Studio,https://developer.android.com/studio
[2] Java,https://www.java.com/en/

1.5.4 Validation

In order to verify the proposed solution a proof-of-concept application using Wi-Fi Aware, mTLS and a decentralized authentication scheme has been created. Experiments using the application were conducted in order to validate the solution based on the research questions.

The experimental trials involved measuring the time it takes for a device to discover a Wi-Fi Aware network, establish a connection to it, and the time it takes to send messages. The resulting times have been compared with similar solutions.

Wi-Fi Aware packets were captured in order to gain an increased understanding of how Wi-Fi Aware works and to analyze the link layer security measures it provides. The battery principles of Wi-Fi Aware have also been studied using the captured packets and analyzing the role assigned to each device in the network depending on the battery level.

The packets exchanged between devices were captured using a GNU Radio transceiver[3] with an Ettus USRP B200mini[4] and were further studied using the packet analyzer tool Wireshark[5]. The Android Developer Bridge (ADB)[6] was used to interact with the devices and perform several runs of devices joining a network and sending messages.

1.6 Outline

The remaining chapters of the thesis are divided into six sections and are structured as follows:

Chapter 2: Background and Related Work Introduces the technologies utilized in order to achieve the proposed solution. Related technologies and research are also presented.

Chapter 3: Proposed Solution Describes the proposed solution and how it achieves secure and authenticated message exchange between devices in a Wi-Fi Aware network. The technologies used are explained in more detail.

Chapter 4: Proof-of-Concept Application Describes how the proposed solution is implemented in a proof-of-concept application.

[3]GNU Radio transceiver, https://github.com/bastibl/gr-ieee802-11
[4]USRP B200mini, https://www.ettus.com/all-products/usrp-b200mini/
[5]Wireshark, https://www.wireshark.org/
[6]ADB, https://developer.android.com/studio/command-line/adb/

Chapter 5: Results Presents and analyzes the experimental results with the aim of validating the proposed solution.

Chapter 6: Discussion Discusses the results produced using the proof-of-concept application regarding the research questions.

Chapter 7: Conclusion: Concludes the work performed in this thesis and presents suggestions on how the proposed solution can be improved or further developed by using related technologies and new Wi-Fi Aware features.

Chapter 2
Background and Related Work

2.1 Background

Wi-Fi Aware, also known as Neighbor Awareness Networking (NAN), uses the Wi-Fi radio integrated in phones to enable connectivity between devices in the vicinity. Independent P2P networks can be created to enable data transfer of images, messages and files over TCP/IP[4]. Wi-Fi Protected Access II (WPA2) based frame encryption is used to provide confidentiality and integrity to protect the data being sent [5]. Figure 2.1 shows multi-directional sharing of information between devices in a Wi-Fi Aware group, also called a cluster. Wi-Fi Aware provides mechanisms that ensure low energy and fair resource consumption. Devices synchronize and wake up simultaneously for a short interval to send and receive messages, thereby reducing battery consumption [5]. Technology with low energy requirements is beneficial when used in phones that, do not have a constant power supply. Preserving power is particularly beneficial when phones are used in out-of-coverage areas. Additionally, the workload is fairly distributed between devices by alternating resource-consuming roles. A more in-depth study of battery usage and resource distribution is provided in Section 3.2.2.

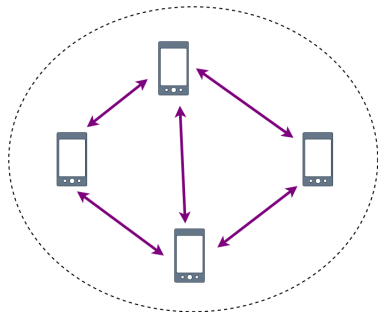


Figure 2.1: Wi-Fi Aware cluster.

A connection is automatically established when a device is in proximity to a cluster, making Wi-Fi Aware suitable for dynamic environments in which devices come and go. The P2P network has a decentralized structure in which each device functions independently of the other devices in the cluster. Consequently, no component serves as a trusted entity responsible for verifying the identity of the users communicating. However, users of an offline communication application expect the same level of security offered by an online messaging application. Mechanisms must be in place for ensuring that an imposter cannot read or modify the contents of messages. Thus, security protocols beyond what Wi-Fi Aware can offer are needed to ensure authentication. mTLS is a protocol that can provide this is.

The TLS protocol is a client-server protocol widely used on the Internet to secure communication between a client, often a web browser, and a server. The protocol was designed to provide reliable high-end services over TCP [7]. The server is responsible for providing a certificate to the client, enabling the client to verify that a mutually trusted party has authenticated the server. The mTLS protocol is an expansion of TLS in which the client is also responsible for providing a certificate, resulting in a mutually authenticated connection. As well as authentication, the protocol offers several cryptographic algorithms to ensure integrity and confidentiality of the messages sent [6]. The general security principles of digital cryptography are covered in the following section to provide a better understanding of TLS and other cryptographic schemes used in the solution.

2.1.1 Security Concepts in Digital Cryptography

Security in computer networks is maintained using three main principles: confidentiality, integrity and availability [8]. These principles are often referred to as the CIA triad and are the fundamental objectives of information and cybersecurity.

Confidentiality is about preventing unauthorized users from accessing data or services. It protects personal privacy in addition to other sensitive information by preventing access by unauthorized recipients. In digital communication, symmetric or asymmetric cryptography is used to provide confidentiality.

Integrity is about preventing unauthorized users from modifying or destructing data. Checks are made to ensure that messages are received and sent without duplication, modification, deletion or replays.

Availability ensures that a system is working correctly and reliably. A loss of availability means that the system cannot provide users with access to information systems or services when needed. Natural disasters, power outages or DoS attacks could cause a loss of availability to services and systems and are particularly important to avoid in critical infrastructure assets such as health, water and communication.

Symmetric Cryptography

Symmetric cryptography, also known as single-key encryption, is used to conceal all kinds of data [8]. The communicating parties use the same secret key to encrypt and decrypt messages, files or passwords. The secret key is established using a secure channel. An analogy used to describe the steps involved in secure digital communication is the communication between Alice and Bob shown in Figure 2.2. Alice encrypts a text using a secret key. When Bob receives the message, he decrypts the text into plaintext using the same secret key. He is now able to read the message from Alice. The TLS protocol uses symmetric cryptography to provide confidentiality to messages.

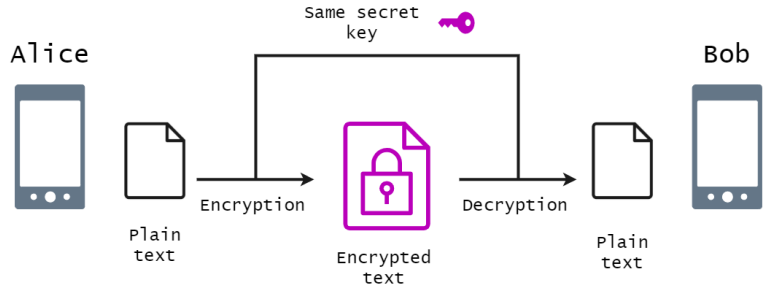


Figure 2.2: Symmetric encryption/decryption.

Asymmetric Cryptography

Asymmetric cryptography, also known as public-key cryptography, is used in digital signatures and certificates to conceal data such as encryption keys and hash values. Unlike symmetric encryption, it uses two separate keys for encryption and decryption: a public key and a private key. The generation of a public-private key pair is shown in Figure 2.3. The public key is used in encryption to conceal a message and only the user with the private key corresponding to the public key can decrypt and read the message contents. Figure 2.4 shows the encryption and decryption process in public key cryptography. When Alice wants to send a message to Bob, she encrypts the message using Bob's public key. When Bob receives the encrypted message, he has to decrypt it using his private key. Since Alice encrypted the message using Bob's public key, his private key is the only key that can decrypt the message. Possession of the private key proves ownership of the corresponding public key. In TLS, key pairs are used to generate and verify signatures.

Digital certificates manage public keys and contain the digital signature of the Certificate Authority (CA) that generated the certificate. Figure 2.5 shows the process of creating and verifying a digital signature. First, the sender generates a hash value for the message using a secure hash function. The hash value is then

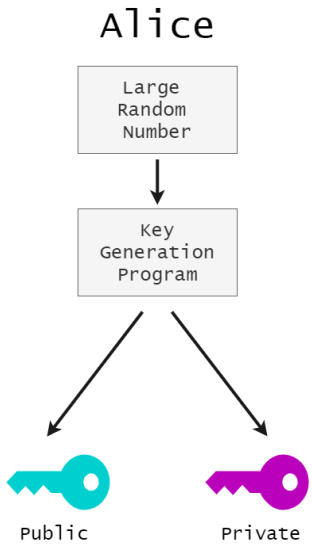


Figure 2.3: Key generation of a public-private key pair.

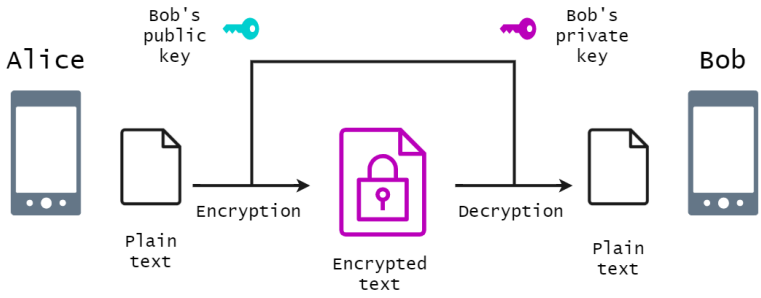


Figure 2.4: Asymmetric encryption/decryption.

encrypted using the senders private key. The result is a short block attached to the message and function as the digital signature. When the receiver receives the message, the sender's public key is used to decrypt the attached signature. The receiver also generates a hash value of the received message. If the hash value and the decrypted signature are the same, the recipient can be certain that no one has forged the message. The process of verifying a signature is used in TLS to verify that a mutually trusted entity, CA, has signed the certificate provided. Only the public key of the CA can be used to decrypt to the same hash value, making the certificate unforgeable.

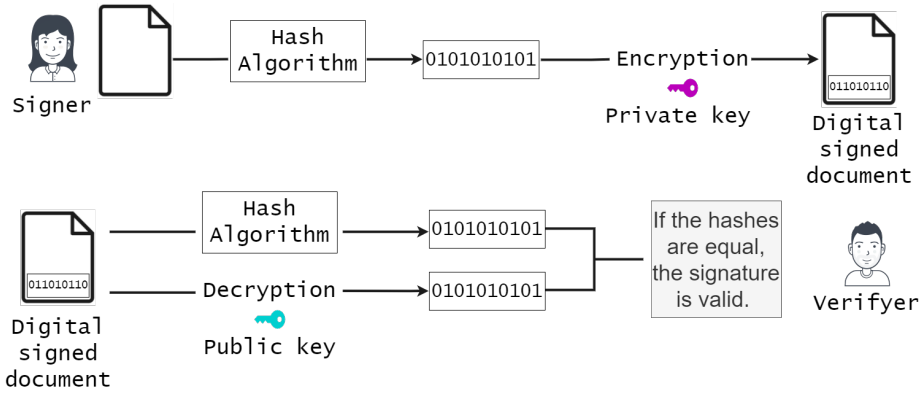


Figure 2.5: The creation and verification process of a digital signature.

2.1.2 Related Technology

There are several technologies that can be used to establish connectivity between devices, other than Wi-Fi Aware. The following sections describe how Wi-Fi Direct and Bluetooth Low Energy (LE) can be used to enable offline communication.

Wi-Fi Direct

Wi-Fi Direct is a technology used in smartphones, cameras, printers and computers to make it easier for users to share content and play games using their devices. Wi-Fi Direct builds upon the traditional Wi-Fi infrastructure used in homes and offices and shares the same security and battery-saving mechanisms [9]. One device must serve as an AP, making it possible for other Wi-Fi enabled devices to connect. The process is similar to connecting to a router. Any client can access by typing in a PIN number to access the network [9]. A device that serves as an AP is called a Group Owner (GO). The GO role is negotiated between the devices in the network. Thus, each device must implement functionality to serve as either a client or a GO. The GO is responsible for forwarding messages to the intended recipients. This places a considerable burden on the GO's resources compared to the other members of the network [9]. In situations in which devices do not have a constant power supply, this can be an issue.

Bluetooth Low Energy

Bluetooth Low Energy (LE) is a connectivity technology used to create a Personal Area Network (PAN) and exchange data using radio waves in the 2.4 GHz frequency band. It provides reliable and low-power operations, making it suitable for medical

and fitness applications that run on low-power devices. The following topology structures are supported: device-to-device, broadcast and mesh networking.

Bluetooth LE supports a data transfer rate of up to 2 Mb/s over 40 channels [10], which is considerably lower than typical Wi-Fi rates. The range of Bluetooth LE is limited and considerably lower than the range of Wi-Fi Direct [11]. Although the theoretical maximum operation range of Bluetooth LE is up to 100 m, the effective range for a smartphone using Bluetooth is around 10 m [12]. The range depends on the environment and strength of the radio signal.

2.2 Related Work

Several applications available for consumers use either Wi-Fi Direct or Bluetooth LE as a connectivity technology to enable instant messaging between devices.

Bridgefy is an offline communication application that uses Bluetooth LE to exchange messages between devices [13]. Before using the application in out-of-coverage areas, it requires users to connect to the Internet [13]. The application sets up a mesh network using Bluetooth LE and broadcasts messages to all users within range, contacts and non-contacts [13]. The mesh network extends the communication range by having peers forward messages to other peers until they reach the intended recipient. Messages sent between two devices are encrypted. Broadcast messages are also encrypted but are visible to all users receiving the broadcast messages [13].

FireChat is another application that uses mesh networking. It enables users to send and receive messages using Wi-Fi and Bluetooth LE radios for connectivity [14]. Users can establish anonymous chat rooms on different topics and subscribe to these topics. The application gained popularity during the mass protests in Hong Kong in 2014 and relieved the heavy network traffic caused by the large gatherings of people [14].

Another application that uses Wi-Fi Direct to establish a network is Signal Offline Messaging. Privacy is an important security feature provided by Signal Offline Messaging so the application is often used by activists and journalists. It is stated that the application has end-to-end encryption and that no data is stored in the cloud, just locally on the devices [15] [16].

Serval Mesh is an offline application developed by the Mesh project, an Australian based company aiming to provide connectivity in rural areas of Australia. The use of mesh networking and ad-hoc Wi-Fi mode enables users to share files, receive calls and send messages [17]. However, Wi-Fi ad-hoc mode is currently not supported by Android smartphones without gaining root access to the phone [18].

2.2.1 Related Research

The following section presents research on the NAN protocol, as well as research on authenticated communication in P2P solutions. The schemes studied were considered as potential authentication schemes for the proposed solution.

An overview of the NAN protocol is presented in the paper by Campus-Mur et al. [19], in addition to an evaluation of cluster formation and mobility patterns. The NAN protocol was implemented in a packet-level simulator based on the network simulator OPNET[1]. Realistic user movement through a crowded street was simulated using MobiREAL [20]. It was observed that there were a maximum of five hops, chain of devices, between the device controlling the cluster to the devices on the periphery. On a few occasions, the devices were unable to discover a cluster, even though it was in range. This resulted in the device becoming an Anchor Master (AM) of its own cluster, leaving it isolated from the rest of the network. It is worth noting that the solution is based on the NAN protocols specified in [5] and was published before the Wi-Fi Aware API was available in Android Studio.

The Signal protocol, used in several messaging applications such as Signal Offline Messaging and WhatsApp, uses public key fingerprints to verify the identity of users [21]. A public key fingerprint is a shorter version of a public key, created by applying a cryptographic hash function to the public key. Fingerprints are shared between users over an independent channel, such as over email or by meeting up in person. If the fingerprint presented by a user matches the one provided over the independent channel, the user is authenticated. Applying this method to an offline instant messaging application presents the challenge of sharing the fingerprint when independent channels are unavailable. One approach could be to require users to exchange fingerprints before moving to out-of-coverage areas. A limitation to this approach is that users who have not exchanged fingerprints are unable to communicate. These users would have to physically meet up and compare keys in order to participate in authenticated communication.

In the solution presented by Khacef and Pujolle in [22], blockchain technology is used to ensure secure and authenticated message exchange. The work proposes an authentication solution that uses smart contracts through the Ethereum blockchain, thereby removing the need for a CA. Peer information such as public keys, digital signatures and general peer information is stored on the blockchain. Blockchain technology has a decentralized architecture whereby computers all around the world participate. A decentralized architecture results in a reliable system. It will not be possible to shut down all the participating computers simultaneously. However, after a contract has been stored on the blockchain, it is not possible to modify it

[1]OPNET, http://www.opnet.com

without using extensive amounts of power. Additionally, adding a smart contract onto the blockchain, making it accessible to others, requires internet access. Thus, the solution is unsuitable for authentication in out-of-coverage areas.

Sigholt, Tola and Jiang used mTLS to provide an authenticated and secure out-of-coverage instant messaging solution using Wi-Fi Direct [23]. The solution comprises an authentication component that issues certificates in addition to a server and a client component. The server component is responsible for forwarding messages and verifying client certificates. However, the solution is unable to authenticate new users in out-of-coverage areas, which is presented as the main drawback. The messaging service is unavailable for unauthenticated users until they can establish a connection to an online authentication server and receive a certificate.

The decentralized authentication scheme presented by Santos-González et al. [24] provides a means of authenticating new users without an internet connection. Wi-Fi Direct, Bluetooth LE or LTE Direct are the proposed connectivity technologies used in the solution. Users with internet access can send authentication requests to a server in order to gain access to private chat rooms. However, if a user cannot reach the server, an authentication request can be made instead to a peer with the required level of trust. The authenticated user signs the public key of the unauthenticated user and provides it with its certificate. These credentials are used to gain access to private chat rooms that require authentication. This solution resolves the issue described by Sigholt et al. [23] of having to be online in order to be authenticated.

Chapter 3
Proposed Solution

This chapter describes the architecture and the components of the proposed solution. The technology and security measures required for authenticated secure communication between peers in an out-of-coverage scenario will be covered.

3.1 The Architecture

The proposed solution comprises an authentication server, responsible for providing a signed certificate to users signing up, a client-server pair and a PA client-server pair. An additional PA client-server pair is only used to initiate and respond to connections made by users who are peer-authenticated. The proposed peer authentication scheme enables users to vouch for other peers while in internet-isolated locations. This is introduced and further explained in Section 3.1.6.

3.1.1 The Authentication Server

The solution for the authentication server is in accordance with the Public Key Infrastructure (PKI) as used in TLS [25]. The authentication server acts as a CA. It is responsible for issuing, signing and revoking the digital certificates of new users who join the proposed instant messaging service. The CA is a Trusted Third Party (TTP), meaning it is trusted by both the owner of the certificate and the party relying on the certificate.

The authentication component is only reachable via the Internet. Thus, a user who has not signed up for the service before moving to an internet-isolated location cannot participate in an mTLS connection, in which both parties are required to provide a certificate signed by a CA.

3.1.2 Certificates

The certificates used in the proposed solution comply with the X.509 certificate standard. The certificate binds an identity to a public key for further use in asymmetric cryptography. Each certificate contains the subject's public key, in addition to the certificate subject, the issuer and the expiry date. The public key is part of a public-private key pair generated when the application is downloaded. The certificate contains a signature signed by either the CA or the certificate owner (self-signed), depending on whether the user signed up for the service before going offline.

3.1.3 Signing Up

When a user connects to the authentication server and signs up to the instant messaging service, a Certificate Signing Request (CSR) is created using the key pair generated during download. This process is illustrated in Figure 3.1. The key pair consists of a private and a public key. The private key will be kept private and only accessible to the user who is signing up. The CSR contains information such as country, email address, common name and public key.

If the authentication server considers the information provided in the CSR to be correct and trustworthy, the certificate request is approved, and a certificate is generated. The certificate is then signed by the CA and returned to the user who requested the certificate.

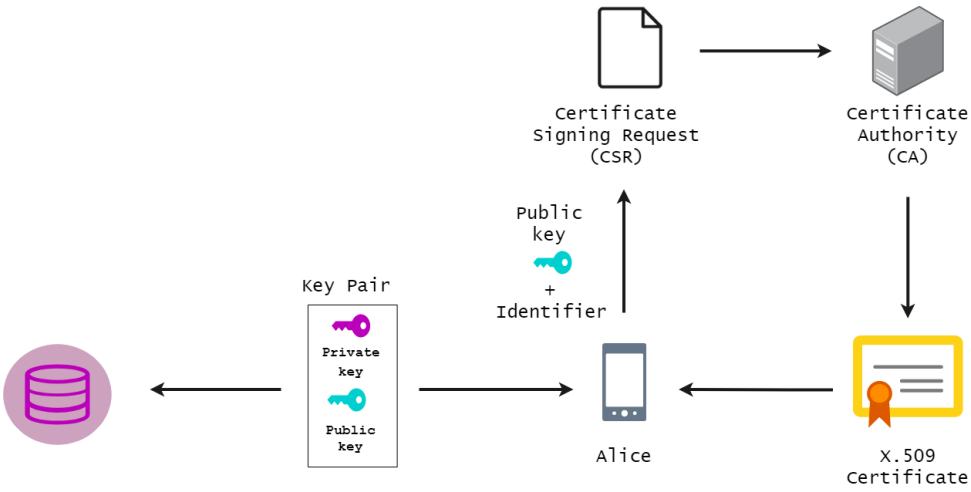


Figure 3.1: Issuing of X.509 certificates.

3.1.4 Certificate Revocation

Certificates are expected to last for their entire period of validity. However, various circumstances can cause revocation. Users who have violated the terms and conditions should be restricted from using the service. If a public key has been compromised, the certificate should also be revoked.

A Certificate Revocation List (CRL) is a collection of certificates revoked before their expiry date that should no longer be trusted [26]. The list is issued and maintained either by the CA or the CRL issuer and made available in a public repository. The list contains the serial number of the revoked certificates, in addition to the signature of the issuer that revoked it. In the process of determining whether a certificate is valid and trustworthy, the verifier checks that the certificate's serial number is not included in the latest CRL acquired.

Downloading a CRL requires access to the authentication server. In order to receive an updated version of the list, a user must frequently move into areas with internet access. A solution to this issue could be to share recent updates of the CRL between trusted peers while in internet-isolated locations.

3.1.5 The Client-Server Model

In the proposed solution, all devices participating in a Wi-Fi Aware network constantly have a server running, awaiting incoming requests from clients. No device acts as an AP forwarding messages for other devices. Thus each device must have a server running in order to be reachable. Consequently, each device must be able to work as both a client and a server. The device that first initiates a connection to another device is assigned the client role.

The Application Server

The server is responsible for awaiting any incoming requests from clients at a specified port and must handle multiple connections to clients simultaneously. Each device runs a server on a unique port, making every device reachable for communication.

Data exchange between a server and a client is over TCP/IP using the Wi-Fi Aware data path and the port advertised by the server. When a server receives an incoming request from a client, mutual authentication and the negotiation of cryptographic protocols are performed using mTLS before setting up a connection.

When using mTLS, both server and client must exchange certificates to verify that both parties have signed up for the service. The server will not accept communication sessions without the client first providing a certificate signed by the CA.

The Application Client

The client is responsible for requesting a communication session from a server and providing it with a certificate. A network can comprise multiple devices. Thus, a device must be capable of handling multiple server connections.

The user initiating the communication session with another peer is automatically assigned the client role for that specific session. Any user participating in a P2P network can use an established data path to another peer (server) to initiate a communication session. The port used to contact the server is exchanged during the establishment of the data path.

The client secures its communication to the server by only accepting certificates signed by the CA. The cryptographic protocol specifies the encryption schemes supported by the client, which are provided to the server during connection setup.

3.1.6 Peer Authentication Model

This thesis proposes a peer authentication scheme based on the technique of Santos-González et al. [24], which enables users to be verified by another authenticated peer instead of the authentication component. The scheme provides unauthenticated users with authentication credentials, instead of a signed certificate. The solution enables new users to communicate with other users, despite not having signed up for the service before moving to an out-of-coverage area.

The technique is incorporated into the solution by using a server that does not rely on a certificate signed by the authentication component. Thus, a non-mutual TLS server that only accepts valid authentication credentials must be used for all Peer Authentication (PA) connections. Details about what authentication credentials comprise and how they are issued are presented in the following chapter.

Peer Authentication Server

The device serving as a PA server is responsible for responding to a PA request from a client and serving incoming TLS connections that only requires the server to provide a certificate.

The PA server uses the authentication credentials provided by the client instead of a certificate signed by the CA to verify that another peer has authenticated the client. The credentials are provided to the application server before starting the server, using Wi-Fi Aware short messages with a 255-byte limit. The server can receive these credentials solicited, request them from the client, or unsolicited, either by the client itself or another user in the network who has exchanged the credentials.

This means that both the server and the client can initiate the process of starting the PA server.

If the client's credentials are valid, the PA server is started using a different port from the regular server. The client is then able to communicate with the server over a mutually authenticated and secure connection. The server can set the terms of the connections it accepts and therefore choose to accept connections in cases where the client does not provide a certificate. However, the server still needs to provide a valid certificate to the client in order to establish a connection. Thus, only a peer that is authenticated by the CA can assume the role of hosting a PA server.

A client that has not been verified by another peer will not be able to communicate with the server. If the client authentication credentials provided to the server are invalid, the PA server will not be started. Once the server is started, it will only accept incoming connections from clients with valid credentials.

The PA server will be started if a request from a client is made and only if the authentication credentials provided by the client are valid. It is assumed that the number of peer-authenticated users is limited, and continuously running an additional server is resource-consuming and unnecessary.

Peer Authentication Client

The peer-authenticated client is responsible for providing the PA server with authentication credentials and requesting a PA connection. By providing authentication credentials, the client proves that another authenticated and trusted user has vouched for its identity. A device that has not signed up to the service before moving to an out-of-coverage area will become the client in any client-server connection. Any connection to a mTLS server will be rejected because the user does not have a certificate signed by the CA.

A peer-authenticated client operates in the same way as a regular client and a connection requires a PA server to provide a valid certificate. Therefore, a connection between two peer-authenticated users is not possible. The client provides the server with its supported cryptographic protocols for negotiation.

Figure 3.2 illustrates how an authenticated device, A, verifies an unauthenticated device, C, thereby enabling B and C to set up a PA TLS connection. Step 2 illustrates the successful verification of C. Consequently, B and C can establish a secure connection.

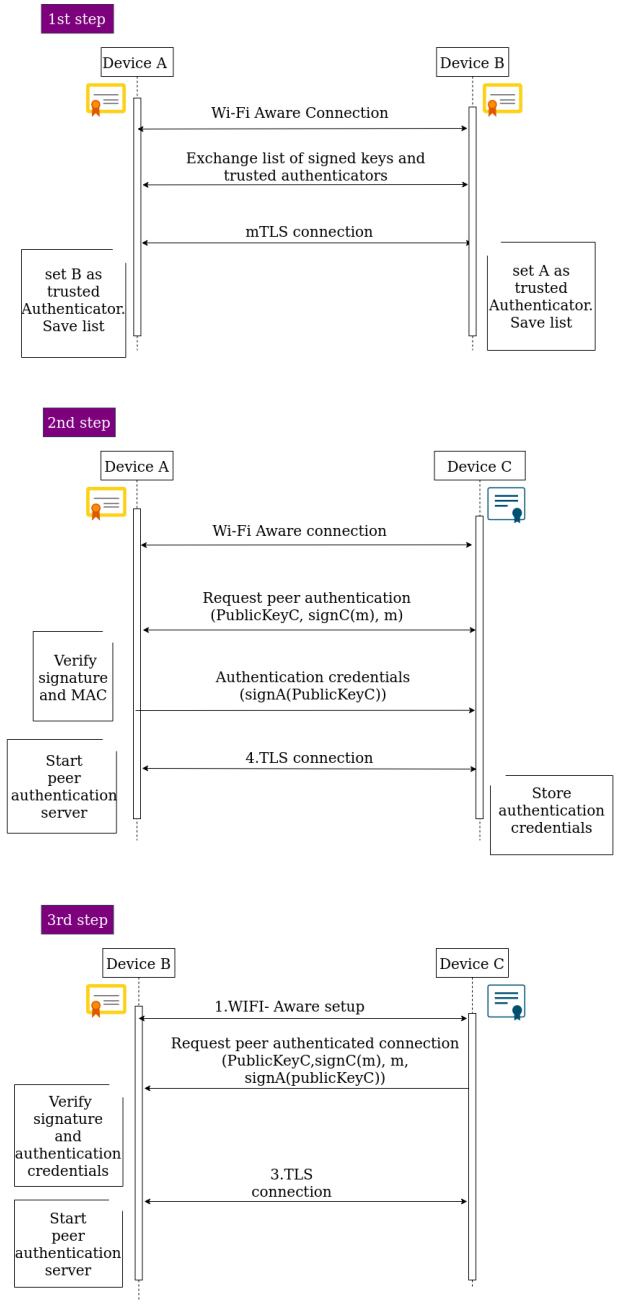


Figure 3.2: Sequence diagram showing the steps necessary to set up a PA connection between two devices, B and C. A and B are authenticated by the CA and C is authenticated by A.

Peer Verification

A user who does not have a certificate signed by the CA can request authentication from another peer in the network, as illustrated in step 2 in Figure 3.2. Similarly, any device with a certificate signed by the CA can verify another peer. When an authenticated user receives a request from a user who wants to be authenticated, it can deny or accept the request. Such a decision depends on the authenticator's trust in the validity of the identity presented by the peer requesting authentication. If two users are standing next to each other, it can be verified that the Media Access Control (MAC) address displayed on the requester's device is the same as the address presented in the request. If a user cannot verify that the request is from a trusted user, the user should not be authenticated.

If a user decides to accept a request, it will generate and send authentication credentials to the other users in the network, informing them about a new authenticated peer. These credentials will enable the peer-authenticated user to prove that it has been vouched for by an authenticated user in future interactions. Thus, the peer-authenticated user is able to participate in authenticated and secure communication, as shown in step 3 in Figure 3.2.

If the request is denied, the user who wants to be authenticated cannot participate in any connections for communication. It can, however, try to request authentication from another peer in the network. This is necessary if a device leaves the network and joins a new network. The devices participating in the new network will have no knowledge of the authenticator and will therefore not trust the authentication credentials presented. The reason why a user must have knowledge of the authenticator is explained in the following section.

Peer Authentication Credentials

The PA scheme proposed by Santos-González et al. uses signed public keys and certificates as authentication credentials. The authentication credentials are provided to the unauthenticated user when it has been verified and then presented to other users when requesting access. Similarly, the proposed solution uses a signed key as proof of PA. However, public keys are used instead of certificates to verify that an authenticated user has provided a signature. Public keys are shorter than certificates and can be sent using Wi-Fi Aware short messages. Unlike the scheme of Santos-González et al., the PA scheme is limited to one level, meaning that peer-authenticated users cannot authenticate other unauthenticated users.

Using public keys to verify an authenticator instead of certificates involves an additional step, step 1, in the authentication process as illustrated in Figure 3.2. Before the authenticated user B and the peer-authenticated user C can communicate,

B must verify that user A is an authenticated user who is trusted to verify C. Thus, A and B must have had a secure and mutually authenticated connection beforehand and saved each other's public key. Instead of performing this procedure with every authenticator, a list of public keys that are trusted to perform PA can also be exchanged between users, reducing the number of interactions between devices. A list of signed keys can also be exchanged in the same way. Information shared by a peer should be discarded if a mutually authenticated connection has not taken place.

An authenticator will only authenticate and sign the keys of users who can prove ownership of the key presented. As shown in Figure 3.2, C provides A with its public key, a signature on a random string and the random string. Device A uses these values to verify the signature and ensure that C holds the private key corresponding to the public key presented. Digital signatures and how they work are covered in more detail in Section 2.1.1.

Before a PA connection can be established, the connection requester must prove ownership of its authentication credentials, as illustrated in step 3 in Figure 3.2. This is similar to how A verifies C in step 2 but, in addition, C must provide B with its public key signed by A. Thus, when B receives the authentication credentials it must verify that the public key of C has been signed by an authenticated user. If the signed key decrypts to C's public key using A's public key, the process has verified that C is the owner of the signed key and the PA connection request is accepted.

3.2 Technologies Used

The general principles of the TLS protocol and Wi-Fi Aware technology are presented in this section together with how they are utilized in the solution. Also, the security principles used in the solution are presented and justified to better understand the schemes and protocols being used.

3.2.1 Wi-Fi Aware

The Wi-Fi Aware Specification, previously known as NAN, is described in the publication of Wi-Fi Alliance [5]. Devices that support Wi-Fi Aware can autonomously form clusters of devices and advertise services to facilitate IP communication via Wi-Fi or Bluetooth LE. Bluetooth LE is used for low energy devices and service discovery to reduce power consumption [4]. Wi-Fi Aware operates on the link layer in the Open Systems Interconnection (OSI) model and is responsible for transferring data frames between nodes across the physical layer [5]. MAC addresses are used to deliver data link frames to and from the correct destination.

A Wi-Fi Aware cluster comprises a group of devices listening on the same channel and synchronized to the same "heartbeat" or clock. A Discovery Window (DW) specifies the channel and clock on which all devices must converge to achieve optimal performance. Devices wake up during the DW to listen for or advertise services, as illustrated in Figure 3.3. Synchronization allows the devices to continuously search for other devices without transmitting and receiving messages, thereby reducing power consumption.

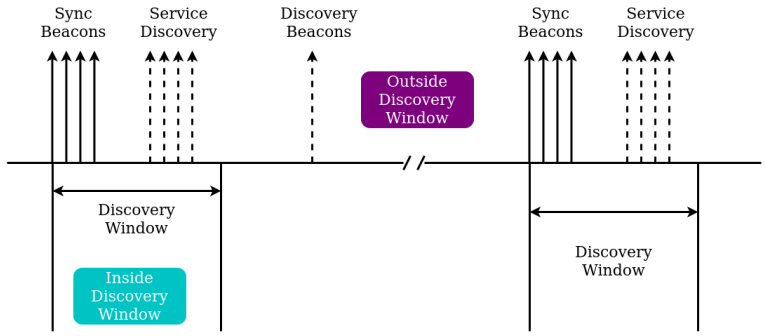


Figure 3.3: Illustration of the DW in which devices wake up to discover services and receive Synchronization Beacons.

3.2.2 Roles

The device responsible for the synchronization of the DW is called the Anchor Master (AM). The Master role is responsible for propagating Discovery and Synchronization Beacons, which is a resource demanding task. Thus, to achieve fairness, this role is periodically alternated. In order to determine the role of each device, their Master Rank (MR) is calculated based on three values: their Master Preference (MP), a Random Factor (RF) and their Wi-Fi Aware MAC address [5]. The MP is a value that each device sets based on how much it wants to be the AM in the cluster. The MR is a value between 0 and 255 and is calculated as follows:

$$\text{Master Rank} = \text{MP} * 2\hat{}(56) + \text{RF} * 2\hat{}(48) + \text{MAC}[5] * 2\hat{}(40) + ... + \text{MAC}[0]$$

Based on how the MR is calculated, the MP is the value that affects the MR the most. According to Wi-Fi Alliance, a device with fewer battery restrictions will have a higher preference and is more likely to have a Master role [5]. Devices with a large battery capacity, stable clock, or devices connected to a power source should choose a large MP value. Devices that are periodically moving, such as mobile devices, should

select a lower preference. If two devices have the same MP, fair battery sharing is provided by having each device change its Random Factor every 1 to 2 minutes [19].

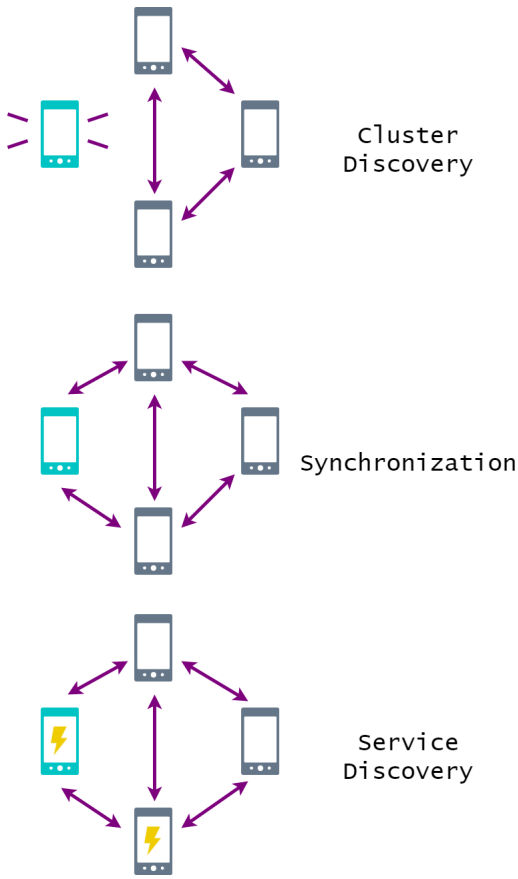


Figure 3.4: Wi-Fi Aware cluster discovery, synchronization and service discovery.

3.2.3 Cluster and Service Discovery

In order to form a cluster, devices must discover and synchronize with each other. Figure 3.4 illustrates this process. Devices can discover nearby clusters by listening for Discovery Beacons transmitted outside the DW by a device operating in a Master role, as shown in Figure 3.3. Transmission is on common channels in either the 2.4 GHz or 5 GHz band. If a cluster is not detected, the device establishes a new cluster and starts transmitting Discovery Beacons advertising its presence. If a cluster is detected, the device receives synchronization information about time frames and channels. The last step is the service discovery process, where devices can publish a

service or subscribe to one that matches their needs, such as an instant messaging service.

The device clock must be synchronized to an AM clock in order to know which channel and what time to listen for or announce services. Thus, devices only participate in one cluster at a time. For this reason, Wi-Fi Aware specifies an algorithm that causes clusters in overlapping areas to converge into one cluster [5]. A device will converge if it receives a Synchronization Beacon from another cluster with a higher Cluster Grade (CG). The CG is calculated based on the AM Rank and Time Synchronization Function (TSF) of the cluster. A TSF keeps the clocks of devices in the same cluster synchronized. If the AM discovers another cluster with a higher CG, it will converge with that cluster. Prior to this, it will transmit a Synchronization Beacon to the previous cluster containing information on how to converge with the new cluster. Cluster merging enables devices to share services with a larger audience and reduces traffic.

3.2.4 Publish and Subscribe Messages

Every device works as both a publisher and a subscriber in the proposed instant messaging solution. The publisher is responsible for continuously transmitting messages on the air advertising its service and awaiting requests from subscribers. The subscriber is responsible for listening for matches to the transmitted messages. Each device must have the ability to publish and subscribe to the instant messaging service advertised in order to be able to initiate and respond to a data path connection. In a P2P network, all devices have equal responsibility and functionality. Being assigned fixed roles as either publisher or subscriber would limit the solution's usability and not correspond well with instant messaging functionality.

Before establishing a data path, the devices must agree upon which devices use its publish session and which devices use its subscribe session in order to avoid setting up two data paths. The MAC address of each device is used to decide which session to use. This is achieved by converting the address to a decimal value for comparison. The device with the highest MAC address keeps the publishing session, and the other keeps the subscribe session.

3.2.5 Security

After a device has obtained either a publish or subscribe session, a data path can be established. This process is illustrated in Figure 3.5. The subscriber initiates a 4-way handshake by requesting the data path. The data path is secured using IEEE 802.11 WPA2 based frame encryption to protect the Data and Action frames being exchanged [5]. Encryption is an optional feature in Wi-Fi Aware and must be implemented by the developer. A Wi-Fi Aware Shared Key Cipher suite is used in

the proposed solution, in which each device knows a Pairwise Master Key (PMK) derived from a Pre-Shared Key (PSK)[5]. The PSK is a passphrase comprising 8 to 63 ASCII characters and is retrieved when downloading the application. Knowledge of the PMK is confirmed during the handshake. Wi-Fi Aware supports several cipher suites and the strongest scheme supported by both devices is selected. The minimum requirement is to use Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP)-128 for frame encryption, SHA-256 for the hashing function and HMAC-SHA-256 for the key derivation function. The CCMP protocol uses the CBC-MAC protocol that uses the Advanced Encryption Standard (AES) and provides confidentiality and integrity. AES is explained in more detail in Section 3.3.1. In order to establish a temporary shared symmetric key for encryption, PMK and nonces are negotiated during the handshake.

A privacy feature provided by Wi-Fi Aware is the use of local Wi-Fi Aware Interface Addresses that change periodically [5]. Unlike fixed MAC addresses, local and dynamic addresses prevent user location and behavior tracking. Address tracking can result in the identification of users and is a breach of privacy.

3.2.6 mutual Transport Layer Security

As previously presented, mTLS is a widely used and studied protocol for securing communication in client-server relationships. The protocol requires both client and server to provide a certificate for the authentication of both parties. In a decentralized P2P network, all peers have the same functionality and privilege. Thus, each user has an equal reason for proving their identity, making the mTLS protocol suitable for authentication in the proposed solution.

In addition to authentication, mTLS uses cryptographic algorithms negotiated during a 4-way handshake to provide message confidentiality and integrity. Messages are encrypted to prevent unauthorized persons from reading the messages and integrity ensures that the receiver can detect any unwanted modification of the message during transmission.

Establishing an mTLS connection between a client and a server requires a 4-way handshake between the participating parties, as shown in Figure 3.6. The purpose of a 4-way handshake is to negotiate cryptographic capabilities, protocol version, exchange certificates and material to establish keys [6]. Each device may have different cryptographic capabilities. Thus, negotiation is needed to select an algorithm that is supported by both client and server. The key established in the handshake protects all messages sent after the handshake and is part of symmetric key cryptography, as explained in Section 2.1.1.

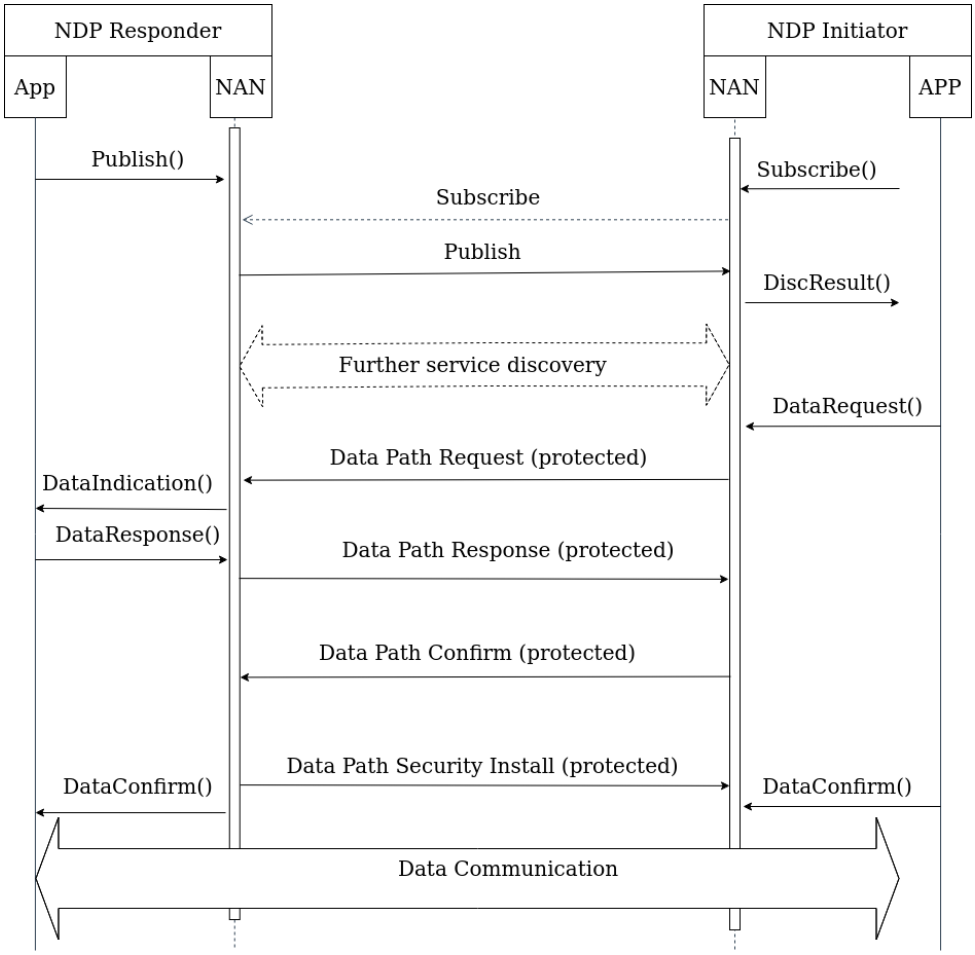


Figure 3.5: The 4-way handshake process of Wi-Fi Aware. The handshake illustrates a subscriber discovering the services offered by a publisher and then requesting a secure data path.

3.3 Cryptography

The proposed solution uses several cryptographic principles to provide confidentiality and integrity protected data transfer over an insecure channel. In mTLS this is achieved using symmetric and asymmetric encryption. The objective is to protect against a malicious adversary trying to read or modify the messages exchanged. A high level of security is achieved by selecting cipher suites and a key exchange algorithm supported by TLSv1.3. TLSv1.3 removed support for several outdated

algorithms and ciphers due to known vulnerabilities and attacks [6].

The key exchange algorithm, Elliptic Curve Diffie–Hellman Ephemeral (ECDHE) with Elliptic Curve Digital Signature Algorithm (ECDSA), uses Elliptic Curve Cryptography (ECC) keys and is the only key exchange algorithm supported by TLSv1.3. The algorithm is used to exchange keys for use in symmetric encryption, and mandates Perfect Forward Secrecy [27]. In the event of long-term key compromise, forward secrecy protects the long-term key from exposing the session keys. Session keys are used to encrypt the messages sent, and a compromise of this key could expose an entire conversation.

ChaCha20 with Poly1305 and AES 128 in Galois/Counter Mode (GCM) are the cipher suites used for symmetric encryption in the proposed solution. Using ChaCha20 with Poly1305 is fast and battery friendly and is therefore suitable for running in phones. However, AES is still the standard encryption algorithm used [28] and is faster than ChaCha when used in a device that includes specialized AES hardware [28]. Two different cipher suites have been selected in order to investigate whether using ChaCha as an encryption scheme would result in shorter message times.

Both cipher suites used in the proposed solution are Authenticated Encryption with Associated Data (AEAD) ciphers. An AEAD is an algorithm that provides confidentiality, integrity and authenticity of messages [29]. The algorithm combine an encryption and a Message Authentication Code algorithm into one, making implementation simple and reducing the amount of computation. TLSv1.3 only supports the use of AEAD algorithms.

3.3.1 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a cryptographic scheme in which the security is based on the discrete logarithm problem being a hard problem. Finding the discreet logarithm problem for a point from a random elliptic curve in polynomial time is not feasible with current computers [30].

An ECC scheme using symmetric encryption and a public-key system is called a hybrid encryption scheme. Implementing ECC as a hybrid scheme is the standard way of using ECC and is called the Elliptic Curve Integrated Encryption System [31]. The material used to establish the key to be used in symmetric encryption is sent using public-key cryptography. The keying material is determined using a key agreement, which is a technique used to agree on the key material to be used when creating keys. In ECC, both participating parties contribute to the input for the keying material. In mTLS, the keying material is exchanged during the 4-way handshake using ECDHE, as shown in Figure 3.6.

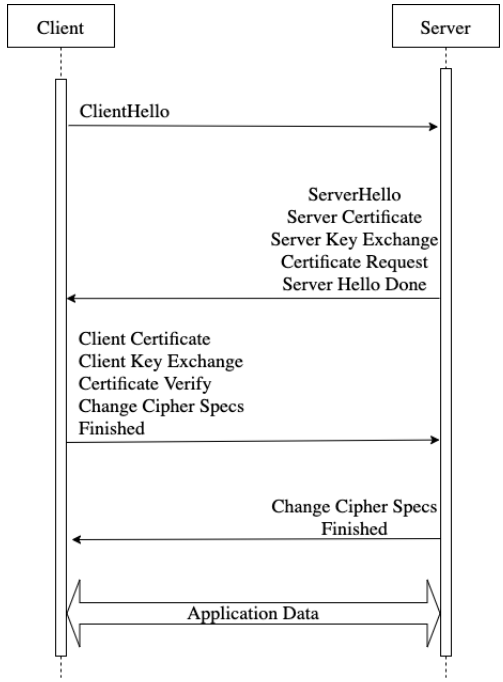


Figure 3.6: Message flow for the mTLSv1.2 handshake protocol [32].

Elliptic Curve Diffie-Hellman Ephemeral

A widely used key agreement protocol is Diffie-Hellman and, more specifically, ECDHE. Diffie-Hellman Ephemeral is a key exchange protocol. The term ephemeral stands for temporal, meaning the protocol discards the keys after use. Diffie-Hellman enables the secure exchange of keying material over an insecure channel in order to establish a shared secret [33].

Elliptic Curve Digital Signature Algorithm

In 2009, a new digital signature technique, ECDSA, was introduced [8]. It uses keys generated by using points on an elliptic curve and can therefore use smaller keys and shorter signatures to achieve the same security features as a traditional Digital Signature Algorithm (DSA) scheme [34].

3.3.2 Encryption Schemes

The solution proposes two symmetric encryption schemes, AES in GCM and ChaCha20 with Poly1305, in order to compare the efficiency of the two schemes implemented in the proof-of-concept application.

A widely used symmetric encryption algorithm is the Advanced Encryption Standard (AES) algorithm, which was finalized in 2001 and designed to replace the previous algorithm Data Encryption Standard [8]. AES increased the key size required, offering 18-, 192-, or 256-bit keys [8]. AES combined with GCM is an authenticated encryption algorithm with AES as the block cipher and GCM as the mode of operation, which incorporates Counter mode [29]. Counter mode can use pipelining in hardware implementation and is the mode of choice when developing high-speed applications. However, it does not provide message authentication. Thus, the AEAD algorithm AES in GCM includes a Message Authentication Code that is based on polynomial hashing [35].

Another AEAD algorithm is ChaCha20 with Poly1305, designed by Adam Langley in 2013 [36]. It combines the symmetric stream cipher ChaCha20 and the cryptographic message authentication code Poly1305, both of which were first designed by David J. Bernstein. ChaCha20 is used to encrypt messages, while Poly1305 ensures the integrity and authenticity of the message [37]. The key size is 256-bit. ChaCha20 with Poly1305 was designed as a response to the concerns of the existing cipher suites in TLS, such as attacks on the stream cipher RC4 [38]. AES-GCM addresses some of these concerns. However, the performance of AES-GCM and implementing it effectively in software are still concerns. Both ChaCha20 and Poly1305 were designed for high performance in software implementation. In addition, ChaCha with Poly1305 was designed for easy implementation into software without it being vulnerable to software side-channel attacks as they minimize the leakage of information [38].

Chapter 4
Proof-of-Concept Application

This chapter describes the proof-of-concept application created in order to verify the proposed solution presented in the previous chapter. Specifics regarding the functionality of the application, simplifications and technical details will be covered.

4.1 The Authentication Component

Users typically connect to an application server over the Internet in order to sign up and register a profile in order to receive a certificate. Creating a server that manages the sign-up of new users and issues signed certificates is beyond the scope of this thesis. In the proposed solution, an authentication component was created by manually signing Certificate Signing Request (CSR)s on an Ubuntu machine using the OpenSSL toolkit[1]. OpenSSL is an open-source command line tool that can be used to generate key pairs and sign CSRs.

The authentication component was created by generating a private-public Elliptic Curve key pair with a corresponding root certificate. The component worked as a CA responsible for signing CSRs and issuing all of the client/server certificates. The process of creating and signing certificates was performed manually due to the limited number of devices involved in the testing process.

4.1.1 Generating Certificates

User credentials were created using the Java Keytool [2] command. The Keytool command is used to store, manipulate and access key pairs and certificates in a Java Keystore. The generated private-public key pair of the user was used to create a keystore and a CSR.

[1] OpenSSL,https://www.openssl.org/
[2] Java Keytool,https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

Two types of certificates were created: a validated certificate signed by the CA and a self-signed certificate. Self-signed certificates are not possible to use in an mTLS connection because it has not been signed by the CA. A user will only use one of these certificates, depending on whether it signed up to the service while online or not. If the user signed up, it will always use the certificate provided by the CA since it is signed by a mutual trusted party.

A valid certificate is generated by providing the CSR to the CA and then using OpenSSL to sign the CSR using the CA's private key. A self-signed certificate is generated by using the client's own private key to sign the CSR.

The root and the client certificate are loaded into the client keystore. This process is the same for the CA signed certificate and the self-signed certificate. The keystore is then manually loaded into the internal file system of each phone and used for setting up a TLS connection. This process was also performed manually due to the limited number of devices.

For simplicity during development, no intermediate certificate acting as a middle-man between the root certificate and the client/server certificate was used. This could be added for an extra layer of protection as it would minimize the damage in the event of a security breach.

4.2 The Application

The application comprises of two graphical user interfaces: the Main Activity and the Chat Activity. The Main Activity lets the user see nearby peers, and the Chat Activity lets the user chat with other peers in the formed cluster.

4.2.1 Main Activity

The Main Activity is the first interface presented to the user and is responsible for handling the Wi-Fi Aware functionality and setting up the application server. The Wi-Fi Aware functionality includes turning on the Wi-Fi Aware hardware, joining or forming a cluster, establishing a subscribe and publish session and establishing a data path.

Figure 4.1 displays the interface presented to the users showing the peers available for communication in the cluster. Selecting a MAC address from the list will launch the Chat Activity and a TLS connection to that user will be initiated. The owner of the MAC address will receive a notification stating that another user wishes to communicate. Also, the Main Activity handles PA functionality and enables users to request authentication before starting to chat with a peer.

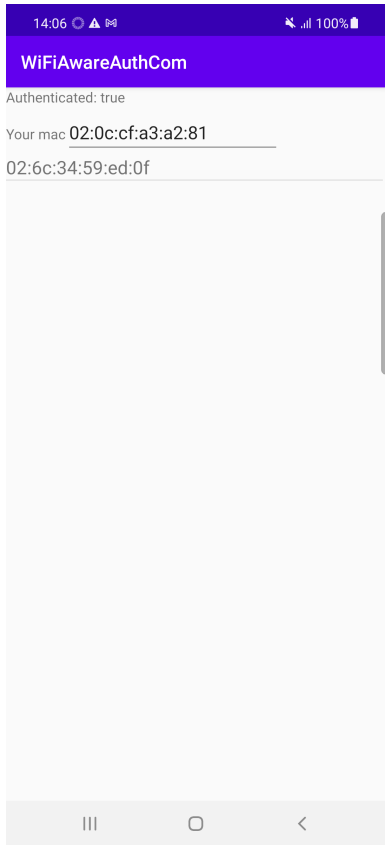


Figure 4.1: The Main Activity interface.

4.2.2 Chat Activity

A TLS connection between two peers is initiated when the Chat Activity is launched. The device that first launches the Chat Activity becomes the client in the connection. If a mutually authenticated connection can take place, messages are exchanged and will appear in chat bubbles. This applies to users with certificates signed by the CA and users who are verified by another peer. The chat view does not differ. Figure 4.2 displays the user interface, in which the purple messages are the messages sent by the user and the white messages are those messages that are received. The messages are displayed in chronological order. To send a message, the user has to tap the message field on the bottom, then click on the send button.

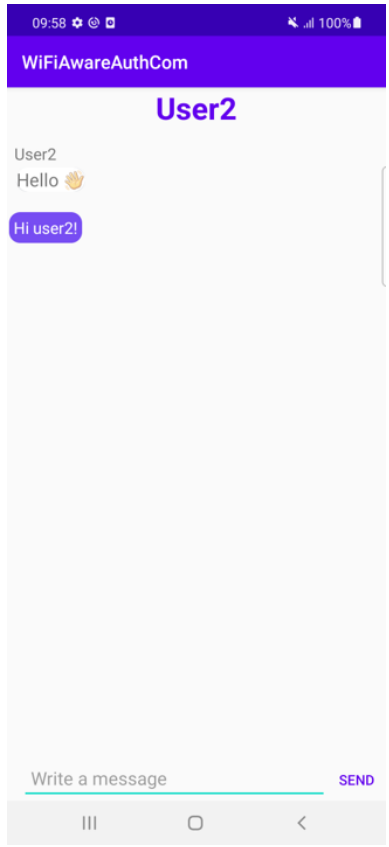


Figure 4.2: The Chat Activity interface.

4.2.3 Peer Authentication

If a user has a self-signed certificate, the authentication status is displayed in the top left corner in Main Activity. Figure 4.3 shows a user with a self-signed certificate that has not yet been verified by a peer. This user can request authentication credentials from a peer in the list by clicking on an address. By utilizing Wi-Fi Aware short messages in the proof-of-concept application, short messages of a length of up to 255 bytes can be exchanged. This enables devices to exchange public keys and signed messages in order to receive PA.

When an authenticated user receives an authentication request, it is presented with an option box that enables them to decline or accept the request. The option box displays the request made by a MAC address as shown in Figure 4.4. By answering "YES", the authentication credentials are returned to the unauthenticated user and the TLS server is started. As a result, the PA display is changed and the user is

notified about the authentication, see Figure 4.3.

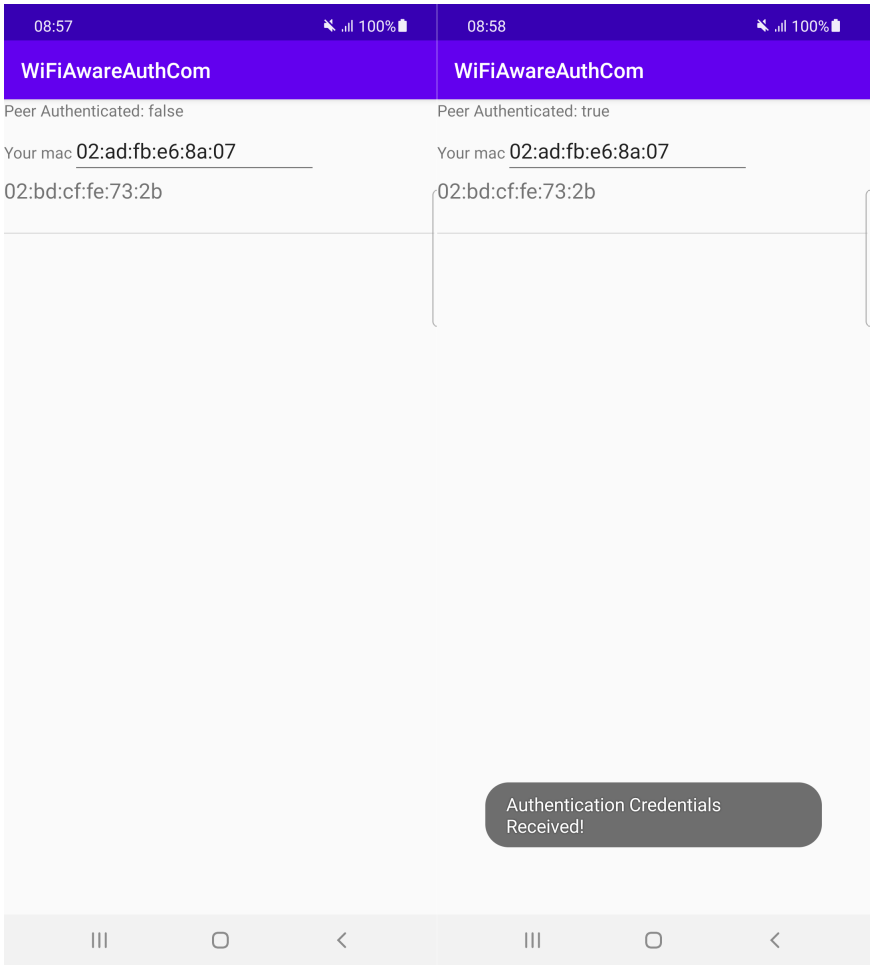


Figure 4.3: Display showing the status of an unauthenticated user before (left) and after (right) PA.

The signed key is stored in internal storage by both the authenticator and the peer-authenticated user. It is used as part of the PA credentials and sent during requests for PA TLS connections. If the credentials provided in the connection request are not valid because the user has no knowledge or does not trust the peer that issued the authentication credentials, it is given the option to authenticate the user itself. An option box will be presented to the user and the same procedure as above takes place. In order to simplify the implementation, only the peer-authenticated user is able to request the start of the PA server in the proof-of-concept application.

In the decentralized authentication scheme presented by Santos-González et al. [24], the unauthenticated user is provided with the certificate of the user authenticating it. However, the certificates used in the application are about 700 bytes, which is too large to send using Wi-Fi Aware short messages. Instead, the public key of authenticated users' are stored in internal storage the first time two devices make an authenticated connection. The public keys used are around 124 bytes. The list of authenticated users is exchanged with other authenticated users upon contact. An option is to divide the certificate into separate parts and send it piece by piece using Wi-Fi Aware short messages. However, messages could be lost or sent in the wrong order, making it difficult to reassemble the certificate [39].

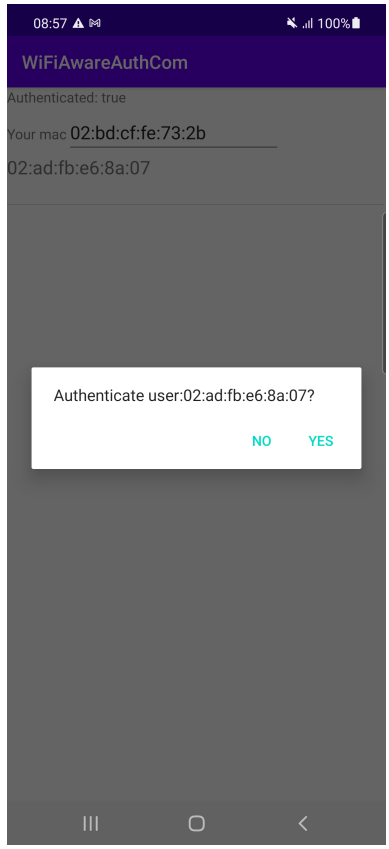


Figure 4.4: Option box including the MAC address of the user requesting authentication, presented to the verifyer.

4.2.4 Cryptography

The cipher suites selected, ChaCha20 with Poly1305 and AES 128 GCM, were hard-coded into the application in order to test which would perform better when it came to sending messages. In order to choose the cipher suite in Android Studio, the TLS version had to be downgraded to version 1.2 since cipher suites cannot be customized when using version 1.3 [40].

Chapter 5
Results

This chapter aims to experimentally validate the proposed solution by performing experiments and analyzing the operation of the proof-of-concept application. Network traffic between devices was captured to validate the security provided by Wi-Fi Aware and examine its performance. In addition, Android Studio logs were examined and presented to validate the security mechanisms that were proposed and implemented in the application. The results are presented according to the layers of the OSI model in a bottom-up order.

Additional experiments were conducted to measure the performance of Wi-Fi Aware technology and the additional security measures implemented in the application. This include measuring the time used to connect to a Wi-Fi Aware network, time used to send messages using the two proposed cipher suites, and time used to set up a PA connection. Finally, the alternation of the Anchor Master (AM) role depending on battery levels was studied in order to analyze the battery distribution principle stated by the Wi-Fi Alliance.

5.1 Security

5.1.1 Link Layer Security

The results from this section were produced using a USRP B200mini to capture data frames sent between Samsung Galaxy A71 and S9 devices. Frame capture was performed on channel 149 in the 5 GHz frequency band and channel 6 in the 2.4 GHz frequency band as these are the channels on which Wi-Fi Aware operates [5]. Wireshark was used to analyze and present the findings of the captured frames. It is worth noting that Wireshark recognizes and displays Wi-Fi Aware frames as NAN frames.

Figure 5.1 shows a Data frame containing the Wi-Fi Aware Interface addresses of two communicating parties. This means that the Wi-Fi Aware Interface addresses are

visible to any third party listening in on the correct channel. These addresses conform to the MAC-48 convention. Frames broadcast by a device running continuously over a long period were captured. It was observed that the addresses changed approximately every 30 minutes, as well as whenever the application was restarted.

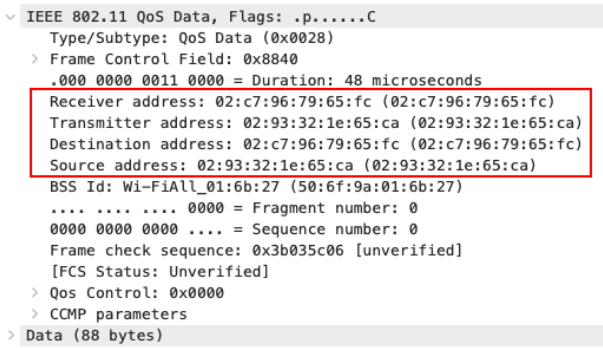


Figure 5.1: Wi-Fi Aware QoS Data frame showing the Wi-Fi Aware Interface addresses of the receiver and the transmitter.

The device that first started the proof-of-concept application created the cluster and therefore also selected the Basic Service Set Identifier (BSSID). A beacon frame containing the BSSID can be seen in Figure 5.2. The BSSID conforms to the MAC-48 convention and uniquely identifies the address of the Wi-Fi Aware cluster. It was noted that the BSSID persisted, even though when the device that started the cluster left. The device that took over as the AM of the cluster used the same BSSID to advertise the cluster's capabilities. It is worth noting that unlike traditional APs, BSSID is not the same as the MAC address of the device creating the cluster. A manual check of the Network Interface Card (NIC) of the device against the BSSID in Figure 5.2 confirmed this. A device leaving the cluster was emulated by closing the application on the device. As long as one device remained in the cluster, the address was the same.

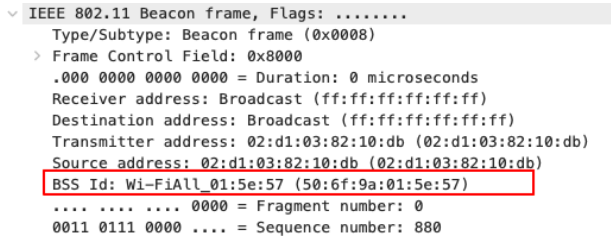


Figure 5.2: Wi-Fi Aware Synchronization Beacon frame showing the BSSID of the cluster.

It was noted that the Samsung Galaxy A71 model generated a unique BSSID each time it created a cluster, while the S9 model used the same BSSID. Figure 5.3 shows the same BSSID used to create two different clusters. A new cluster was created by restarting the application on all the devices.

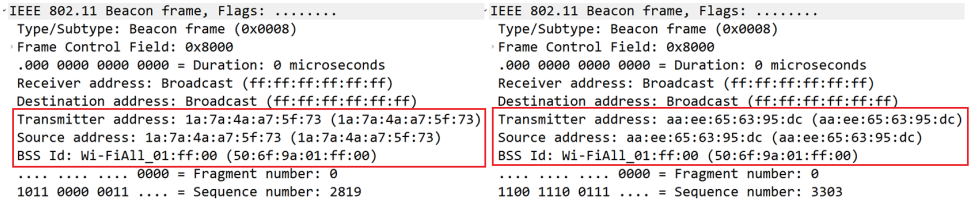


Figure 5.3: Wi-Fi Aware Synchronization Beacon frames sent from a Samsung Galaxy S9, showing the same BSSID used for two different clusters.

A WPA2 protected data path with pairwise security association was established during the 4-way handshake, as shown in Figure 5.4. This corresponds with the Wi-Fi Aware Specification regarding how a secure data path is established [5]. During the setup, a symmetric key was established from the PSK. The PSK was specified during implementation and installed on the devices. After the handshake, the established symmetric key was used in encryption to protect the Data frames, as shown in Figure 5.5. A third party capturing a handshake cannot decrypt messages without having access to the PSK from the application's source code.

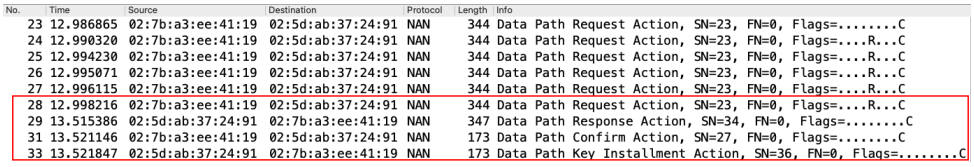


Figure 5.4: Wi-Fi Aware handshake showing the data path establishment and the installation of keys.

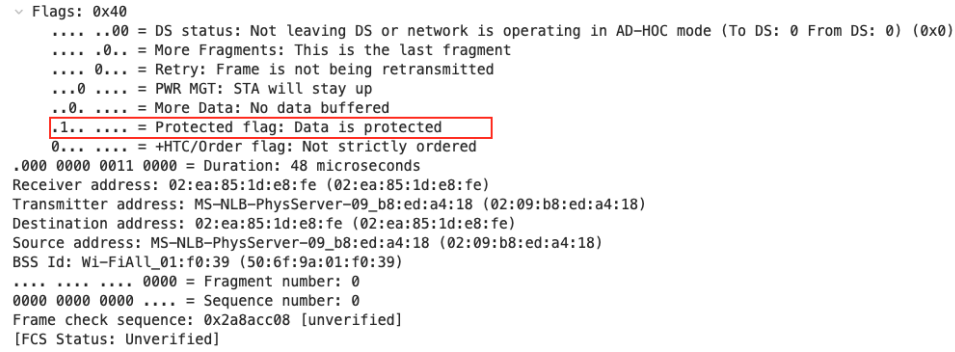


Figure 5.5: Protected flag set in QoS Data frame after the 4-way handshake.

Unlike Wi-Fi Direct, Wi-Fi Aware does not use Wi-Fi Protected Setup (WPS) where the AP manually confirms incoming connections to the network. Anyone with the same Wi-Fi Aware application can discover, connect and set up encrypted data paths using the PSK installed on the device. The observation that all devices running the proof-of-concept application were able to join clusters and set up networks validates this.

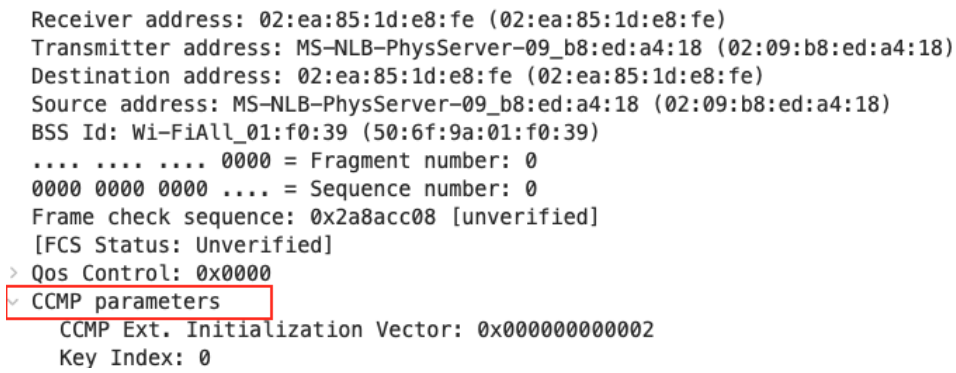


Figure 5.6: Capture of frame were CCMP encryption protocol is enabled.

The captured packet in Figure 5.6 shows that the CCMP encryption protocol was enabled after a complete handshake. The CCMP protection was present in Data frames and Action Management frames.

5.1.2 Transport Layer Security

The link layer alone is not sufficient to provide verification of users' identities because of the lack of authentication mechanisms in Wi-Fi Aware. Thus, authentication of users is provided using the mTLS protocol. The TLS protocol ensures that messages cannot be tampered with, in both mTLS and PA communication.

A high level of security is achieved by forcing every device to use the cipher suites recommended in TLSv1.3, as previously explained in Section 3.3. If a device lacks support for the cipher suites available or does not support the necessary TLS version, a communication request will be denied. Figure 5.7 shows a failed handshake due to the client using TLSv1.1 instead of TLSv1.2, as required by the server. A client or server without a valid certificate in an mTLS connection will also result in the handshake failing.

```
javax.net.ssl.SSLHandshakeException: Read error: ssl=0x7157e9bed8: Failure in SSL library, usually a protocol error
error:100000f0:SSL routines:OPENSSL_internal:UNSUPPORTED_PROTOCOL (external/boringssl/src/ssl/ssl_versions.cc:321 0x70a86e805a:0x00000000)
    at com.android.org.conscrypt.SSLUtils.toSSLHandshakeException(SSLUtils.java:362)
    at com.android.org.conscrypt.ConscryptEngine.convertException(ConscryptEngine.java:1134)
    at com.android.org.conscrypt.ConscryptEngine.unwrap(ConscryptEngine.java:919)
    at com.android.org.conscrypt.ConscryptEngine.unwrap(ConscryptEngine.java:747)
    at com.android.org.conscrypt.ConscryptEngine.unwrap(ConscryptEngine.java:712)
    at com.android.org.conscrypt.ConscryptEngineSocket$SSLInputStream.processDataFromSocket(ConscryptEngineSocket.java:858)
    at com.android.org.conscrypt.ConscryptEngineSocket$SSLInputStream.access$100(ConscryptEngineSocket.java:731)
    at com.android.org.conscrypt.ConscryptEngineSocket.doHandshake(ConscryptEngineSocket.java:241)
    at com.android.org.conscrypt.ConscryptEngineSocket.startHandshake(ConscryptEngineSocket.java:220)
    at com.android.org.conscrypt.ConscryptEngineSocket.waitForHandshake(ConscryptEngineSocket.java:572)
    at com.android.org.conscrypt.ConscryptEngineSocket.getInputStream(ConscryptEngineSocket.java:296)
    at com.example.testaware.AppServer.lambda$new$0$AppServer(AppServer.java:150)
    at com.example.testaware.-$$Lambda$AppServer$hZzppmJFtF9DPJe3vM_ME9M9JCA.run(Unknown Source:6)
    at java.lang.Thread.run(Thread.java:923)
Caused by: javax.net.ssl.SSLProtocolException: Read error: ssl=0x7157e9bed8: Failure in SSL library, usually a protocol error
error:100000f0:SSL routines:OPENSSL_internal:UNSUPPORTED_PROTOCOL (external/boringssl/src/ssl/ssl_versions.cc:321 0x70a86e805a:0x00000000)
    at com.android.org.conscrypt.NativeCrypto.ENGINE_SSL_read_direct(Native Method)
    at com.android.org.conscrypt.NativeSsl.readDirectByteBuffer(NativeSsl.java:568)
    at com.android.org.conscrypt.ConscryptEngine.readPlaintextDataDirect(ConscryptEngine.java:1095)
    at com.android.org.conscrypt.ConscryptEngine.readPlaintextData(ConscryptEngine.java:1079)
    at com.android.org.conscrypt.ConscryptEngine.unwrap(ConscryptEngine.java:876)
```

Figure 5.7: TLS Handshake failure caused by a client using the wrong TLS version.

5.1.3 Application Layer Security

A client requesting PA credentials or a PA connection is verified in the application layer. Instead of mutual authentication being performed in the transport layer, as in mTLS, the client is authenticated using the cryptographic schemes implemented in the proof-of-concept application. Figure 5.8 shows the successful verification of a peer. The server is started and accepts incoming connections from the recently verified peer.

An authenticated peer will only start the PA server if the authentication credentials provided are valid and signed by a known authenticator, as shown in Figure 5.9.

```
D/Log-Main: Signed string : MEUCIQDttPf9/YeKLPkEg49S8xQTPipF8Q4nWh1uOLP2g
D/Log-Main: random String: eamVnGB
I/Dialog: mIsSamsungBasicInteraction = false, isMetaDataInActivity = fals
I/Log-Test-Aware-Verify-Credentials: Signed string is valid!
I/Log-Test-Aware-Peer-Signer: Signing Peer Key
I/Log-Test-Aware-Verify-User: Saving key to file
D/Log-Main: Starting PA server
D/Log-Test-Aware-No-Auth-App-Server: User is PA. Accepting connection
D/Log-Client-handler: inputstream ClientHandler
```

Figure 5.8: Android Studio log showing an authenticated user verifying a peer. The signature was valid so the PA server could start and accept the connection.

```
/Log-Main: Peeripv6: /fe80::73:d5ff:fe74:59c4%aware_data0
/Log-App-Server: Port: 0
/Log-Main: onCapabilitiesChanged with Network 617
/Log-Main: Peeripv6: /fe80::73:d5ff:fe74:59c4%aware_data0
/Log-Main: authenticator key MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE5eOhhadkoxDc8Ywf120Lr6
/Log-Main: Signed string : MEQCIGNZSyz6zXfZiVDI95kMTUBL9Xiyt4F+8I0CqtniS8cCAiAHEpvHckHe
/Log-Main: random String: tnQKtSa
/Log-Test-Aware-Verify-Credentials: Signed string is valid!
/Log-Test-Aware-InitPeerAuthConn: No match for authenticator key. Not starting server.
```

Figure 5.9: Android Studio log showing that the PA server did not start because the credentials provided were not signed by a known authenticator.

5.2 Performance

To evaluate the proposed solution, several aspects were tested and measured. This include the discovery and connectivity delay of Wi-Fi Aware, the performance of the proposed PA scheme, the time it took to send messages with mTLS with AES and ChaCha20 separately, and the alternation of the AM role according to battery levels. The following sections contain the measured results and observations from experimental trials, involving devices running the proof-of-concept application. In

experiments that required multiple application runs, a bash script interacting with the devices through ADB was used.

5.2.1 Wi-Fi Aware Connectivity

The Wi-Fi Aware connectivity performance was tested using two Samsung Galaxy A71 smartphones and the proof-of-concept application. The application was restarted 500 times on each phone. The application was first started on one device, followed by a few seconds delay, and then started on the second device. This allowed the first device to create a cluster before the second device joined. After the devices had discovered each other and a connection had been made, the application was restarted on both devices. The measured times were obtained by writing them to a file on the second device.

The main Wi-Fi Aware events leading up to connectivity were measured and isolated, as shown in Table 5.1. The time required to start the application was also included. Starting the application includes the time used to set up the SSLContext [1] for each run. The SSLContext handles device keys and the certificate necessary for establishing an mTLS connection. Restarting the application between each use and setting up an SSLContext is not expected user behavior. Thus, the total time is somewhat less for normal use. The service discovery time shows the time used to discover a published service. The connectivity time shows the time from the moment the service is discovered until a secure Wi-Fi Aware data path was set up between the devices, see Figure 5.4.

Table 5.1 presents statistical summaries for total connectivity time deconstructed into separate events. Connectivity is the most time-consuming event with a mean of 2.63 seconds. There was considerable variation in connectivity time across the experimental runs. The mean value exceeded the median, indicating a right skewness in the distribution of connectivity times. The connectivity time included the time used to perform the 4-way handshake. Observations made analyzing captured Wi-Fi Aware frames showed that it could take between 0.22 and 4.75 seconds to perform a handshake or parts of a handshake. The second most time-consuming event was starting the application. It can be seen that the time it took to start the application is almost constant across the 500 runs, with a Standard Deviation (SD) of 0.01 seconds. Starting the application included setting up an SSLContext, which takes around 1.17 seconds. The third most time-consuming event was the service discovery, which presented a mean value of 0.64 seconds and a SD of 0.14 seconds. Finally, the time it took to attach to a cluster is close to negligible.

[1] SSLContext, https://developer.android.com/reference/javax/net/ssl/SSLContext

	Mean	Median	SD
Starting the application	2.46	2.46	0.01
Attach to cluster	0.08	0.09	0.02
Service discovery	0.64	0.62	0.14
Connectivity	2.63	2.53	0.93
Total time	5.81	5.69	0.95

Table 5.1: Statistical summaries of Wi-Fi Aware connectivity times (seconds), across 500 runs.

A further analysis is provided by visualizing the time used to discover a service and establish a connection to another peer. The Cumulative Distribution Function (CDF) plot from Figure 5.10 shows that the discovery time was almost constant, with the exception of a few large outliers. This corresponds to the right skewness of the distribution previously mentioned. The six individual runs that caused the right skewness in the discovery times can be identified from the highest spikes in Figure 5.11. There was a low autocorrelation (lag-1) of 0.018, meaning that there is no evident pattern between two subsequent runs. This indicates that the spikes in discovery delays occur at random.

The variation in connectivity times was much greater than for discovery times, see Figure 5.10. This process can take up to 8 seconds. It can also be seen that the figure has a few plateaus, which correspond to an aggregate of connection times of around 2 seconds and 2.6 seconds, respectively and Figure 5.12 confirms this. The frequency of the aggregated times is similar. Figure 5.12 also shows that the distribution is right skewed, meaning there are runs that take significantly more time than the mean. Figure 5.13 shows a scatter plot of two subsequent runs. There were no occurrences of two long subsequent connection times. This yields a negative autocorrelation (lag-1) of -0.078. However, no formal statistical test has been conducted, so it cannot be claimed that the negative autocorrelation is statistically significant.

To investigate whether it was a dependency between the discovery and connectivity times within a run, the correlation between the two data sets were calculated. The correlation was found to be 0.025, which means that there is no dependency.

An additional experiment was performed to further investigate the variation in connectivity times. The experiment was carried out in the same way as before, but information about the subscriber/publisher session role was also recorded. The experiment showed that the device was acting as publisher in approximately 50% of cases, and subscriber in the remaining cases. Figure 5.14 explains the aggregate of times around 2 and 2.6 seconds, as previously mentioned. When the device serves as

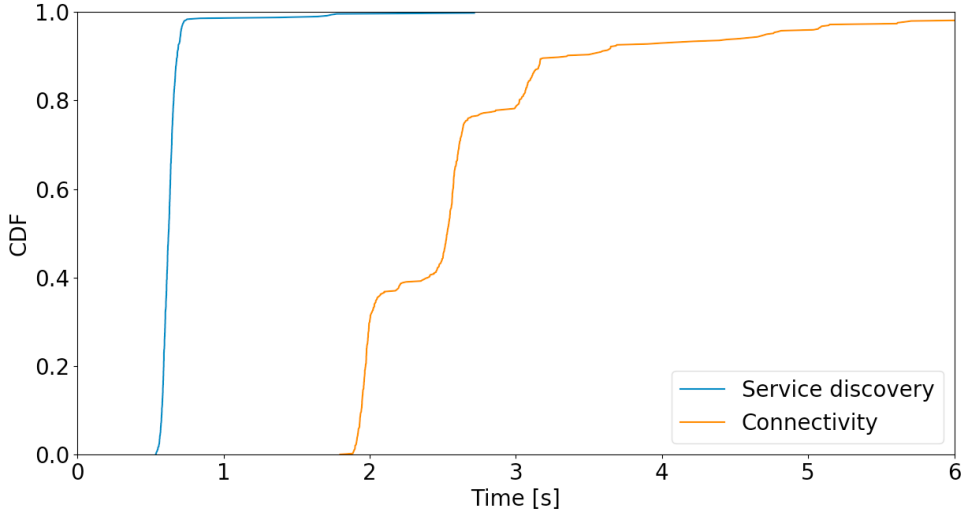


Figure 5.10: CDF plot of service discovery and connectivity times (seconds), across 500 runs.

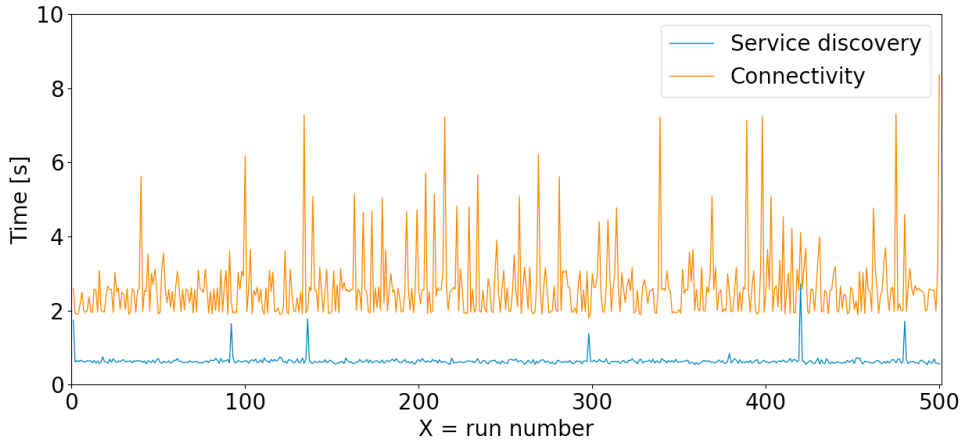


Figure 5.11: Discovery and connectivity times (seconds), across 500 runs. Shown by run.

the subscriber, it uses less time to establish connectivity.

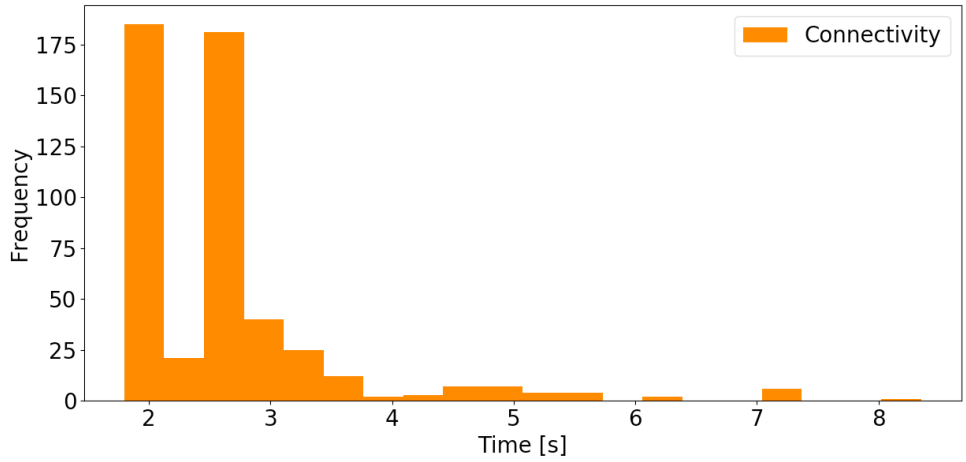


Figure 5.12: Histogram of connectivity times (seconds), across 500 runs.

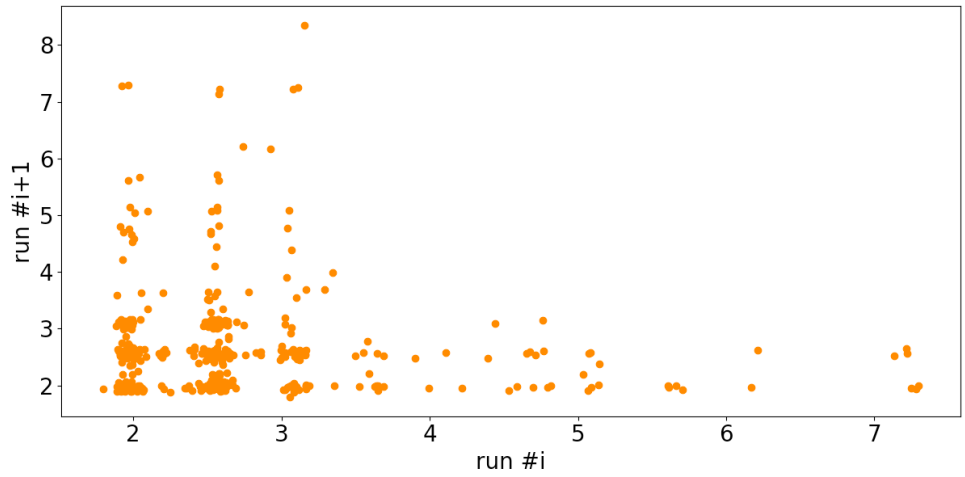


Figure 5.13: Scatter plot of connectivity times (seconds) of two subsequent runs, across 500 runs.

5.2.2 Peer Authentication Connectivity

The performance of the PA scheme was evaluated based on the time it took to receive PA and the time it took to request and start the PA server. The time was measured and recorded across 500 runs. Three Samsung Galaxy A71 devices were used in the experiment. The prerequisites for the experiment was that a data path had been

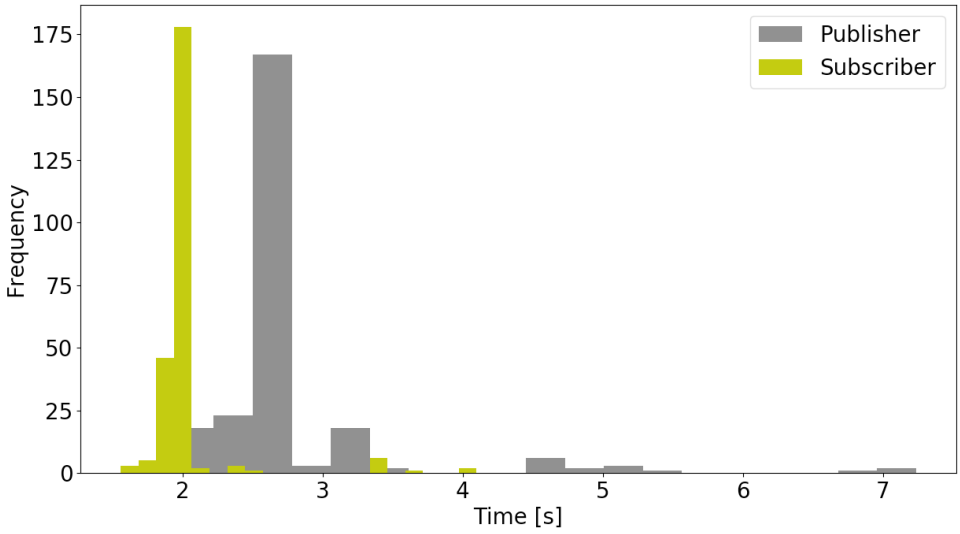


Figure 5.14: Histogram of connectivity times (seconds) depending on session role, across 500 runs.

established between the communicating devices. For testing purposes, all requests and responses were automated, and no user input was required. All times were recorded on the peer-authenticated device.

Table 5.2 shows statistical summaries calculated from the data collected during the experiment. The verify peer event was measured from the time of the authentication request until the user received the authentication credentials. The average time used, a mean of 1.78 seconds, is significant.

The request connection event is the time it took to request and receive a response that the PA server had started and was ready for communication, as shown in Table 5.2. The average time used, a mean of 2.32 seconds, is considerable. The event is required every time a peer-authenticated user wants to communicate with another peer and is therefore added to the total connectivity time.

	Mean	Median	SD
Verify Peer	1.78	1.77	0.10
Request Connection	2.32	2.24	0.42

Table 5.2: Statistical summaries of the time (seconds) used to verify a peer and request a PA connection, across 500 runs.

5.2.3 Alternation of the Anchor Master Role

According to the Wi-Fi Aware Specification [5], the AM Role alternates between devices in order to evenly distribute resource usage. In order to validate this statement, a USRP B200mini was used to capture the Wi-Fi Aware packets. The packets were analyzed in Wireshark. The testing was be divided into two scenarios, in which scenario 2 comprises of three parts. For the first scenario, four identical Samsung Galaxy A71 smartphones were used. The same four phones were used during the second scenario, with the addition of the Samsung Galaxy S9 smartphone.

Scenario 1: Four Samsung Galaxy A71s

In the first scenario, the devices had different battery levels. One device was also connected to a power source. The capture from Figure 5.15 shows the Synchronization and Discovery Beacon frames transmitted by the AM, broadcasting information about the newly established cluster. The device that created the cluster started as the AM with both Master Preference (MP) and Random Factor (RF) set to 0, as shown in Figure 5.16. It can also be noted that the Hop Count to AM was zero, which verifies that this device was actually the AM. When the second device joined, it assumed the role of AM because it had a higher MP value, as shown in Figure 5.17. Observations showed that the second device to join always had the Master Rank (MR) set higher than the device that established the cluster, with MP and RF set to zero.

```
2 0.144314   02:5e:a1:46:78:c7   Broadcast   NAN   98 Discovery Beacon frame, SN=41, FN=0, Flags=........, BI=100
3 0.864280   02:5e:a1:46:78:c7   Broadcast   NAN   98 Discovery Beacon frame, SN=50, FN=0, Flags=........, BI=100
4 1.324794   02:72:97:98:6e:ad   Broadcast   NAN   98 Discovery Beacon frame, SN=3, FN=0, Flags=........, BI=100
5 2.085970   02:5e:a1:46:78:c7   Broadcast   NAN   98 Sync Beacon frame, SN=57, FN=0, Flags=........, BI=512
```

Figure 5.15: Discovery and Synchronization Beacons broadcast by the AM.

By examining the frames in Wireshark when all four devices were connected, it was clear that all devices had the MP set to one. Because the MAC address was the same on each device during the experiment, the only factor that affected the MR value was the RF.

Scenario 2: Four Samsung Galaxy A71s and one Samsung Galaxy S9

Scenario 2 included five devices, four A71 devices and one S9. Three experiments were conducted using this setup. In the first experiment, the five devices were fully charged. The second experiment was performed similarly but the S9 was connected to a power source. Both these experiments were conducted once each. For the last experiment, a more in-depth study of how the MP was affected by the battery level on the S9 was conducted.

```
NAN protocol
  Master Indication Attribute
    Attribute Type: Master Indication Attribute (0)
    Attribute Length: 2
    Master Preference: 0x00
    Random Factor: 0
  Cluster Attribute
    Attribute Type: Cluster Attribute (1)
    Attribute Length: 13
    Anchor Master Information
      Anchor Master Rank: 170751159932747776 (0x025ea14678c70000)
      Hop Count to Anchor Master: 0
      Anchor Master Beacon Transmission Time: 0x00000000 (0)
```

Figure 5.16: MP and RF of the device that initiates the cluster.

```
NAN protocol
  Master Indication Attribute
    Attribute Type: Master Indication Attribute (0)
    Attribute Length: 2
    Master Preference: 0x01
    Random Factor: 212
  Cluster Attribute
    Attribute Type: Cluster Attribute (1)
    Attribute Length: 13
    Anchor Master Information
      Anchor Master Rank: 170751159932747776 (0x025ea14678c70000)
      Hop Count to Anchor Master: 1
      Anchor Master Beacon Transmission Time: 0x49004000 (1224753152)
```

Figure 5.17: MP and RF of the second device to join the cluster.

In the first two experiments, the application was first started on the four A71 devices, then on the S9. An examination of the captured frames in Wireshark showed that the S9 took over as the AM after it had connected to the cluster in both experiments. The MP was 0x76 in the first experiment and 0x61 in the second.

The last experiment investigated how different battery values affected the MP by recording the MP values for the S9 at two different battery levels. In order to record the values of a fully charged device, 100% battery level, the charger was connected for the entire experiment. The second values were recorded when the device started at 50% battery level decreasing to 40%, since it was not connected to a power source. The application was restarted 100 times on the S9, while on the four A71 devices it was started once and kept running throughout the experiment.

	Mean	Median	SD
100% battery level and connected to a power source	56.7	51.0	35.2
40% to 50% battery level	63.2	63.0	37.3

Table 5.3: Statistical summaries of the MP, value from 0 to 255, for a Samsung Galaxy S9, across 100 runs.

The average MP value for the device connected to a power source was lower than the average MP value for the device starting at 50% battery level, see Table 5.3. The average MP was higher for a device with less battery, and not the opposite. The variation was significant for both parts of the experiment and the SD of the averages is 3.52 and 3.73, as shown in Figure 5.18.

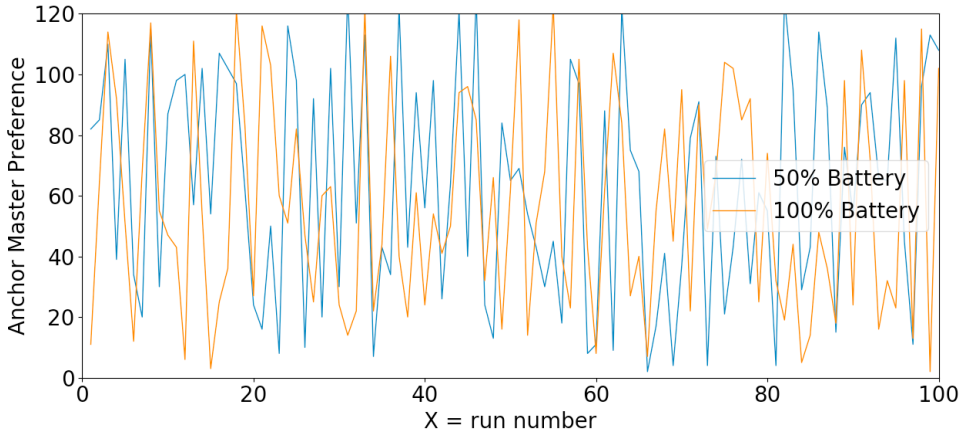


Figure 5.18: Recorded values (decimal) for the change of the MP value for the Samsung Galaxy S9.

5.2.4 Message Times with mTLS

The time it takes to send messages between two devices with mTLS was measured, using both AES in GCM and ChaCha20 Poly1305 as cipher suites for comparison. The system clocks on the phones were not synchronized. Thus, it was difficult to accurately measure the time it takes to send a single message. Instead, two messages were exchanged between the devices and the round trip time of the messages was measured. The same device measured the transmission time by measuring how long it took to send one message and then receive one back. Consequently, the time it took to send one message is half the time it took to send a message and receive

a message back. The measuring was carried out using two Samsung Galaxy A71 phones. The experiment included sending both short 3-byte and long 32,680-byte messages. A prerequisite for the experiment was that an mTLS connection had been established.

The time it takes to send two long messages using ChaCha20 as an encryption scheme, was slightly less than the time it takes with AES, see Table 5.4. However, a two-sided t-test gives the p-value 0.45, making the findings statistically insignificant. It cannot be concluded that there were significant differences in the times.

The average time it takes to send two short messages between two devices was also slightly less for ChaCha than for AES. A two-sided t-test gave a p-value of $7.37 \cdot 10^{-6}$, which is very statistically significant. The mean of ChaCha was 0.083 seconds and the mean of AES was 0.097 seconds. Hence, this confirms the hypotheses that messages sent using ChaCha perform better.

	Mean	Median	SD
Total time ChaCha20 Poly1305	0.185	0.179	0.021
Total time AES in GCM	0.192	0.180	0.095

Table 5.4: Statistical summaries of the time (seconds) it takes to send two long messages of 32,680 bytes with an mTLS connection, across 100 runs.

	Mean	Median	SD
Total time ChaCha20 Poly1305	0.083	0.072	0.027
Total time AES in GCM	0.097	0.094	0.017

Table 5.5: Statistical summaries of the time (seconds) it takes to send two short messages of 3 bytes with an mTLS connection, across 100 runs.

Chapter 6
Discussion

The following sections will present the limitations and advantages of the proposed solution by using the results obtained from the experiments and observations made using the proof-of-concept application.

6.1 Security

The first research question in the introduction is as follows: "Is Wi-Fi Aware a suitable connectivity technology in terms of ensuring confidentiality, integrity and availability for an instant messaging application in areas in which internet connectivity is unavailable?". The extent to which the chosen technology is suitable for instant messaging is discussed throughout this chapter.

A solution to the research question "How can users be verified and participate in authenticated communication, for both online and offline scenarios?" is presented in Chapter 3. The limitations and vulnerabilities of the proposed authentication solution are discussed.

As shown in the results in Section 5.1.1, Wi-Fi Aware can be used to establish a data path, thereby ensuring the availability of an instant messaging service in out-of-coverage areas. A data path is established by using a session obtained from either a subscribed or a published service. Neither Android Developer nor Wi-Fi Alliance specify best practices on how a device can be both a publisher and a subscriber of the same service simultaneously. Use of the proposed and implemented solution, described in Section 3.2.4, avoids the need to establish two data paths. The devices decide which session should be used to establish a data path based on the two MAC addresses involved. The use of two paths as redundancy was regarded as being unnecessary due to the additional resource consumption.

A potential vulnerability regarding cluster formation and advertisements is the device's inability to distinguish between authentic and false Synchronization Bea-

con frames. If a malicious device were to transmit false Synchronization Beacons containing false information and disrupting the discovery process, it could result in a DoS attack. This could disrupt the entire network, thus denying users access to the service being sought. This would reduce the availability of the instant messaging service. No attempt has been to incorporate a way of preventing/detecting this in the solution.

User privacy is maintained by using dynamically changing Wi-Fi Interface addresses instead of fixed MAC addresses, as shown in Figure 5.1. These findings match the feature of using dynamic Wi-Fi Aware Interface addresses specified by the Wi-Fi Alliance [5]. Randomization of the address increases user privacy by preventing malicious users from logging and tracking the addresses of nearby users. Address tracking can result in identification of a user and can be used to keep track of which users are communicating. If a malicious user was able to correlate an address to an identity, addresses and identities would have to be mapped every 30 minutes, making it resource consuming and unlikely to occur.

The S9 model used in the experiment generated the same BSSID every time it created a cluster, introducing a potential breach of privacy. If an attacker were to gain access to the identity that generated the cluster, the BSSID could be used in any future session to map the address to an identity. Given that the findings are based on testing a limited number of devices, it is difficult to state whether this is a widespread issue.

The two models used in the experiment behaved differently regarding generation of the BSSID. The differences can be explained by different Wi-Fi Aware settings and configurations in different phone models. This is supported by findings regarding the different AM preference properties in the two models.

Confidentiality is one of the fundamental security objectives of the CIA triad, as presented in Section 2.1.1. The results in Section 5.1.1 show that confidentiality is provided by Wi-Fi Aware on the link layer by encrypting messages with a key established during a 4-way handshake. It could be argued that configuring WPA2 in the proof-of-concept application is redundant, as the confidentiality of the messages sent is also provided in the transport layer that uses TLS. However, this was implemented in order to observe and test the protection and performance of Wi-Fi Aware technology. Adding security measures to several layers of the OSI model is a common practice in network operation and ensures secure communication. If an attacker were to gain access to the PSK installed on the devices, it would be able to decrypt the link layer frame. Due to the TLS protocol implemented in higher layers, the attacker would still not be able to read the content of the message.

Observations made during the testing regarding access control showed that anyone

with access to the application can prove knowledge of the passphrase (PSK) and would therefore also be able to set up a protected data path to access the service. The Wi-Fi Alliance states that the communication is authenticated, but this is purely based knowledge of the passphrase. A user should not be considered trustworthy only because of its ability to join the network and set up an encrypted data path. A TTP does not authenticate a user's identity making it possible for man-in-the-middle attacks.

The shortcomings of Wi-Fi Aware were addressed by implementing authentication schemes in the transport layer and application layer. The TLS protocol was mainly added to provide authentication. However, the protocol also provides end-to-end encryption and integrity checks to packets sent, adding redundancy to the solution. Forward secrecy is provided in the solution by using ECDHE with ECDSA.

The proposed offline authentication scheme enables peers to verify other peers in a web-of-trust way. As shown in the results, a peer with a valid signature on a string can receive PA credentials if it can prove ownership of its public key. The peer-authenticated user can then participate in TLS connections if it can provide valid PA credentials.

There is no system to prevent a rogue user from accepting any PA request it receives. However, since an authenticator must sign the authenticated peer's credentials, it is possible to keep track of malicious users. A solution similar to the CRL, used in PKI architecture, could be used in the PA scheme. If a user is no longer trusted, it should be included in a list of keys that should not be trusted to sign a peer's public key.

Unauthenticated users depend on being authenticated by other users. If no peer wants to authenticate the user, it cannot participate in the communication. This denies it access to the service.

The design of the PA scheme was based on the assumption that most devices have been authenticated by the CA before going offline. Two devices that are peer-authenticated are unable to communicate because they only have the functionality to serve as the client in communication. This limitation reduces the user experience and availability if more than one of the devices participating in a cluster is verified by a peer. On one hand, this could deny users the ability to communicate. On the other hand, the PA scheme increases the total availability of the system by enabling users to be verified by other peers and thereby giving them the ability to communicate. It is assumed that most devices sign up to the service and that the PA scheme works as intended, which is as a backup solution to enable availability for unauthenticated users.

6.2 Performance

The service quality provided by Wi-Fi Aware used in the proof-of-concept application will be discussed in this section. The delay of key events in the network setup is used to answer the research question "How will Wi-Fi Aware perform as a connectivity technology in a secure communication application, compared to other related solutions regarding connectivity time and resource distribution across devices?". The performance of the PA system is also discussed in the following section.

6.2.1 Wi-Fi Aware

The spikes in the connectivity time, found in Figure 5.11, can be explained by the delay caused by the 4-way handshake. The handshake from Figure 5.4 took approximately 0.5 seconds. However, observations show that it could take between 0.22 and 4.75 seconds to perform a handshake. The difference is caused by having to retransmit frames, which also explains the large variation found in the data set.

A reason for the high number of retransmitted frames could be devices using a different Discovery Window (DW) to send and receive messages. A device is only awake for a short period of time and is not able to receive messages sent outside the DW. Frames are retransmitted until a response is provided, as shown in Figure 5.4. A device might not even wake up during the scheduled DW. This would explain why there are such a high number of retransmissions. However, expectations are that this behavior would affect the variation in the discovery delay, seeing as service discovery frames are transmitted during the DW. The time used to discover a service is more stable and less time consuming compared to the connectivity delay.

Synchronization Beacons are sent within the DW. If the device is somehow not able to receive these beacons, it will not receive the most recent information about a change of the DW.

Another reasonable explanation for the spikes in the connectivity delay and the large variation is the unstable nature of Wi-Fi Aware messages. Short messages are used to exchange MAC and IP addresses and are necessary for establishing a data path. These messages might not have been delivered or are received the wrong order. The frames must therefore be retransmitted before a handshake can occur, adding to the connectivity delay and variation. This could happen, according to the Android Developer pages [39].

An interesting finding is that most runs had a connectivity delay of either 2 or 2.6 seconds, as shown in Figure 5.12. The only factor that changed during each run of the experiment was which device initiated the 4-way handshake. An additional investigation showed that the connectivity delay was approximately 0.6 seconds

longer when the device is acting as a publisher, as shown in Figure 5.12. However, this implies that the second device participating in the communication, the device that created the cluster, acted as a subscriber. The total connectivity delay should be the same for each device and it cannot be concluded that being assigned the role of initiator leads to less connectivity delay.

The USRP B200mini was unable to capture the full handshake of every run. Knowing the total time of each handshake would have provided more insight into how much the handshake added to the connectivity delay.

It is worth nothing that because each device broadcast publish messages and listens for matching services, resource consumption is higher than for a device that is just listening for or sending broadcast messages. This adds to the total resource consumption of each device.

	Wi-Fi Aware			Wi-Fi Direct		
	Mean	Median	SD	Mean	Median	SD
Discovery	3.18	3.16	0.14	3.42	3.07	1.17
Connection	2.63	2.53	0.93	2.03	1.90	0.46
Total time total	5.81	5.69	0.95	5.45	5.00	1.54

Table 6.1: Statistical summaries of Wi-Fi Aware (left) and Wi-Fi Direct (right) discovery and connectivity delay (seconds), across 500 runs. SD is standard deviation.

The performance of Wi-Fi Aware as a connectivity technology regarding discovery and connection delay is similar to the observation of Wi-Fi Direct presented by Sigholt, Tola and Jiang [23]. A comparison of the statistical summaries of both solutions is presented in Table 6.1. Wi-Fi Direct discovery is the time used by a device to discover broadcasts sent by a group, and connectivity is the time used by the same device to join the group. To compare the performance on an equal basis, the Wi-Fi Aware discovery time is a summation of three events: starting the application, attaching to cluster and service discovery.

On average, the Wi-Fi Aware solution used 0.24 seconds less to discover a service than the Wi-Fi Direct solution. However, Wi-Fi Aware had a longer connectivity delay, making the average total time to set up a connection 0.36 seconds less for Wi-Fi Direct. This difference is probably unnoticeable in terms of user experience and the average total time of the two connectivity technologies compared well.

Sigholt et al. [23] noted that the total time used to discover and connect to a group could take up to 10 seconds in the edge cases. Considerable variation in connectivity times is also present in the Wi-Fi Aware solution, making the total time

to establish a network unpredictable.

The devices used in the Wi-Fi Direct solution experienced some difficulties discovering group broadcasts, causing the device to set up a separate group. This issue was not present in the experiment conducted with the Wi-Fi Aware application.

6.2.2 Peer Authentication

The verify peer event, a mean of 1.78 seconds, as shown in Table 5.2, is an additional step that must be performed by an unauthenticated user before it can participate in communication. Compared to signing up for a service, this cannot be considered a time-consuming event. Signing up for a service often requires users to verify ownership of an email address, in addition to providing in other personal information. The verify peer event only has to be performed once within a cluster. However, if the user joins a new cluster, in which the devices do not trust the verifier, the user must receive new authentication credentials. If a user joins and leaves clusters on a regular basis, the verify peer event would have to be performed frequently. This would make the total time for connectivity considerable.

The request connection event, a mean of 2.32 seconds, only needs to be performed by a peer-authenticated user, requiring additional time for communication compared to an authenticated user. Since the time used to request a connection is measured after a full data path has been set up, the total time to set up a connection is closer to 8 seconds. This can be a considerable amount of time.

Several cryptographic computations are made in order to verify the peer requesting the connection, which adds to the time. If the list of trusted authenticators is long, the process of searching and checking signatures would be computationally expensive and not suitable for resource-scarce devices. In addition, several short messages are exchanged in a request connection event.

6.2.3 Alternation of the Anchor Master Role

The results from scenario 1 in Section 5.2.3 confirm that the device with the highest MP value is assigned the AM role. The results also confirm that the device that created the cluster started with MP and RF both set to zero. The devices that joined the cluster always had these values set higher than zero. However, the results from both scenarios 1 and 2 show that the Samsung Galaxy A71's MP default value was set to one. Thus, the only variable that contributes to alternating the AM role is the periodically changing RF. This is consistent with how the MR value is calculated, as explained in Section 3.2.2.

The results from scenario 2 confirm that the device with the highest MP value works as the AM, taking into account that the Samsung Galaxy S9 device was assigned the AM role every time it joined a cluster during all three experiments. The experiments also indicate randomness in how the MP values are generated, see Figure 5.18. Surprisingly, the average value for the device with battery levels below 50% was higher than the average values for the device that was connected to a power source, as shown in Table 5.3. The results indicate that battery level has seemingly less relationship with the choice of the AM role. However, observations show that the device model affects the Wi-Fi Aware settings and behavior. It cannot be concluded that the same behavior is present in every device that uses the Wi-Fi Aware hardware, due to the limited number of device models included in the experiments.

Although the MP has not been calculated as expected, the results confirm that the AM role alternates between devices. Resource usage is distributed between devices, unlike the solution presented by Sigholt, Tola and Jiang [23], in which one device is responsible for broadcasting discovery information and managing connectivity among devices.

6.2.4 Messaging

As expected, the average time used to send long messages was longer compared to short messages. The average time was almost twice as much, indicating that the byte stream had to be split up and sent in separate frames. It is unlikely that an instant messaging application will be used to send messages with a length exceeding 32,680 bytes. Thus, the average time used to send short messages is therefore more representative in terms of instant messaging scenarios.

Conducting the experiment with a smaller message size gave a very small p-value. Based on the results, it could be concluded that ChaCha performs better than AES when it comes to sending short messages. The speed for AES128 in GCM is 1909.1 Mb/s on a 2 GHz Intel Core i7, compared to ChaCha20 Poly1305 with a speed of 625.2 Mb/s [41]. However, without support for AES in the hardware, ChaCha20 Poly1305 is much faster, with 130.9 Mb/s on a Snapdragon S4 Pro compared to AES with a speed of 41.5 Mb/s. Considering most phones do not include hardware support for AES in GCM [41], ChaCha would perform better when used in an instant messaging application.

Even though the average time of ChaCha is lower and the findings are statistically significant, the times are so similar that a user would be unable to detect any difference, performance wise, between the two cipher suites.

6.3 User Experience

Several simplifications have been made to the proof-of-concept application. Functionality and methods were primarily added in order to validate the proposed solution in regard to the research questions. Thus, the user interface lacks features that are common in instant messaging solutions.

Before a user has established a TLS connection with another user, the list of available devices will identify other peers by their Wi-Fi Aware MAC address. User names are saved with the certificate and only shared after the TLS handshake has been successfully established. This makes it difficult for users to know who is in the group without establishing a TLS connection or meeting up.

Another limitation regarding user experience relates to PA. As previously mentioned in Section 3.1.6, it is not possible for two peer-authenticated users to communicate. However, the list of peers in the vicinity does not indicate this. All peers are included in the list without anything indicating whether peers have been authenticated or have a valid certificate issued by a CA. A connection must be established before the application knows whether a peer has a valid certificate or has been peer-authenticated.

In order for peers to always be able to receive messages, they have a server continuously running after a network has been established. This is resource consuming for the device hosting the application. However, in order to achieve the functionality expected in an instant messaging application, in which all users receive and initiate connections, this was considered necessary.

Chapter 7
Conclusion

People are becoming increasingly used to always being connected to the Internet and sharing content such as messages and images. However, if a network goes down and people find themselves in an area with no connectivity, social applications that require access to a central server become unavailable. Using technologies that offers device-to-device connectivity enables users to communicate with users who are close by in internet-isolated locations.

The thesis presents a solution that provides secure instant messaging in out-of-coverage areas, using Wi-Fi Aware. Wi-Fi Aware enables devices to be aware of the services and information that are available to them as they move around and to establish a data path for secure communication if certain matching criteria are met. Measures to enable secure and authenticated communication are provided using mTLS certificates downloaded from an online authentication server. The solution also provides an alternative authentication scheme: PA, for users who have not signed up for the service before moving to an internet-isolated locations.

In order to validate the proposed solution, a simple chat application has been developed using Android Studio. The application offers secure communication with end-to-end encryption between communicating peers, in addition to authentication of users in both online and offline scenarios.

The performance and security features of the application have been tested and considered. Even though Wi-Fi Aware is a good alternative to Wi-Fi Direct in terms of performance and operation, the time used to set up a network between devices is significant. Additional results showed that Wi-Fi Aware uses a fair resource principle that alternates the responsibility of each device within a group. However, observations made showed that not all phones fully support this functionality. Thus, in order to benefit from this functionality in the future, the Wi-Fi Aware hardware on phones might need an upgrade.

The main drawback of the presented solution is the inability of peer-authenticated devices to send instant messages to other peer-authenticated devices. The decentralized authentication scheme is presented as a backup for use in emergency scenarios or for one-time use. If a user is unable to reach the authentication server, it can resort to the PA scheme for authentication. Taking into account the considerable verification and connection time, the PA scheme should not be used as a standard authentication method. It is assumed that once a user restores connectivity, it will sign up for the service and download a certificate.

Another drawback is that the solution does not provide forwarding of messages, meaning that users can only send messages to users within range of establishing a network connection. The solution does not provide upper layer measurements that enable one device to forward a message on behalf of another to a peer that is out of range of the transmitter.

7.1 Further work

Below are suggestions about how the proposed system in this thesis could be improved, in addition to alternative technologies that could be used to restore availability in out-of-coverage areas.

Additional work on the PA scheme could be to further investigate the use of certificates instead of public keys and different levels of trust. Similar to the web-of-trust scheme, a user would have to have a minimum trust level before being able to authenticate another user.

7.1.1 New features in Wi-Fi Aware

Wi-Fi Aware APIs are still being developed and new features are being added from API level S from Android version 12. One new feature is the option to create a network connection with multiple peers rather than having to create one for a specific peer. This makes it possible to implement group chat functionality. Another new feature is the option to measure the distance to other peers using a Wi-Fi Round Trip Time [42]. It can be used to limit the distance at which Discovery Beacons can be detected. In order to use this feature, the hardware must implement the 802.11-2016 FTM standard, which is not included in most phones. In addition to using Wi-Fi Aware features implemented in the APIs by Android Development, the Wi-Fi Aware Specification comprises many features and best practices that can be further explored.

7.1.2 TLSv1.3

TLSv1.3 includes several upgrades compared to TLSv1.2. One upgrade is a faster handshake with Zero Round-Trip Time key exchange, enabling clients to send data in the first message. However, this feature is not yet supported by Android Developer [40]. Cipher suites that do not support perfect forward security are no longer supported. Even though TLSv1.2 has been used in this project, only cipher suites supported by TLSv1.3 have been used. Also, the renegotiation of parameters and the generation of new keys is not supported by version 1.3, which prevents the risk of downgrade attacks.

7.1.3 Bluetooth Mesh

Bluetooth Mesh was introduced by the Bluetooth Special Interest Group (SIG) as a way of creating a large-scale network of devices. It uses the capabilities of traditional Bluetooth BR/EDR and the more recent low energy consumption standard Bluetooth Low Energy [43]. The Bluetooth standard introduces a many-to-many topology, which means that potentially tens of thousands of nodes can participate in the network [43]. If one node is damaged or leaves the network, the network simply reroutes the packets immediately. Currently, Android and IoS-operated devices do not natively support Bluetooth Mesh networking [44]. However, this could be an interesting topic to look at for further studies.

References

[1] W. Tang, C. Wu, L. Qi, X. Zhang, X. Xu, and W. Dou, "A WiFi-aware method for mobile data offloading with deadline constraints," *Concurrency and computation*, vol. 33, no. 7, pp. 1–1, 2021.

[2] A. Pyattaev, K. Johnsson, S. Andreev, and Y. Koucheryavy, "Proximity-Based Data Offloading via Network Assisted Device-to-Device Communications," in *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2013.

[3] S. Trifunovic, A. Picu, T. Hossmann, and K. A. Hummel, "Slicing the Battery Pie: Fair and Efficient Energy Usage in Device-to-Device Communication via Role Switching," in *Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks*, CHANTS '13, p. 31–36, Association for Computing Machinery, 2013.

[4] Wi-Fi Alliance, "Wi-Fi CERTIFIED Wi-Fi Aware™ Technology Overview," tech. rep., 2020.

[5] Wi-Fi Alliance, "Wi-Fi Aware Spesification version 3.2," tech. rep., 2021.

[6] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, Aug. 2018.

[7] I. Grigorik, "Transport Layer Security (TLS)." https://hpbn.co/transport-layer-security-tls/, 2013. Accessed: 2021-06-01.

[8] W. Stallings, *Cryptography and network security : principles and practice*. Pearson Education, 7th, global ed., 2017.

[9] M. Jin, J.-Y. Jung, and J.-R. Lee, "Dynamic power-saving method for Wi-Fi direct based IoT networks considering variable-bit-rate video traffic," *Sensors (Basel, Switzerland)*, vol. 16, no. 10, pp. 1680–1680, 2016.

[10] Bluetooth SIG, "Bluetooth Technology Overview." https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. Accessed: 2021-05-05.

[11] Wi-Fi Alliance, "How far does a Wi-Fi Direct connection travel?." https://www.wi-fi.org/knowledge-center/faq/how-far-does-a-wi-fi-direct-connection-travel. Accessed: 2021-04-05.

[12] Huawai, "Working range of Bluetooth." https://consumer.huawei.com/en/support/content/en-us00411008/. Accessed: 2021-05-05.

[13] Bridgefy, "How does the Bridgefy App Work?." https://bridgefy.me/how-does-the-bridgefy-app-work/. Accessed: 2021-05-04.

[14] L. P. Morrison, B. Team, B. Nguyen, S. Kannan, N. Ray, and G. C. Lewin, "AirChat: Ad hoc network monitoring with drones," in *2017 Systems and Information Engineering Design Symposium (SIEDS)*, pp. 38–43, IEEE, 2017.

[15] Signal Support, "Backup and Restore Messages." https://support.signal.org/hc/en-us/articles/360007059752-Backup-and-Restore-Messages. Accessed: 2021-05-06.

[16] Signal, "Is it private? Can I trust it? – Signal Support." https://support.signal.org/hc/en-us/articles/360007320391-Is-it-private-Can-I-trust-it-. Accessed: 2021-05-05.

[17] A. Bettison, "The Serval Project." http://developer.servalproject.org/dokuwiki/doku.php?id=content:about, 2013. Accessed: 2021-04-19.

[18] S. Mesh, "Serval Mesh - The Serval Project Wiki." http://developer.servalproject.org/dokuwiki/doku.php?id=content:servalmesh:main_page. Accessed: 2021-04-19.

[19] D. Camps-Mur, E. Garcia-Villegas, E. Lopez-Aguilera, P. Loureiro, P. Lambert, and A. Raissinia, "Enabling always on service discovery: Wifi neighbor awareness networking," *IEEE Wireless Communications*, vol. 22, pp. 118–125, 4 2015.

[20] K. Maeda, K. Sato, K. Konishi, A. Yamasaki, A. Uchiyama, H. Yamaguchi, K. Yasumoto, and T. Higashino, "Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation," in *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 151–158, 2005.

[21] M. Marlinspike, "Facebook Messenger deploys Signal Protocol for end-to-end encryption." https://signal.org/blog/facebook-messenger/. Accessed: 2021-05-06.

[22] K. Khacef and G. Pujolle, "Secure Peer-to-Peer communication based on Blockchain," in *Workshops of the International Conference on Advanced Information Networking and Applications*, pp. 662–672, Springer, 2019.

[23] L. Sigholt, B. Tola, and Y. Jiang, "Keeping Connected When the Mobile Social Network Goes Offline," Master's thesis, Norwegian University of Science and Technology, 2019.

[24] I. Santos-González, P. Caballero-Gil, J. Molina-Gil, and A. Rivero-García, "Decentralized Authentication for Opportunistic Communications in Disaster Situations," in *Ubiquitous Computing and Ambient Intelligence*, vol. 10586 of *Lecture Notes in Computer Science*, (Cham), pp. 558–569, Springer International Publishing, 2017.

[25] J. Huang and D. Nicol, "An anatomy of trust in public key infrastructure," *International Journal of Critical Infrastructures*, vol. 13, p. 238, 2017.

[26] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 5280, May 2008.

[27] Y. Sheffer, R. Holz, and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)." RFC 7525, May 2015.

[28] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols." RFC 8439, June 2018.

[29] D. McGrew, "An Interface and Algorithms for Authenticated Encryption." RFC 5116, Jan. 2008.

[30] A. Corbellini, "Elliptic Curve Cryptography: finite fields and discrete logarithms." https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/. Accessed: 2021-05-06.

[31] N. Smart, *Elliptic Curve Based Protocols*, p. 3–20. London Mathematical Society Lecture Note Series, Cambridge University Press, 2005.

[32] Eric Rescorla and Tim Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246, Aug. 2008.

[33] S. A. V. Alfred J. Menezes, Paul C. van Oorschot, "Handbook of Applied Cryptography," 2017.

[34] M. Al-Zubaidie, Z. Zhang, and J. Zhang, "Efficient and Secure ECDSA Algorithm and its Applications: A Survey," vol. 11, no. 1, pp. 7–35, 2019.

[35] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (GCM) of operation," in *Lecture notes in computer science*, vol. 3348, pp. 343–355, Springer, 2004.

[36] V. Krasnov, "It takes two to ChaCha (Poly)." https://blog.cloudflare.com/it-takes-two-to-chacha-poly/. Accessed: 2021-05-23.

[37] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols." RFC 7539, May 2015.

[38] A. Langley, W.-T. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)." RFC 7905, June 2016.

[39] Android developers, "Wi-Fi Aware overview: Send a message." https://developer.android.com/guide/topics/connectivity/wifi-aware#send_a_message. Accessed: 2021-05-29.

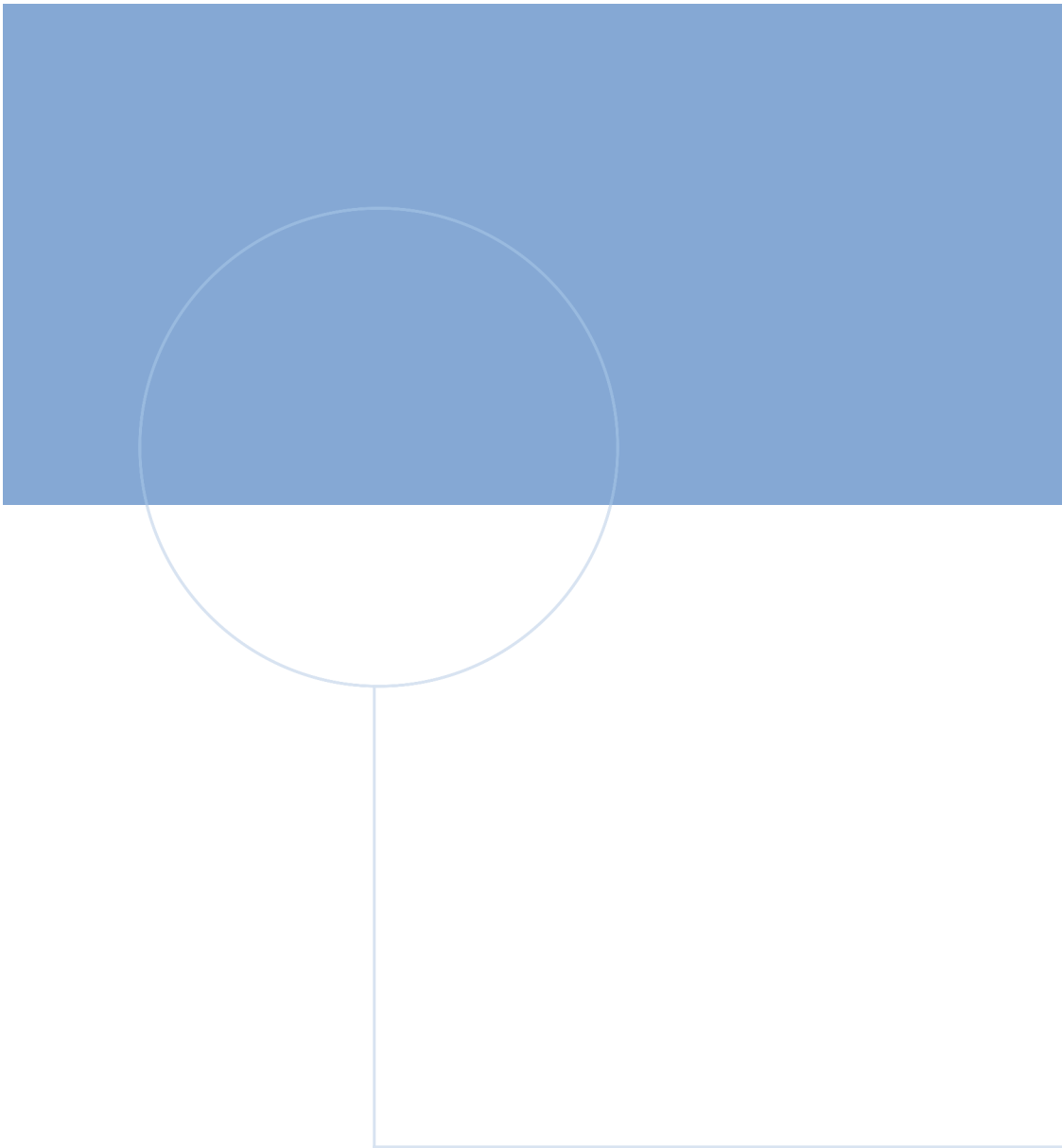
[40] Android Developers, “Behavior changes: all apps.” https://developer.android.com/about/versions/10/behavior-changes-all. Accessed: 2021-05-22.

[41] J. Mattsson, “Overview and Analysis of Overhead Caused by TLS,” 2014.

[42] Android developers, “Wi-Fi Aware overview: Ranging peers and location-aware discovery.” https://developer.android.com/guide/topics/connectivity/wifi-aware#ranging_peers_and_location-aware_discovery. Accessed: 2021-06-02.

[43] M. Woolley, “How Bluetooth Mesh Puts the ‘Large’ in Large-Scale Wireless Device Networks.” https://www.bluetooth.com/blog/mesh-in-large-scale-networks/, 6 2018.

[44] Silicon Labs, “AN1200.1: Bluetooth® Mesh 2.x for iOS and Android ADK.” https://www.silabs.com/documents/public/application-notes/an1200-1-bluetooth-mesh-2x-for-android-and-ios-adk.pdf. Accessed: 2021-04-03.



NTNU

Norwegian University of
Science and Technology