Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Security Challenges and Pitfalls in Wireless Connection Establishment and Communication

DISSERTATION

with the aim of achieving a doctoral degree at the
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
University of Hamburg

submitted by
Johanna Ansohn McDougall

October 2024

Gutachter:

Prof. Dr. Hannes Federrath
Prof. Dr. Janick Edinger

Tag der Disputation:

10. April 2025

Abstract

Wireless communication can be divided into three stages: Device discovery, connection establishment and data transfer. In the first stage, device discovery, the communicating parties have to be made aware of each other's presence. In the subsequent initialisation of the connection, the parties exchange communication parameters, for instance to provide data encryption or authentication. Upon finalisation of this stage, data can be exchanged.

From a security research perspective, each of these stages exhibit their own specific pitfalls: In **device discovery**, maintaining the anonymity and privacy of the device bearer is crucial. This ensures that albeit devices are mobile and capable of communication, they are not misused for tracking or tracing of their users. This is explored by using probe requests as an exemplary technology: Probe requests are packets transmitted by Wi-Fi capable devices to identify known Wi-Fi networks within proximity. When probe requests were first implemented as a means of device discovery, they typically contained the Universally Administered Address (UAA) – the hardcoded Media Access Control (MAC) address – of the sender, and additionally often a whole list of known Service Set Identifiers (SSIDs). Both attributes allow for trivial device fingerprinting, and therefore both tracking and tracing of their users. While privacy preserving techniques subsequently introduced safeguarded their user's privacy more efficiently, probe requests still contain enough information to enable attackers to track their user's devices. This dissertation discusses four means of protecting user privacy in the phase of device discovery, the latter three of which were newly devised in the context of this dissertation: The first suggests to cease the use of active discovery, relying on improved passive discovery instead. The latter concern firstly, the reduction of probe requests content, in order to unify them in

their appearance and inhibit tracking. Secondly, the use of a generic MAC address for probing instead of inadequately implemented randomisation schemes to prevent information inference via the MAC address. And thirdly, a hash-based scheme for covert SSID transmission that would enable the privacy-friendly probing for hidden networks.

The second stage, **connection establishment**, is a particularly vulnerable stage from a security perspective, since key material is exchanged for the subsequent data transfer. If the keys are compromised, all subsequent communication can be decrypted by the attacker, and possibly even forged. This is explored using the example of public Wi-Fi networks and their susceptibility to passive and active attacks, and the use of a Virtual Private Network (VPN) to protect users in such insecure environments. With the help of a VPN, both metadata as well as the content of the communication can be protected from eavesdropping and modifications. But during connection establishment, particularly in public Wi-Fi networks, VPNs can exhibit undesired behaviour: One is the leakage of data during VPN tunnel establishment. The other is the possible inability to use a VPN in a public network containing a captive portal, resulting in a captive deadlock: A VPN application attempting to connect to its dedicated servers, but not programmed to allow for captive portal detection can leave their users caught in a captive portal, in which neither captive portal remediation is possible, nor connection establishment with the VPN servers. For this problem, this dissertation provides a solution that enables privacy-friendly and leakage-free use of VPNs in public Wi-Fi networks by proposing a scheme for selective VPN bypassing to mitigate captive deadlocks.

In the stage of **data transfer**, the authenticity and integrity of the transmitted messages is of particular importance. This is explored using the Automatic Dependant Surveillance-Broadcast (ADS-B) protocol as an example, a broadcasting system for aircraft, featuring neither encryption nor integrity protection or authentication. Therefore, ADS-B messages are neither protected from eavesdroppers nor from attackers injecting own messages into the system. To mitigate this, various approaches can be used, ranging from retrofitted cryptographic protection to radiometric fingerprinting or origin verification. This thesis explores the applicability of the distinct approaches and introduces an additional one: LoVe, a **Lo**cation **Ve**rification scheme using distributed public sensors.

Using the three protocols as case studies, this dissertation explores the challenges and pitfalls wireless protocols can present, and mitigations and countermeasures that can enhance the security and privacy of their users.

Zusammenfassung

Drahtlose Kommunikation kann in drei Phasen eingeteilt werden: Geräteerkennung, Verbindungsaufbau und Datenübertragung. In der ersten Phase müssen die kommunizierenden Parteien einander finden. Im anschließenden Verbindungsaufbau tauschen die Parteien Kommunikationsparameter aus, um beispielsweise Verschlüsselung, Authentifizierung oder Integritätsüberprüfung zu ermöglichen. Nach Abschluss dieser Phase können Daten übertragen werden.

Aus der Perspektive der Sicherheitsforschung weist jede dieser Phasen ihre eigenen Fallstricke auf: Während der **Geräteerkennung** ist die Wahrung der Anonymität und Privatsphäre entscheidend, um sicherzugehen, dass die mobilen Geräte nicht zum Tracking ihrer Nutzer:innen verwendet werden. Dies wird anhand von Probe Requests untersucht: Probe Requests sind Pakete, die WLAN-fähige Geräte übermitteln, um bekannte WLAN-Netzwerke in Reichweite zu identifizieren. Die erste Generation von Probe Requests propagierte oftmals die Universally Administered Address (UAA) – die hardgecodete Media Access Control (MAC)-Adresse des WLAN-Harewaremoduls des mobilen Gerätes – und oft auch eine Liste bekannter Service Set Identifier (SSID). Beide Attribute ermöglichen ein einfaches Fingerprinting von Geräten und somit auch das Tracking und Tracing ihrer Nutzer:innen. Obwohl später eingeführte Techniken die Privatsphäre weniger stark kompromittieren, enthalten Probe Requests nach wie vor ausreichend Informationen, um Angreifer:innen das Tracking zu ermöglichen. Diese Dissertation diskutiert vier Möglichkeiten zum Schutz der Privatsphäre der Nutzer:innen in der Phase der Geräteerkennung, von denen die letzten drei im Rahmen dieser Dissertation neu entwickelt wurden: Die erste betrifft die Nutzung von *Passive Discovery* anstelle von *Active Discovery*, dabei wird

auf die Übertragung von Probe Requests verzichtet und stattdessen Geräteerkennung ausgehend vom Access Point (AP) betrieben. Die anderen drei betreffen erstens die Reduzierung des Inhalts von Probe Requests, um sie zu vereinheitlichen und das Tracking zu erschweren. Zweitens wird die Verwendung einer generischen MAC-Adresse während der Geräteerkennung anstelle von ungenügend implementierten Randomisierungsschemata erforscht, um die Möglichkeit des Informationsgewinns über die MAC-Adresse zu reduzieren. Der dritte Ansatz führt ein hashbasiertes Schema für die verdeckte Übertragung von SSIDs ein. Dies ermöglicht die datenschutzfreundliche Erkennung versteckter Netzwerke.

Die zweite Phase, der **Verbindungsaufbau**, ist aus der Sicherheitsperspektive eine besonders vulnerable Phase, da in dieser Phase das Schlüsselmaterial für die anschließende Datenübertragung ausgetauscht wird. Im Fall der Schlüsselkompromittierung kann die gesamte nachfolgende Kommunikation von Angreifenden entschlüsselt und möglicherweise sogar gefälscht werden. Dies wird am Beispiel von öffentlichen WLAN-Netzen und deren Anfälligkeit für passive und aktive Angriffe und der Verwendung eines Virtual Private Network (VPN) zum Schutz der Nutzer:innen untersucht. Mit Hilfe eines VPNs können sowohl Metadaten als auch der Inhalt der Kommunikation vor dem Abhören und der Modifikation geschützt werden. Speziell in öffentlichen WLAN-Netzen kann während des Verbindungsaufbaus unerwünschtes Verhalten auftreten. Einerseits existiert die Möglichkeit von Datenlecks während des VPN-Tunnelaufbaus. Andererseits verfügen viele VPN-Applikationen nicht über Erkennungsmöglichkeiten von Captive Portals, welche häufig in öffentlichen Netzen anzutreffen sind. Das kann zu einem *Captive Deadlock* führen: Eine VPN-Anwendung, die versucht, eine Verbindung zu ihren dedizierten Servern herzustellen, aber nicht über Captive Portal-Erkennung verfügt, kann ihre Benutzer:innen im Captive Portal gefangen halten, in dem weder die Bedingungen des Captive Portal erfüllt werden, noch ein Verbindungsaufbau mit den VPN-Servern möglich ist. Diese Dissertation löst dieses Problem, indem sie eine datenschutzfreundliche und datenleckfreie Nutzung von VPNs in öffentlichen WLAN-Netzen vorschlägt, bei der selektiv nur die für die Captive Portal-Auflösung relevante Datenübertragung erlaubt wird.

In der dritten Phase, der **Datenübertragung**, sind die Integrität und Authentizität der übermittelten Nachrichten besonders relevant. Dies wird am Beispiel des Automatic Dependant Surveillance-Broadcast (ADS-B)-Protokolls untersucht, einem Nachrichtenübermittlungssystem für Flugzeuge, das weder Verschlüsselung noch Integritätsschutz oder Authentifizierung bietet. Die unverschlüsselten ADS-B-Nachrichten können deswegen mitgeschnitten werden, und der mangelnde Schutz der Integrität und Authentizität ermöglicht es Angreifer:innen zusätzlich, eigene Nachrichten in das System einzufügen. Um dies zu verhindern, können verschiedene Ansätze verwendet werden: Auf der einen Seite können kryptographische Schutzmaßnahmen

nachgerüstet werden, auf der anderen Seite Fingerabdrücke des Funksignals genommen oder der Signalursprung verifiziert werden. Diese Dissertation untersucht die Anwendbarkeit der verschiedenen existierenden Ansätze. Zusätzlich führt sie einen weiteren Ansatz ein: LoVe (**Lo**cation **Ve**rification), ein Protokoll zur Verifikation des Signalursprungs, das mithilfe von *crowdsourced* Sensoren eine Datenbasis zum Abgleich neu eingepflegter Daten erstellt.

Anhand dieser drei Beispiele untersucht diese Dissertation typische Herausforderungen und Probleme drahtloser Kommunikation. Diese werden als Grundlage genommen, um Maßnahmen zur Verbesserung der Sicherheit und Privatsphäre zu präsentieren, zu implementieren und zu evaluieren.

Acknowledgements

This dissertation is the result of my employment at the working group for Security in Distributed Systems (Sicherheit in verteilten Systemen, SVS) at the University of Hamburg between 2019 and 2024. The journey has been very educational and rewarding, and also fun, emotional, exciting, lovely, and an enrichment to my life. I would like to take this opportunity to thank express my gratitude to a few people who have accompanied and supported me throughout this journey.

Firstly, I would like to thank Hannes Federrath for giving me the opportunity to write my doctor thesis under his supervision and supporting me in my endeavours. Thanks also to Mathias Fischer for his encouraging ways, and fascinating insights into all sorts of topics. Also, I'd like to extend my gratitude to Janik Edinger. I hope our paths will cross again!

A gigantic hug and a THANK YOU! to Joshua Stock, Matthias Marx, Christian Burkert, Anne Kunstmann, Monina Schwarz and Tom Petersen who made work so much nicer and always had time for tea (or coffee, or Mate, or Tschunk, or beer). It is wonderful to have you as friends. And it's also thanks to you that the difficult parts of this journey were manageable.

I had the chance to work on publications with fantastic people while writing this thesis, all of whom I'd like to express my gratitude to. First, there is Christian Burkert, who took me along on paper-journeys and gave me the confidence to do the same. Alessandro Brighente, who was always ready for the next paper and had brilliant ideas on how to improve it. Whoever said that spending time on the playground is not productive was proven wrong: A big thank you to Willi Großmann for the inspiring conversations, exciting insights and great results in between building

sandcastles. To Monina Schwarz for endless brainstorming sessions on entirely different topics. And a big thanks also to Anne Kunstmann, Joshua Stock, Niklas Zapatka and Daniel Demmler who were always willing to spend the extra hour. And to Ida Bruhns: Our paper never made it out there, but working on it with you was truly a pleasure. I hope our friendship lasts forever.

All the other colleagues at SVS and NET, whom I've met over the years include Stefan Bavendiek (whose WLAN is very trackable ;-)), Max Blochberger (yay martial arts!), Heiko Bornholdt, Cornelia Brülhart (with whom I could discuss tattoo ideas forever), Eleftherios Eleftheriadis, Doganalp Ergenç (of whose jokes I never tire), Jeetesh Gupta, Steffen Haas, Malte Hamann, Dominik Herrmann, Maya Herrscher, David Jost (I still can't believe we can not agree on a single song), Liliana Kistenmacher, Kevin Köster, Majd Latah, Jens Lindemann, Sadaf Momeni, Tobias Mueller (one day I'll make it to one of your legendary parties!), Henning Pridöhl, Kevin Röbert, Janik Noel Schug, August See (best travel buddy when you go to Edinburgh, would recommend, 5 of 5 stars), Nurefşan Sertbaş Bülbül, Sehrish Shafeeq, Bahareh Shojaie, Britta Skulima (who always has all of our backs), Marius Stübs (whose music I will remember forever, and whose straightforwardness is inspirational), Erik Sy (whose business ideas were the best in the world), Jens Wettlaufer, Pascal Wichmann, Florian Wilkens (whose impression of a pigeon I hope to never forget, and whose humour I also highly recommend), Tatjana Wingarz and Ephraim Zimmer. It was always a pleasure to meet all of you, whether online or in the hallway. I'm looking forward to seeing you again!

I am enormously grateful to my family – especially to my parents, who have always believed in me, who have been excited about every little bit of news and who have supported me in every way. To my children Ada and Roxanne, who turn my life upside down in the best way possible. And my love, my pillar, my rock, my pillow, my one and only, Ben. I am forever grateful for your never-ending support and your constant reminders of what really matters.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

ACAS Airborne Collision Avoidance System. 159, 168

ADS-B Automatic Dependant Surveillance-Broadcast. 4, 6, 27, 29, 30, 33, 34, 37, 38, 41, 54–58, 157–171, 173, 177–181, 185, 189–192

AP Access Point. 6, 28, 43, 44, 48, 59, 60, 63, 84, 85, 90–93, 95, 97, 98, 105, 109, 111, 112, 117, 118, 128, 188

COTS Commercial Off-The-Shelf. 29, 133, 162, 168

CPD Captive Portal Detection. 49, 141, 144

ECDSA Elliptic Curve Digital Signature Algorithm. 164

ESP Encapsulating Security Payload. 126

GNSS Global Navigation Satellite System. 56–58, 161, 168, 178–180

IE Information Element. 28, 35, 36, 44, 45, 63, 64, 82–84, 96, 102, 104, 105, 107–109, 112–120

IKEv1 Internet Key Exchange version 1. 52, 135

IKEv2 Internet Key Exchange version 2. 52, 126, 135

LAA Locally Administered Address. 42–44, 65, 66, 74, 90, 95, 101, 105, 106, 108, 114

MAC Message Authentication Code. 164

MAC Media Access Control. 3, 5, 6, 28, 30, 32, 35, 36, 41–47, 60–66, 72–75, 81–87, 90, 94–102, 106, 113, 115, 116, 118–120, 184, 186, 187, 189

ML Machine Learning. 166, 178, 181

MLAT Multilateration. 166, 180–182

NIC Network Interface Controller. 42, 47

OS Operating System. 45–47, 50, 53, 62, 65–69, 71, 72, 75, 88, 96, 118

OUI Organisationally Unique Identifier. 36, 42, 46, 47, 61, 62, 86, 94, 102, 119

OWE Opportunistic Wireless Encryption. 17, 128, 130–132, 154, 155, 184, 188

PMK Pairwise Master Key. 130, 131

PNL Preferred Network List. 28, 44, 66, 69, 72, 75–77, 80, 82, 83, 87–90, 111, 118

PSK Pre-Shared Key. 128, 131

PSR Primary Surveillance Radar. 29, 31, 160, 165, 167, 191, 192

PTK Pairwise Transient Key. 127, 128, 131, 132, 189, 191

RSNA Robust Security Network Association. 48, 60, 109, 127, 129–131

SAE Simultaneous Authentication of Equals. 131, 132

SDR Software Defined Radio. 58, 159, 168, 179

SSID Service Set Identifier. 3–6, 15, 28, 30, 32, 35, 43, 44, 46, 47, 59–61, 63, 64, 66, 67, 69–73, 75–88, 90, 92, 96, 98, 102–105, 108, 110–112, 116–120, 183, 184, 186, 187, 189, 191, 192, 195, 197

SSR Secondary Surveillance Radar. 29, 31, 54

SVM Support Vector Machine. 178

TDoA Time Difference of Arrival. 166

TLS Transport Layer Security. 52, 53, 123, 127, 144

UAA Universally Administered Address. 3, 5, 42, 43, 45, 74, 84, 90, 95, 101, 114, 191

VPN Virtual Private Network. 4, 6, 15, 17, 19, 27–31, 33, 34, 36–38, 41, 49, 51–53, 121, 122, 124–127, 132–155, 184, 188–190, 192

WEP Wired Equivalent Privacy. 48, 123, 126, 127, 131, 184, 191

WPA Wi-Fi Protected Access. 48, 85, 123, 126–132, 139, 140, 154, 155, 184, 187–189, 191

1

Introduction

Ever since wireless communication technologies have emerged, more and more additional protocols enabled connections for mobile devices. The protocols are manifold and all serve different purposes, and the applications and necessities per protocol and device vary. Wireless communication can be grouped into the following key phases:

- Device Discovery
- Connection Establishment
- Data Transfer
- Connection Termination

A secure protocol design for each of these phases bears different challenges and pitfalls, with the exception of connection termination, which is considerably less security relevant and will therefore be omitted in this dissertation. Device discovery, connection establishment and data transfer, on the other hand, are particularly interesting with respect to their security and privacy guarantees. The focus of this dissertation is therefore a technical examination of each phase, illustrated using specific use cases: Device discovery in Wi-Fi networks, connection establishment during the use of a Virtual Private Network (VPN) and data transfer in the Automatic Dependant Surveillance-Broadcast (ADS-B) protocol.

Device discovery is the process of two wirelessly communicating parties identifying each other. In mobile communication, this entails announcements from either of

the parties. If the announcements are made via a stationary party, in the case of Wi-Fi communication an Access Point (AP), only the existence of the AP is revealed, which typically does not entail significant privacy implications. However, if the announcing party is the mobile device, its movements – and consequently its user's movements – might be disclosed due to the announcements. In Wi-Fi communication, the two possible sending parties define whether the protocol is denounced active or passive: Passive discovery is the act of APs transmitting *beacons*, which serve to announce a network's presence and Service Set Identifier (SSID). Mobile devices capturing these beacons to detect nearby known networks compare the announced SSIDs with their list of known networks, the Preferred Network List (PNL). In case they identify a matching SSID, they can initiate connection establishment. Active discovery, on the other hand, entails the mobile device transmitting *probe requests* to discover known networks. APs receiving a probe request respond with a *probe response*, containing their SSID. The first generations of mobile devices performing active discovery transmitted probe requests using their hardware Media Access Control (MAC) address, an unchanging identifier permanently tied to their wireless network interface card. Additionally, they often contained a list of known networks, with each successive probe request transmitted querying for a different SSID. The transmission of SSIDs is not only a poor policy in terms of data protection, but also serves as a fingerprint of the device transmitting it. In conjunction with the hardware MAC address, these first probe requests contained ideal unchanging identifiers that facilitated device tracking and tracing [PS07; Fre15; Van+16]. Once manufacturers became aware of the privacy implications of active discovery via probe requests, they started omitting the SSID unless querying for a hidden network, and employed MAC address randomisation to remove the unchanging identifier [Aar14; And23]. But nevertheless, tracking devices via the information contained within the probe request, namely the field called *Information Element* (IE), remains possible [Rob+17; Zha+19; DPČ19; Gu+20; Ura+20; TC21; PA22; HTC23]. Chapter 3 therefore explores ways to improve device discovery in Wi-Fi networks, introducing and discussing methods to increase user privacy while maintaining functionality.

The second stage of communication, **connection establishment**, exhibits other challenges: Here, encryption keys are exchanged and the communicating parties authenticated. If an attacker succeeds in compromising this phase, all security guarantees regarding the subsequent data exchange are void. The challenges and pitfalls arising during connection establishment are illustrated using the example of VPNs in public Wi-Fi networks: Public Wi-Fi networks are a convenient means of obtaining internet connectivity on the go. However, it is possible for the Wi-Fi operators to monitor all traffic. Depending on the protection scheme used, the monitored traffic can even be unencrypted, or the key exchange or encryption scheme one such that it is possible to retrieve the key and decrypt all subsequent communication [TB09; VP17; Dar]. To

introduce their own layer of security, users can employ a VPN client, which encrypts all traffic and transmits it via VPN servers. This way, both the metadata as well as the content of the traffic is protected from eavesdropping. But the use of a VPN can also lead to a false sense of security. Particularly during connection establishment, traffic leakage is likely to occur [Kha+18; Ikr+16]. Additionally, when used in conjunction with public Wi-Fi networks, VPNs can interfere with the detection of captive portals, as will be shown in Chapter 4. In such a case, the VPN can prohibit access to a captive portal, which is necessary to gain internet access. Simultaneously, the captive portal continues to block internet access, preventing the VPN from establishing a connection. In such a deadlock situation, applications on the mobile device might already attempt to transmit data, expecting the established Wi-Fi connection to work, causing traffic leakage outside the not-yet-established VPN tunnel. This is unacceptable since the use case of privacy-friendly surfing relies on *all* traffic being tunnelled via the VPN servers. As this dissertation will show, leakage occurring during the use of VPNs often has its root in the phase of connection establishment. Designing VPNs in a way that they fulfil their full functionality and are able to provide all privacy guarantees is therefore fundamental.

Subsequent to connection establishment, **data** can be **transferred**. As was previously stated, data leakage during connection establishment has implications on the whole subsequent connection. This dissertation regards a special case of data transfer, which does not rely on successful connection establishment or prior key exchange: Unencrypted and unsigned broadcast communication, using the example of the ADS-B protocol. ADS-B is a messaging system in which aircraft transmit their positional data and identifier, among other things, every 0.5 seconds [SLM15a]. There are two positioning systems for aircraft surveillance: The Primary Surveillance Radar (PSR) and the Secondary Surveillance Radar (SSR). The main positioning system is the PSR, which uses conventional radar technology. ADS-B is part of the SSR, and therefore used as a secondary source of information for real-time air monitoring. However, its air-to-ground range, covering up to 370 km [Sca02], makes it the ideal protocol and often the only source of positioning data in remote, sparsely populated areas, over oceans, and in difficult terrain. The ADS-B protocol contains no means of content or integrity protection, making it possible to eavesdrop on the communication even with Commercial Off-The-Shelf (COTS) equipment. The lack of confidentiality allows for public data collection, enabling providers like the OpenSkyNetwork [Sch+14] to collect ADS-B data via distributed public sensors and provide it for personal or scientific research. The lack of integrity, on the other hand, allows attackers to inject their own messages into the system, to modify copies of messages transmitted by other aircraft, delete messages and attack the system via GPS attacks (cf. Section 5.1). Since this can have real-life implications, from disruptions in the surveillance system to potential collisions [CF12; Gre12; RK17], it is crucial to identify means of verifying

the authenticity of these messages. In Chapter 5, existing strategies to verify signal integrity and authenticity are discussed, and a new one – LoVe, Location Verification using distributed sensors – is introduced.

The subsequent chapters of this dissertation will explore the three main stages of wireless communication from **two perspectives** – that of the attacker attempting to glean information from each protocol, abusing existing vulnerabilities, and that of the defender striving to protect the content and user privacy.

The remainder of this chapter is structured as follows: In the subsequent Section 1.1, the problem statement is summarised. Section 1.2 presents the research questions that this dissertation strives to answer, and the methodology employed to answer them. Thereafter, the contributions of this dissertation are summarised in Section 1.3. Lastly, the structure of the dissertation is explained in Section 1.4.

1.1 Problem Statement

When considering wireless communication protocols, one would assume that in their initial design, the question of how to implement the protocol in the most privacy-preserving and secure way would be addressed at a very early stage and treated with utmost importance. In reality, however, **security and privacy are often not considered** in the initial protocol design. Probe requests and the ADS-B protocol are the perfect examples for this: In their initial design, probe requests were transmitted containing the real MAC address of the sender device and often also a list of known SSIDs. The privacy implications of this protocol design were such that mobile devices could be **trivially tracked** via their probe requests. Mobile devices have a comparatively short lifecycle, and changes to the protocol would intuitively have a rapid impact in the real world. On the other hand, the installation of the transponders transmitting ADS-B signals is mandatory for a significant number of aircraft, and modifications on such a mandated installation are significantly more difficult to execute. The ADS-B protocol can therefore not easily change over time. Instead, such a long-lived protocol has to be designed to be usable over a long period of time and **can not be easily modified** in case of problems. Therefore, the realisation that the ADS-B protocol does not contain any mechanisms for encryption or authentication did not result in protocol changes. Instead, it remains easy for attackers to inject their own messages into the system.

The example of the use of VPNs for encrypted data transfer showcases another problem: Even if the protocol design, and sometimes even the documentation, state that the security measures implemented in the application are sufficient to protect user data and privacy, flaws in the implementation can cause leakage. In such a case,

the **theoretical security guarantees provided by the use of a VPN are not met in practice**. If additionally used to safeguard against eavesdroppers in a public Wi-Fi network, leaking VPN implementations can have **significant privacy implications** for their users.

The questions this dissertation strives to answer are:

- Which are the pitfalls of each communication phase?
- How can security considerations be taken into account from the very beginning?
- And in case protocol modifications are impossible, how can security measures be retrofitted nevertheless?

The **use cases for each stage are carefully chosen to be representative and widespread**: Most mobile devices nowadays are Wi-Fi capable, making Wi-Fi networks and their secure use a relevant focus. Active discovery (cf. Section 2.1.4) is the **predominantly used mechanism for device discovery** in Wi-Fi networks, which underscores its importance. An examination of probe requests as a representative protocol for the stage of device discovery therefore **reflects the practical realities of network environments**.

The same applies to the use of VPNs in public Wi-Fi networks: As will be shown in Section 4.2, connection establishment in public Wi-Fi networks can be compromised in many ways, and VPNs are advertised to protect all user traffic, even in poorly protected networks. This is a strong promise, and failure to comply can have serious ramifications. Connection establishment in VPNs in public Wi-Fi networks is therefore a representative example of the importance of **hardened implementations that prevent data leaks and secure the communication process**.

ADS-B is a protocol used worldwide, mandated by European and US-American air regulation agencies, and current deployment around 97 % (cf. Section 2.4). However, it is used in an entirely different context: as part of the SSR, it transmits the location, altitude and direction the aircraft is heading to at regular intervals. While the PSR using conventional radar technology remains the primary source of location information near ground towers, ADS-B is often the only source of information in sparsely populated areas and in difficult terrain. It's **reliability is, therefore, crucial**. Despite its critical role, ADS-B was developed without any means of integrity protection or authentication. This makes it a **prime example of an insecure protocol in widespread use**, and to demonstrate how **security mechanisms can be retrofitted retroactively**.

In the following, the research questions specifying these topics in more detail are introduced.

1.2 Research Questions and Research Methodology

This dissertation addresses four guiding research questions which are particularly interesting in the context of the problems previously discussed. Questions 1 concerns the security challenges in the stage of device discovery, focusing on privacy-preserving active discovery. Question 2 addresses pitfalls arising in the stage of connection establishment. While the main focus of question 3 is retrofitting security mechanisms in the stage of data transfer, it is additionally extended to regard the two other communication phases as well. Question 4 attempts to emphasise the broad picture of all wireless connections and strives to investigate the effectiveness of knowledge transfer from previous protocol versions to current ones.

1.2.1 Research Question 1: Privacy-Friendly Device Discovery

How can device discovery be performed in a privacy-friendly manner? What modifications have to be employed in current protocols to improve the state of the art?

The first protocols to implement active discovery in mobile devices constructed a discovery mechanism that could not only be used to discover access points, but also to trivially track devices via unchanging identifiers such as the real hardware MAC address or known SSIDs. While latter protocol versions contain less unchanging identifiers, probe requests still contain plenty of information to facilitate tracking. Chapter 3 of this dissertation therefore attempts to answer the question of how to approach the topic of device discovery differently, and how a privacy-preserving way of performing active discovery can be implemented.

To answer this research question, a **systematic analysis** of existing threats and mitigations is performed. Both **scientific literature** as well as **standardisation documentation** are analysed and evaluated in detail. A **field study** provides an overview over the state of active discovery in a real-world scenario. Its **evaluation** shows a significant amount of sensitive data in the transmitted SSIDs. Both these findings, as well as a **comprehensive review** of existing attacks result in several **proposals for protocol modification and improvement** to provide more privacy friendly active discovery.

1.2.2 Research Question 2: Challenges of Connection Establishment

What pitfalls can endanger the phase of connection establishment? How can they be overcome while preserving the functionality?

Using public Wi-Fi networks is dangerous from a data security and privacy perspective, since attackers residing within an unprotected network can trivially monitor traffic. Chapter 4 first explores the attacks that are possible particularly during connection establishment to differently protected networks. To nevertheless use public Wi-Fi networks, but simultaneously preserve data security and privacy, users can introduce an additional layer of security by using a VPN. In this context, Chapter 4 explores the pitfalls this entails, particularly when used in conjunction with the use of public Wi-Fi networks. The chapter additionally provides mitigations for the safer use of VPNs in public Wi-Fi networks.

This research question is answered by first performing a **systematic review** on the threats to the use of public Wi-Fi networks. Subsequently, a **documentation analysis** on the status quo of native VPN APIs is performed. An **experimental analysis** then verifies or disproves the documented capabilities and provides an overview over the capabilities of commercial VPN clients. A **protocol proposal** then introduces an improved technique for captive deadlock mitigation and reduced leakage during VPN bootstrapping.

1.2.3 Research Question 3: Retrofitted Security

How can security measures be retrofitted into protocols, and secondly, if protocol changes are impossible, what other means of securing the communication and its authenticity can be made? What means can be used to protect unencrypted and unauthenticated data transfer from modifications? How can modifications be detected in a protocol lacking integrity protection?

On one hand, this thesis regards protocols that are in use on rather short-lived devices: Probe requests, for example, are continuously transmitted by mobile devices like smart phones. These typically have a support span of a maximum of six years, but are often already replaced after less than six years. Protocol changes on such short-lived devices tend to have a visible effect within a short period of time, particularly if the protocol features the use of non-standardised elements. Aircraft, on the other hand, contain transponders to transmit ADS-B messages, and, unless they fail to work, these are unlikely to be exchanged. The messages they transmit are highly standardised since they have to be evaluated automatically by receiving devices, e.g.

ground towers or ADS-B-In capable devices. While all main chapters focus on the improvements that can be made within protocols and implementations, Chapter 5 regards amendments that can increase security and signal trust without requiring protocol changes.

To answer this research question, a **literature review** of the techniques proposed to increase and retrofit security in the absence of security mechanisms as in the ADS-B protocol is performed. The approaches are **systematised and evaluated** with respect to their applicability and effectiveness. Subsequently, an additional approach is **proposed, implemented and evaluated**. Additionally, the security measures that can be used to retrofit guarantees into the ADS-B protocol are **compared** to those applicable in other contexts, like probe requests, the Wi-Fi standard in general and VPN client implementations.

1.2.4 Research Question 4: Knowledge Transfer

Are the "lessons learned" from older protocols transferred to newer versions of the protocol, and which elements are particularly critical?

The focus of this dissertation is to provide an overview of the security challenges present in select protocols and the solutions used to reduce or mitigate their impact. This question strives to answer whether and to what extent the knowledge concerning problems detected in prior protocol versions is transferred to newer versions, and which problems are most likely to receive imminent notice and provoke protocol changes. It is a question that recurs throughout the main chapters of this dissertation.

With respect to probe requests, this research question is answered using a **systematic review** of the privacy enhancing techniques employed within probe requests over the years. A **literature analysis** on existing attacks and an **analysis of the documentation** on the protection schemes of (public) Wi-Fi networks showcases the development of both over time. With respect to ADS-B, a **literature analysis** reveals a significant number of ideas for improved protection schemes, and that protocol modifications are unlikely to occur. Instead, defence mechanisms not requiring protocol changes are **analysed and evaluated**.

1.3 Contributions of this Dissertation

The contributions of this dissertation can be summarised as follows:

C1. Analysis of privacy implications of SSIDs in probe requests and proposition of hash-based SSID transmission to increase user privacy during the discovery of hidden networks. Scientific research on probe requests has so far focussed on means of tracking users via elements contained within the probe requests. This contribution, presented in Chapter 3, places a focus on the privacy implications the transmitted SSIDs entail for their users: In a field study presented in [McD+22], probe requests transmitted by mobile devices were recorded over a period of three hours. The ensuing analysis revealed that the SSIDs collected thereby contained a wealth of personal data, among them first and last names of the device users and passwords. Additionally, various spelling variations of the same string could be observed, increasing the individual fingerprints. To propose a real-world solution, a hash-based approach for covert SSID transmission was devised, in which the hash of the SSID, with a salt consisting of the current randomised MAC address and the sequence number of the packet, would be transmitted instead of the plaintext SSID. Upon receiving such a covertly transmitted SSID, routers configured to provide a hidden network could then hash their own SSID with the MAC address and sequence number of the received packet, and compare it to the received hash. The feasibility of this scheme was verified in terms of computational and bandwidth overhead. As many of the SSID transmissions are likely to be caused by misconfiguration, this contribution additionally proposes user interface modifications. These enhance device safety by safeguarding SSID and hidden network entries, and introduce user controls to actively prevent tracking via probe requests.

C2. Proposition of content reduced, generic probe requests. The general problem of active discovery is its susceptibility to tracking: The *Information Element (IE)* of probe requests often contains a unique combination of values which enables device fingerprinting, and thereby user tracking and tracing. While several scientific publications had so far suggested to reduce the IE content, the implications of such a reduction on functionality, privacy and security had never been analysed in depth. Such an analysis was therefore published in [McD+24b], which is presented in Chapter 3: First, the minimum number of fields required to maintain functionality was determined. Subsequently, it could be shown that a reducing the content to the bare minimum has no adverse impact on functionality, as the security parameters required for connection establishment are actually transmitted in the subsequent probe responses and association requests. Additionally, the pre-transmission of parameters via probe request was shown to have no positive effect on the time required for connection establishment, either. A subsequent analysis on the privacy impact discovered that 82.55 % of the devices emitting probe request would

share the same anonymity group, and would therefore be indistinguishable from one another, if the content was reduced to the bare minimum. Lastly, it could be shown that a content reduction to the bare minimum would render all previously published attacks using IE fingerprinting infeasible.

C3. Proposition of the use of a generic MAC address. The absence of a standardised scheme for MAC address randomisation allows manufacturers to implement it without guidelines to reduce information leakage. Therefore, multiple schemes emerged, among them 24-bit randomisation (cf. Section 2.1.3), which leaks information on the manufacturer of the device. To propose an alternative to the current standard of using a randomised MAC address for probe requests, [McD+24a] explores the use of one generic MAC address over all devices, which is herein presented in Chapter 3. The use of just one generic MAC address over all devices would increase the anonymity set the probe requests are contained within, and thereby allow single devices to be less distinguishable, and bursts of probe requests considerably more difficult to pinpoint to individual devices. Additionally, it would impede information inference and tracking via the Organisationally Unique Identifier (OUI) of the MAC address when employing 24-bit randomisation, and would inhibit tracking devices using other insufficient MAC address randomisation implementations. The idea was implemented as a proof-of-concept and analysed both in terms of scalability and in terms of comparability with MAC address randomisation. The tests showed that its performance is comparable to MAC address randomisation, and that it scales well in tests with up to five real-world devices running the scheme.

C4. Analysis of leakage during VPN establishment in public Wi-Fi networks. When using public Wi-Fi networks, security-conscious users employ a VPN to introduce an additional layer of security, and protect both their meta data as well as their content from being monitored from within the network. In [Bur+21], which is presented in Chapter 4, data leakage occurring during the connection establishment of a VPN client with its VPN servers is examined. An analysis of various clients, both native and commercial, on different platforms showed that all VPN providers exhibited some form of data leakage during the VPN tunnel establishment. The publication additionally explored the capability of VPN clients to establish a connection in networks employing a captive portal, which is commonly used in public Wi-Fi networks to enforce user consent with the terms of service. Most VPN applications tested in this experiment exhibited insufficient captive portal detection, resulting in a deadlock, where neither the captive portal could be remediated, nor the VPN connection established: Out of 25 VPN applications tested in a captive network, only 9 successfully remediated the captive portal and established a VPN connection.

To additionally provide guidelines for successful and leakage-free VPN connection establishment in captive networks, a selective VPN bypass was proposed: Here, all data transfer during VPN connection establishment is reduced to the bare minimum required for VPN establishment and captive portal remediation. All traffic exceeding this bare minimum is blocked, and failure to connect at any stage leads to a complete traffic block until the issue is manually resolved by the user. Using this selective VPN bypass, both the traffic leakage typically observed after unsuccessful VPN establishment, as well as the occurrence of a deadlock situation can be circumvented.

C5. Proposition of a location verification scheme of ADS-B signals using distributed public sensors. ADS-B is a data transmission protocol that allows aircraft to inform both ground towers as well as other aircraft of their velocity, altitude, destination and current location. The protocol was, however, designed without mechanisms to ensure signal integrity, authenticity or encryption. Previous scientific attempts to verify signal integrity concentrated on cryptographic protection mechanisms, signal fingerprinting or signal origin verification. LoVe, published in [McD+23] and presented in this thesis in Chapter 5, extends the latter approach by exploring the possibility to verify the accuracy of a transmitted location using previously recorded signals. Using a geospacial hexagonal indexing system, a mask of legitimate sensor transmissions was constructed. With this mask, subsequently received signals could be verified with respect to their proclaimed location data. The approach can make use of any distributed ADS-B sensor network, and was tested on data stemming from Flightradar24 and the OpenSky network. Since it is neither dependent on sensor density nor duplicate signals from multiple receivers, both of which are a typical hindrance to previously published location verification approaches, it is a ubiquitously usable, lightweight location verification tool.

In summary, the stages of device discovery, connection establishment and data transfer are explored with respect to the challenges they present. The distinct stages are analysed using different protocols as case studies to illustrate existing flaws. The previously suggested approaches for mitigation are systematised and new solutions proposed.

1.4 Structure of this Dissertation

This dissertation is structured as follows, as can also be observed in Fig. 1.1: Chapter 2 provides a background on Wi-Fi networks and their inherent capabilities and

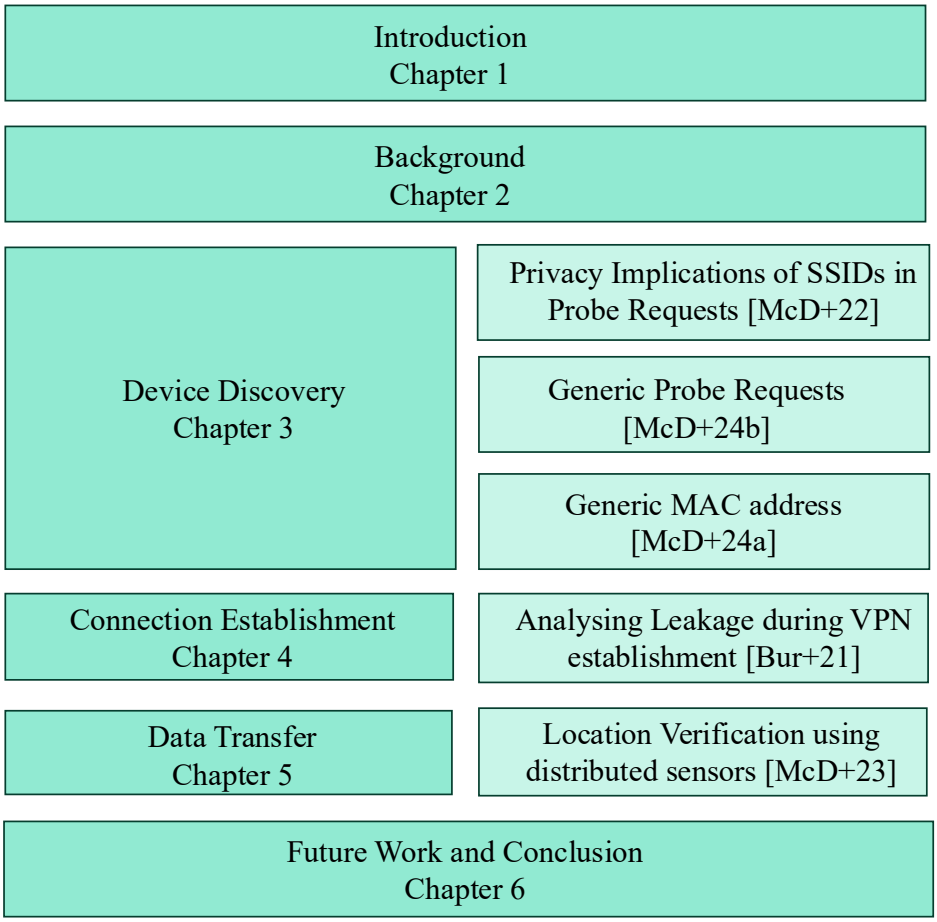


Figure 1.1: An overview over the structure of this dissertation. The light green boxes indicate the relevant publications per chapter.

requirements, VPNs, Captive Portals and the ADS-B protocol. Subsequently, Chapter 3 showcases challenges and mitigations surrounding the use of probe requests for active discovery in Wi-Fi networks. This chapter contains the results of contributions **C1, C2** and **C3**, and answers **research questions 1 and 4**. Chapter 4 then illustrates the pitfalls arising during the connection establishment of VPN applications, particularly those arising during the use of public Wi-Fi networks. This chapter is based on the results published in contribution **C4**, which addresses **research questions 2 and 4**. Afterwards, Chapter 5 uses the example of the ADS-B protocol to illustrate security challenges when attempting to verify the authenticity of unencrypted and unsigned broadcast messages. This chapter presents and enhances the results of contribution **C5** and addresses **research questions 3 and 4**. Afterwards, Chapter 6

first discusses the results, presents an outlook on future work and concludes this dissertation.

2

Background

The main technologies considered in this thesis concern Wi-Fi networks and the Automatic Dependant Surveillance-Broadcast (ADS-B) protocol. The following sections provide a background on both, with Wi-Fi introduced in Section 2.1, where an overview is given of Media Access Control (MAC) addresses and MAC address randomisation, as well as network discovery and connection establishment in Wi-Fi networks. Section 2.2 then introduces the functionality of captive portals and Section 2.3 provides a background on Virtual Private Network (VPN)s and their use. Finally, the ADS-B protocol and its inherent privacy and security features are introduced in Section 2.4.

2.1 Wi-Fi

This section presents the background required for Chapters 3 and 4. First, Section 2.1.1 introduces specifics on MAC addresses. Active and passive methods of network discovery are introduced in Section 2.1.2. Subsequently, MAC address randomisation is covered in Section 2.1.3. Finally, Section 2.1.4 provides a background on connection establishment in Wi-Fi networks.

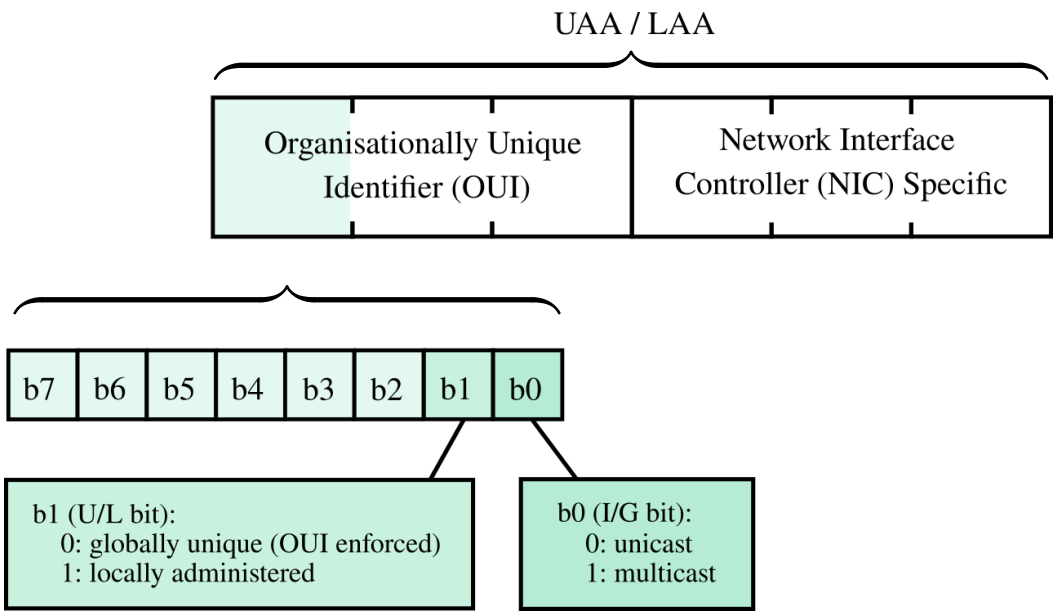
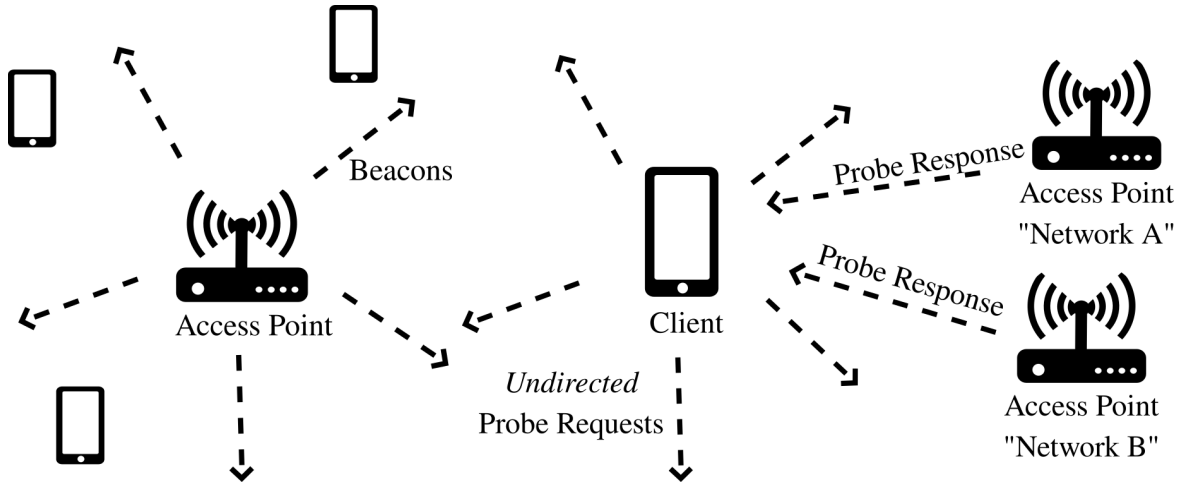


Figure 2.1: MAC Address format as specified in IEEE 802.11. Adapted from [ZBA24].

2.1.1 MAC Addresses

A MAC address, as outlined in the IEEE 802.11 specification [IEE20], is a 48-bit long network interface card identifier, required to differ between devices in wired or wireless networks. Its primary function is to identify the senders and receivers of frames transmitted on the data link layer. When two or more nodes within a network share the same MAC address, it results in a MAC address collision. This collision leads to an inability to distinguish affected nodes on the data link layer. A frame intended for a specific destination may then be received by any node with the colliding address.

The use of a Universally Administered Address (UAA) serves to prevent MAC address collisions: Each device is assigned a globally unique and fixed UAA by its manufacturer, the structure of which is also visualised in Fig. 2.1. The first three bytes of the MAC address contain the Organisationally Unique Identifier (OUI), identifying the manufacturer. The remaining three bytes, the Network Interface Controller (NIC) specific identifier, is assigned by the device manufacturer. Whether a MAC address is a globally unique UAA or a Locally Administered Address (LAA), is distinguished by the U/L bit, the second-least significant bit of the most significant byte. The least significant bit of the most significant byte of a MAC address contains the I/G bit. This



(a) In passive discovery, access points transmit beacons. Mobile devices receiving a beacon can initiate association.

(b) In active discovery, mobile devices transmit probe requests, awaiting probe responses. These probe requests are typically *undirected*, which means the Service Set Identifier (SSID) field is empty.

Figure 2.2: A comparison of passive discovery and active discovery using undirected probe requests.

bit differentiates between Individual (or unicast) and Group messages (multicast) [IEE14].

The use of an LAA provides an alternative to the UAA in transmitted frames. However, when employing an LAA, manual enforcement of network-wide uniqueness becomes necessary to avoid MAC address collisions. A widely-used example of the use of an LAA is MAC address randomisation: Here, parts of the MAC address or the whole address are randomised to inhibit tracking via an unchanging MAC address. This is covered in more detail in Section 2.1.3.

2.1.2 Network Discovery

To establish a connection between an Access Point (AP) and a mobile device, two different methods can be used: passive and active network discovery, as sketched in Fig. 2.2. In passive discovery, shown in Fig. 2.2a, a router transmits beacons, advertising its SSID, MAC address and other elements, e.g. supported cipher suites. Beacons are transmitted every 102.4 ms [Goo+19], and upon recording a beacon, devices can respond with a Wi-Fi association frame. Relying on passive discovery is particularly privacy friendly, as only the AP is required to transmit information.

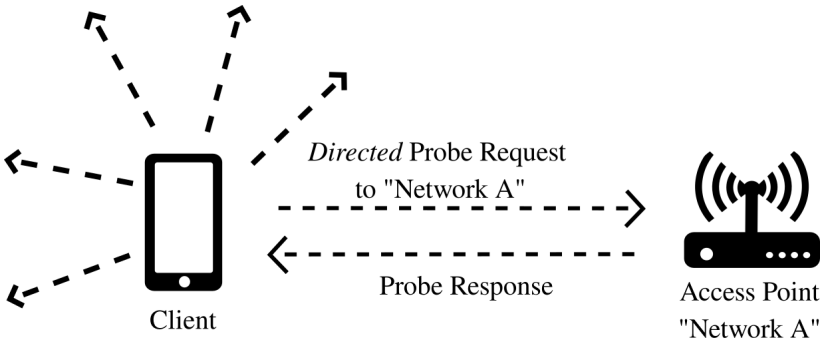


Figure 2.3: A sketch of active discovery using directed probe requests: the SSID of the known network is contained within the probe request, making it a *directed* request. Only the network with the SSID responds with a probe response.

Active discovery, on the other hand, requires devices to actively transmit requests for information on nearby APs, as can be observed in Fig. 2.2b. These requests are called probe requests. A probe request containing the SSID of a known network is called a *directed* probe request, and can also be observed in Fig. 2.3. The SSIDs of known networks are stored within the so-called Preferred Network List (PNL) of the device [WK18]. An undirected probe request instead contains an empty SSID field.

Probe Requests are typically sent in *bursts*: A burst of packets denotes a number of packets sent within a short period of time, and typically via multiple or all 14 channels that the 2.4 GHz spectrum comprises. Waltari et al. performed an analysis of probing behaviour within a burst, the results of which can be observed in Fig. 2.4: While the probing behaviour of a specific device exhibits certain characteristics, no generalisation on the duration of a burst or the order in which the channels are queried can be made. Depending on the device, bursts of probe requests are additionally sent via the 5 GHz spectrum. When MAC address randomisation is used (cf. Section 2.1.3), the address is typically randomised with each *burst* of packets and changed before the next burst. Many devices additionally randomise the sequence number with each burst. This behaviour can also be observed in Fig. 2.5. Directed probe requests typically transmit one probe request per SSID per channel.

Once a device recognises a known network from the SSID contained in its probe response, it often transmits another few probe requests, using its LAA. These are typically first undirected, and then directed.

Probe requests contain several *tags*, also called *fields*, the Information Element (IE) being the most identifying one, often used to single out and track devices (cf. Section 3.2.4). The IE contains information on the capabilities of the transmitting device. These capabilities include the Supported Rates and Extended Supported Rates, as

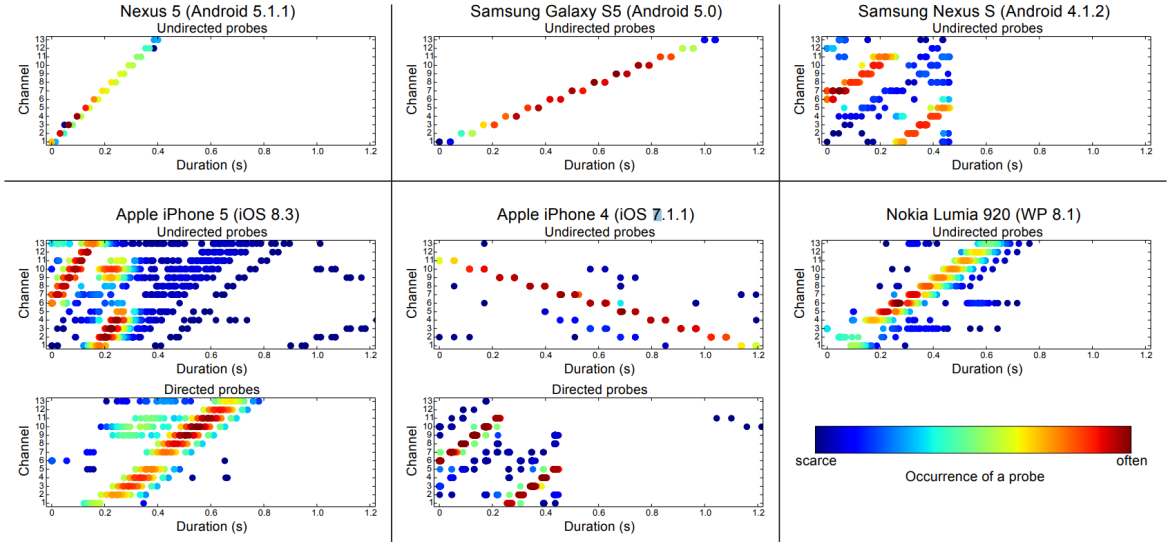
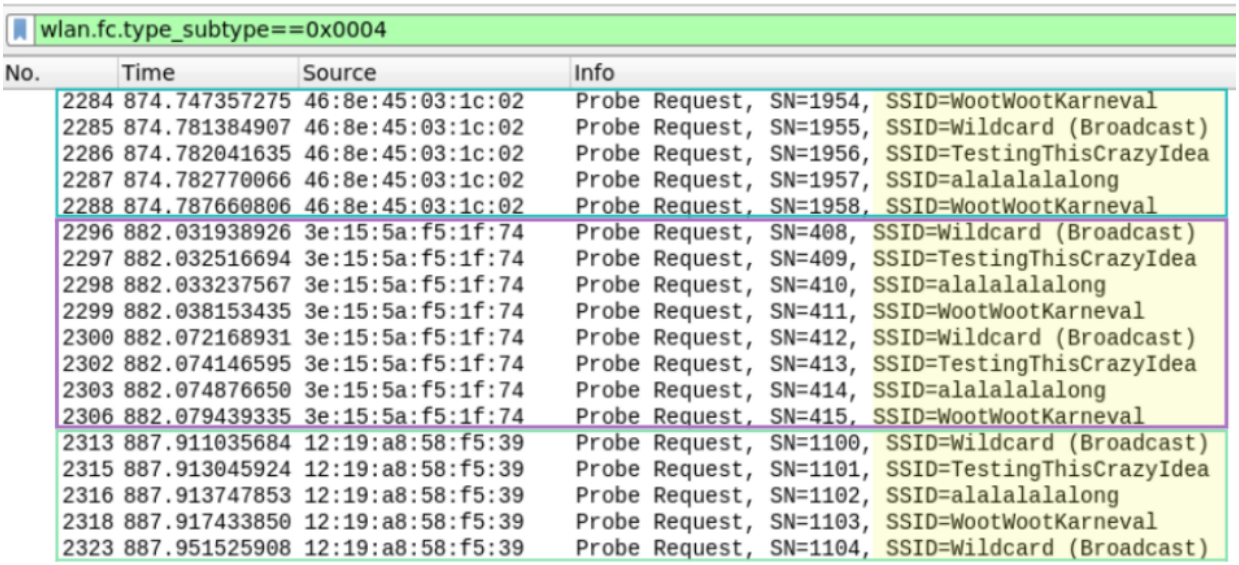


Figure 2.4: Probing behaviour of different devices as recorded by Waltari et al. Both the duration of a single burst as well as the channel selection is different per manufacturer and device type. Image source: [WK16].

well as the DS Parameter, which specifies the transmission channel. High Throughput (HT) and Very High Throughput (VHT) Capabilities declare the support for the IEEE 802.11n and IEEE 802.11ac standards, and Extended Capabilities that of additional features extending the HT and VHT Capabilities. Support for the IEEE 802.11ax standard is declared via the High Efficiency (HE) capabilities. Seamless connectivity within heterogeneous networks as defined in IEEE 802.11u is advertised via the Interworking tag. Probe requests also often contain a Vendor Specific tag, containing information inserted by the device's vendors. [IEE20]

An example of a directed probe request can be seen in Fig. 2.6. It contains an extensive IE, which is transmitted via the Wireless Management field. The observations described in Section 3.3.6.2 show that directed probe requests can contain considerably more information in their IE than undirected ones. Additionally, the newer the probing device's Operating System (OS) is, the less identifying information is contained in probe requests: Instead of transmitting their UAA, they employ MAC address randomisation and use a different MAC address and initial sequence number in every burst. [Van+16]

The newer the OS, the more privacy features are included. The older the OS, in turn, the weaker the privacy measures, e.g. devices running Android 8 and older automatically assume that manually added networks are hidden networks, whereas in newer devices, the hidden status has to be explicitly selected (cf. Table 3.1 and Section 3.2.3).



wlan.fc.type_subtype==0x0004

No.	Time	Source	Info
2284	874.747357275	46:8e:45:03:1c:02	Probe Request, SN=1954, SSID=WootWootKarneval
2285	874.781384907	46:8e:45:03:1c:02	Probe Request, SN=1955, SSID=Wildcard (Broadcast)
2286	874.782041635	46:8e:45:03:1c:02	Probe Request, SN=1956, SSID=TestingThisCrazyIdea
2287	874.782770066	46:8e:45:03:1c:02	Probe Request, SN=1957, SSID=alalalalalong
2288	874.787660806	46:8e:45:03:1c:02	Probe Request, SN=1958, SSID=WootWootKarneval
2296	882.031938926	3e:15:5a:f5:1f:74	Probe Request, SN=408, SSID=Wildcard (Broadcast)
2297	882.032516694	3e:15:5a:f5:1f:74	Probe Request, SN=409, SSID=TestingThisCrazyIdea
2298	882.033237567	3e:15:5a:f5:1f:74	Probe Request, SN=410, SSID=alalalalalong
2299	882.038153435	3e:15:5a:f5:1f:74	Probe Request, SN=411, SSID=WootWootKarneval
2300	882.072168931	3e:15:5a:f5:1f:74	Probe Request, SN=412, SSID=Wildcard (Broadcast)
2302	882.074146595	3e:15:5a:f5:1f:74	Probe Request, SN=413, SSID=TestingThisCrazyIdea
2303	882.074876650	3e:15:5a:f5:1f:74	Probe Request, SN=414, SSID=alalalalalong
2306	882.079439335	3e:15:5a:f5:1f:74	Probe Request, SN=415, SSID=WootWootKarneval
2313	887.911035684	12:19:a8:58:f5:39	Probe Request, SN=1100, SSID=Wildcard (Broadcast)
2315	887.913045924	12:19:a8:58:f5:39	Probe Request, SN=1101, SSID=TestingThisCrazyIdea
2316	887.913747853	12:19:a8:58:f5:39	Probe Request, SN=1102, SSID=alalalalalong
2318	887.917433850	12:19:a8:58:f5:39	Probe Request, SN=1103, SSID=WootWootKarneval
2323	887.951525908	12:19:a8:58:f5:39	Probe Request, SN=1104, SSID=Wildcard (Broadcast)

Figure 2.5: A screenshot of a capture of probe requests originating from the same device and probing for three different networks. The first burst is marked in blue, spanning request number 2284 to 2288. The second burst is marked in purple, spanning packets 2296 to 2306, and the third in green concerning packets 2313 to 2323. The bursts can be distinguished by their timestamp, with all packets being sent within less than 0.1 seconds. MAC address randomisation and sequence number randomisation per burst can also be observed in the screenshot. As the device is running an Android 8 OS, the manually inserted SSIDs are assumed to be hidden networks (cf. Section 3.2.1), and it therefore probes for the networks continuously using directed probe requests.

2.1.3 MAC Address Randomisation

To reduce the attack vector introduced via unchanging MAC addresses, MAC address randomization was introduced and discussed in 2014 [ZBA24], followed by testing and publication [BZO15]. With respect to mobile device OSs, Apple first implemented it in iOS 8 in 2014 [Aar14], and Android followed in 2015 with Android 6.0 [And23]. The way randomisation should be implemented is not standardised. However, its use in different circumstances, e.g. newly generated randomised addresses during each device boot, distinct per-network generated MAC addresses and others are summarised in a Request for Comments (RFC) draft [ZBA24].

The most common strategies are 46-bit randomisation, 40-bit randomisation and 24-bit randomisation. In **46-bit randomisation**, all bits except for the U/L-bit and the I/G-bit (cf. Section 2.1.1) are randomised, resulting in the devices using randomised OUIs and therefore identifiers which are not assigned to their actual manufacturer.

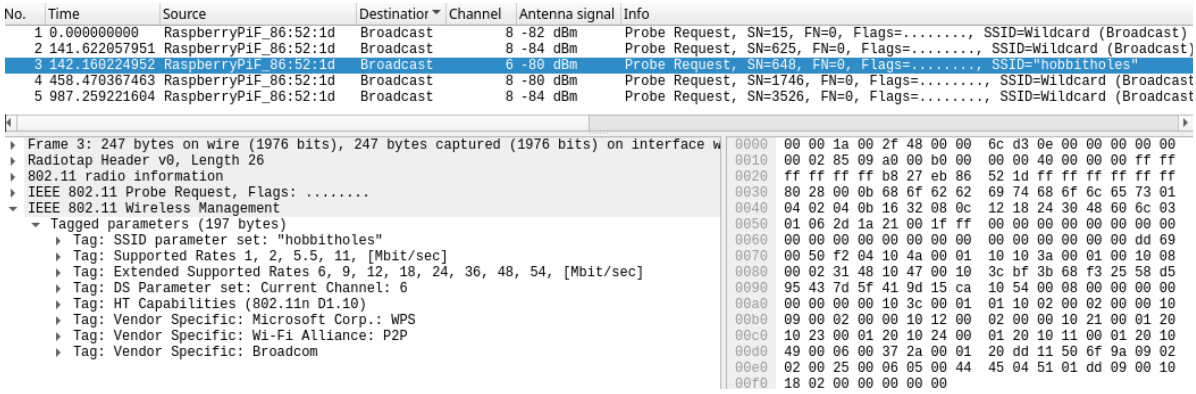


Figure 2.6: A directed probe request from a Raspberry Pi. The content of the IE can be seen in the Wireless Management tag. Apart from the SSID, it concerns Supported Rates, Extended Supported Rates, the DS Parameter Set, HT Capabilities and three vendor specific tags.

This behaviour is common in all Apple devices running modern OS, and in most devices running Android 10 and 11, and some of those running Android 9. [Fen+21]

In **40-bit randomisation**, the first byte of the MAC address is fixed to `02`, and the remaining MAC address randomised. This behaviour is common in many Samsung devices running Android 8 and 9 [Fen+21]. **24-bit randomisation** uses a fixed OUI (e.g. `92:68:C3` or `DA:A1:19`) and randomises only the NIC of the MAC address. This behaviour leaves devices particularly vulnerable to tracking since such a fixed prefix allows attackers to infer information on the device, e.g. the manufacturer or device type: While the OUI `92:68:C3` is typically used by Motorola Nexus 6 devices, `DA:A1:19` is commonly used by manufacturers other than Google producing Android devices. [Fen+21]

The term *46-bit randomisation* is used by Martin et al. [Fen+21] to describe randomisation of the whole address except for the U/L-bit and the I/G-bit. While they describe the behaviour of devices using 40-bit and 24-bit randomisation, they do not introduce a name for this behaviour. Both terms are therefore introduced by this thesis.

2.1.4 Connection Establishment in Wi-Fi Networks

The IEEE 802.11 standard defines Wireless Local Area Networks (WLANs) as wireless networks identified by an SSID and detectable via network discovery as explained in Section 2.1.2 [IEE20]. The term *Wi-Fi* was initially introduced by the branding firm InterBrand, which was hired by the founding companies of what later

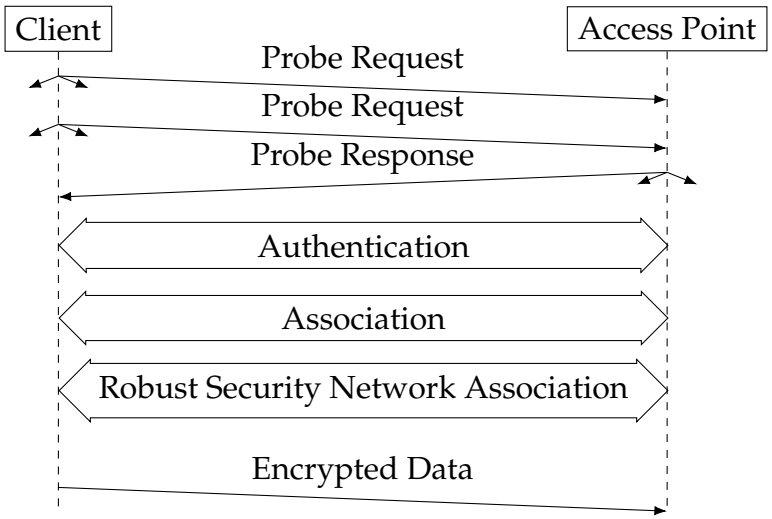


Figure 2.7: IEEE 802.11 connection establishment. The authentication and association phases are part of the 802.11 authentication and association, required for backwards compatibility with Wired Equivalent Privacy (WEP). The key material required for encryption using state of the art encryption schemes is exchanged in Robust Security Network Association (RSNA). Adapted from [McD+24a].

became the Wi-Fi Alliance. InterBrand was tasked to "develop a consumer-friendly name", which could be used to refer to the IEEE 802.11b High Rate Wireless Local Area Network standard [Enn]. Nowadays, the Wi-Fi Alliance still certifies devices with respect to their Wi-Fi interoperability, and the term Wi-Fi is typically used interchangeably with the term *WLAN*, but while WLAN describes the standard the devices adhere to, anglophone publications typically use Wi-Fi when referring to WLAN networks.

The association between a client and an AP, which enables the device to access the Wi-Fi network, encompasses the steps also shown in Fig. 2.7: Network discovery, either via active scanning using probe requests and probe responses as shown in the figure, or via passive discovery using beacons, allows Wi-Fi capable devices to discover nearby known networks. The subsequent IEEE 802.11 authentication and association allow backwards compatibility with the obsolete WEP standard and enable devices to transfer frames on higher layers. Subsequently, Wi-Fi Protected Access (WPA), WPA2 and WPA3 networks negotiate and exchange security parameters in the Robust Security Network Association (RSNA). Successful completion of RSNA enables devices to then exchange encrypted data frames. [IEE20]

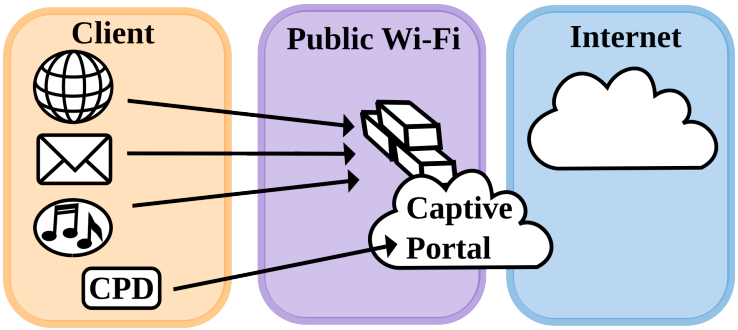


Figure 2.8: A client attempting to connect to the internet via a public Wi-Fi network, blocked by a captive portal. The Captive Portal Detection (CPD) of the respective platform attempts to connect the user to the captive portal, so captive portal remediation can commence.

2.2 Captive Portals

A captive portal is a means of intercepting a user's Wi-Fi connection to enforce credential submission or user agreement with policies inherent to the Wi-Fi provider [KK20]. As can be seen in Fig. 2.8, a captive portal typically blocks all internet access except for that necessary for the provisioning of the captive portal. Once the terms of the captive portal are fulfilled, the network block is lifted. This process is called *remediation* [Bur+21]. A network containing a captive portal is in the following referred to as a *Captive Network (CN)* [Bur+21].

In Chapter 4 of this thesis, the use of VPNs (cf. Section 2.3) in networks utilising a captive portal is discussed. Empirical evidence suggests that when using a captive network, captive portal detection is not always reliable and can constitute a problem on various platforms. The following sections therefore provide a background on both the Captive Portal Detection (CPD) implemented in Wi-Fi capable devices (cf. Section 2.2.1), as well as the implementation of captive portals within captive networks (cf. Section 2.2.2) to explain the implementation challenges many VPN providers experience according to the results presented in Chapter 4.

2.2.1 Captive Portal Detection (CPD)

There are multiple ways of announcing the presence of a captive portal within a network: On one hand, the two options of using the Dynamic Host Configuration Protocol (DHCP) or a router advertisement (RA) can be used. Both were proposed as a standard in RFC 8910 [KK20]. In both cases, the URI of the captive portal is transmitted within DHCP and RA options, advertising their existence to clients.

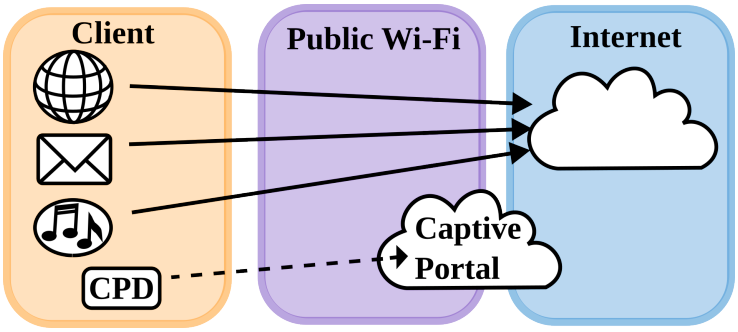


Figure 2.9: A client connected to the internet via a public Wi-Fi network, with the blockage by a captive portal remediated. The connection between the Captive Portal Detection (CPD) and the captive portal is reinitiated in case the internet connection fails.

Another method, which is predominantly used by devices initially connecting to a network, is the transmission of a HTTP request to captive portal detection URLs, with the expectation of a standardised response, as defined in [PT20]. For Apple devices, an HTTP request to the URL http://captive.apple.com/hotspot-detect.html is expected to return the plain text response `Success`. For devices running Android or ChromeOS, an HTTP status code 204 is expected as a result of an HTTP request to the URL http://connectivitycheck.gstatic.com/generate_204. Similar URLs and expected responses exist for Windows devices and Linux devices running NetworkManager. If the response to either of these requests differs from the expected result, the connecting device assumes the network is a captive network and can trigger the login process with the captive portal. [Wip16; Wik24a]

2.2.2 Implementation

The announcement of a captive portal within a network can be done using various methods, including an ICMP redirect and DNS redirection. Additionally, it is possible to perform HTTP redirection, particularly in response to requests to the above mentioned captive portal detection URLs: Since the captive network providers have full control over the network, they can modify the responses to HTTP requests. Upon receiving a HTTP request to the captive portal detection URLs, the captive network performs a variation of a Man-in-the-Middle-attack in which it issues a HTTP 302 status code as a response, which is a redirect to another page, forwarding the user to the login form for the captive portal.

Upon successfully remediating the terms of the captive portal, a user can gain access to the network as shown in Fig. 2.9.

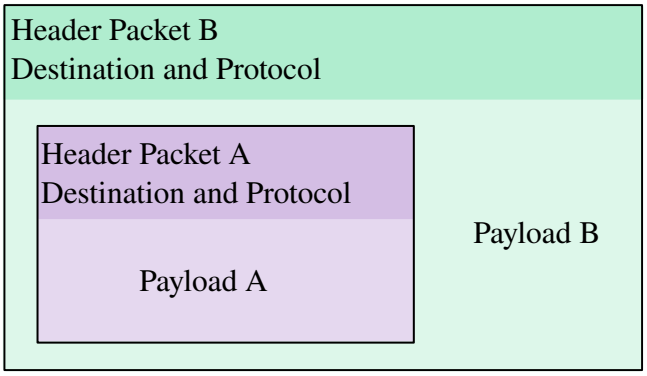


Figure 2.10: Encapsulation of a packet A within another packet B. Packet A is the original packet transmitted from a client device. The VPN client application running on the client device then encrypts the whole packet, and encapsulates it as the payload in packet B. Packet B is then transmitted to the VPN server, which decapsulates and decrypts it, and then forwards it to its original destination. This way, both meta data as well as the content are concealed from eavesdroppers. Adapted from [Lin23].

2.3 Virtual Private Networks (VPNs)

A widely used method for ensuring safer use of potentially unencrypted public Wi-Fi networks is the utilisation of a Virtual Private Network (VPN). While their original purpose was to enable remote access to services in a private network, VPNs are now mainly advertised as a means of ensuring data confidentiality and privacy in untrusted Wi-Fi networks. [Bur+21]

While VPNs also ensure sender authentication and message integrity, their relevant features in the context of this thesis are the privacy and confidentiality guarantees: The VPN client on the client device uses encryption to ensure that the content of the transmitted packages remains undecipherable to attackers. The encrypted packets are additionally encapsulated into new packets with the VPN server's IP address as the destination, as depicted in Fig. 2.10. Both measures in combination ensure that attackers sniffing traffic within a public Wi-Fi network are unable to infer information on either metadata or data content. The receiving VPN server processes the packets received from a client, decapsulates them and forwards them to the original destination address. Since it replaces the source address with its own address, it receives the responses and can then encrypt and re-encapsulate them to forward them to the client. The process is also depicted in Fig. 2.11.

Common VPN protocols include, but are not limited to the following:

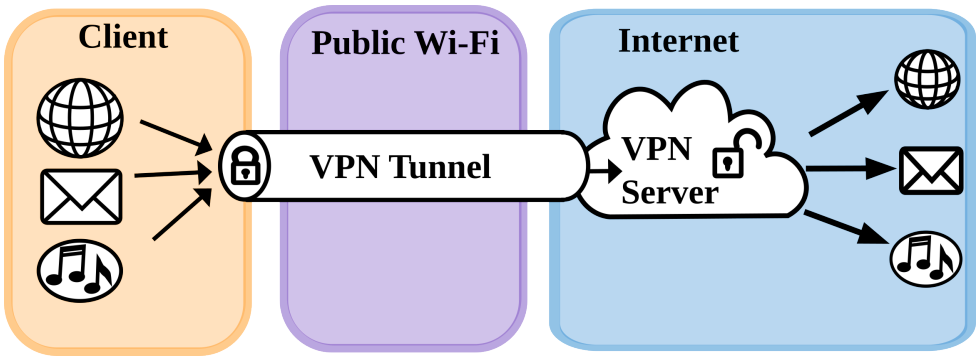


Figure 2.11: A VPN tunnel used within a public Wi-Fi network. The VPN client encrypts the packets received by the client and encapsulates them in packets with the VPN server as the destination to disguise the original recipient. The VPN server then decapsulates and forwards the original packets, additionally setting itself as the source address. It therefore receives the responses and can, again, encrypt and encapsulate them to forward them to the VPN client.

IPsec The Internet Protocol Security (IPsec) is a network protocol suite for packet authentication and encryption. Here, IP packets are encapsulated within IPsec packets, and tunnels established using the Internet Key Exchange (IKE) protocol, of which two different versions exist, Internet Key Exchange version 1 (IKEv1) and Internet Key Exchange version 2 (IKEv2). Even though IKEv1 *can* still be used for the implementation of VPNs, it was superseded by IKEv2 in 2005 and suffers from specification weaknesses, potentially leading to differing protocol implementations being unable to create a security association despite correctly appearing configuration [Kau05]. IKEv2 additionally introduces improvements over IKEv1 to mitigate cryptographic weaknesses, enabling state-of-the-art key exchange [Kau+14].

WireGuard WireGuard[1] is a lightweight, open-source VPN software. It encapsulates IP packets via UDP to evade potential transmission loss, e.g. induced by tunnelling TCP via TCP.

OpenVPN OpenVPN[2] is another open-source VPN tool. It utilises Transport Layer Security (TLS) for authentication and key negotiation [Ope]. In contrast to WireGuard, OpenVPN encapsulates packets via TCP, which can cause catastrophic loss in connectivity in case of insufficient bandwidth, resulting in a *TCP meltdown* [Hon+05].

PPTP The Point-to-Point Tunnelling Protocol (PPTP) is now obsolete, but was one of the first tunnelling protocols and widely used on Microsoft machines, since

1. https://www.wireguard.com/
2. https://github.com/OpenVPN/

it was natively included in the Windows OS. PPTP contains severe weaknesses [SMW99], but is still occasionally used for compatibility with legacy systems, and the possibility to set up a VPN connection using PPTP is still implemented in current Windows OS.

SSTP The Secure Socket Tunnelling Protocol (SSTP) is Microsoft's replacement for PPTP and enhances it by using SSL/TLS for transport encryption [Lin23]. It encapsulates packets via TCP, and therefore faces the same risks of a TCP meltdown in case of reduced bandwidth as OpenVPN does.

L2TP The Layer 2 Tunnelling Protocol (L2TP) provides an unencrypted layer 2 tunnel. To enhance it with encryption features, it is commonly combined with IPsec [Boo+01], the combination of which is referred to as L2TP/IPsec.

In the context of the use of VPNs, the term *VPN Bootstrapping* denotes the blocking of all traffic unnecessary for the establishment of the VPN tunnel. Once the VPN tunnel is successfully established, the blockage is lifted and traffic unrelated to VPN establishment, which is referred to as *third-party traffic* in Chapter 4, can be transmitted.

While it is possible to set up an own VPN using a dedicated server, many users instead resort to employing commercial VPN providers for ease of use. In this case, full trust is placed in the VPN provider, since they are able to access all packets tunnelled via their servers. To access VPN services, most OS platform providers implement native VPN clients. These typically support a number of VPN protocols, and connection settings have to be modified by the users. Third-party clients, on the other hand, are typically preconfigured to connect to specific services, requiring less user modifications.

While the main focus in this thesis is the use of VPNs to protect from eavesdropping on the traffic within the same public network, VPNs are also popular to circumvent censorship in authoritarian countries and to spoof location information to circumvent geographic restrictions, e.g. for streaming services that restrict access to content based on the country it is streamed to.

2.4 Automatic Dependant Surveillance-Broadcast (ADS-B)

The previous sections introduced different aspects of the Wi-Fi standard, and additionally captive portals and VPNs to provide a background for the stages of **device discovery** (Chapter 3) and **connection establishment** (Chapter 4). This section, on

the other hand, is required as background for Chapter 5, in which the ADS-B protocol is used as an example for unencrypted and unsigned **data transfer**, and different means of protecting the integrity and ensuring authenticity are explored.

The ADS-B system is a broadcast system in which aircraft transmit their identifier, location, velocity and altitude among other data regularly, every 0.5 seconds [SLM15a], via a specific transmitter on the plane. There are two competing ADS-B **link standards**, the Universal Access Transceiver (UAT) and the 1090 MHz Extended Squitter (1090ES). **UAT** was created specifically for the transmission of protocols such as ADS-B on the 978 MHz frequency [SLM13b]. Its has great uplink capacity and lower cost, and was therefore chosen as the protocol for use in general aviation [Sca02], meaning civil flights that are not part of scheduled avionic transportation [SLM13a]. UAT requires dedicated hardware, which the **1090ES** does not depend on: Here, the ADS-B signals can be received and sent via the *Mode S transponder* that is part of the Secondary Surveillance Radar (SSR). Commercial airlines are typically equipped with a Mode S transponder already, and the ADS-B functionality can be retroactively integrated into it [SLM15b].

To enhance air surveillance via ADS-B, the Federal Aviation Administration (FAA), a U.S. agency regulating civil aviation, **mandates** the use of the 1090ES [Bab10] for ADS-B transmissions for commercial aircraft flying above 5.5 km. For general flights flying below 5.5 km, the transmission of ADS-B signals via UAT is **recommended**, since the commercial airlines flying at higher altitude cause a significant congestion of the 1090 MHz spectrum already. Nevertheless, the use of 1090ES is also allowed for general aviation flying below 5.5 km. [Bab10].

Particularly **Europe and the USA**, with their air regulation agencies EUROCONTROL and FAA, strive to increase air safety by (1) mandating the use of ADS-B by all aircraft and (2) supporting the implementation of ground stations with ADS-B functionality. The requirement to install ADS-B transmitters **in Europe** applies to all aircraft built after the 8th of January 2015, and for all other aircraft by the 7th of December 2017. Several additional transition periods and exemptions later moved the deadline to 2020, then 2023 and lastly 2025. By 2021, around 97 % of the commercial airliners were observed to be ADS-B compliant. With respect to all aircraft, including general aviation aircraft and helicopters, the pervasiveness of the compliance and capability of transmitting ADS-B messages, is at 81.4 %. [Sun+21]

The other previously mentioned task both agencies fulfil with respect to ADS-B is the support of the **deployment of ground stations** to receive ADS-B signals and thereby increase aircraft visibility across the whole continent. The current operational coverage of ground stations in 2024 is around 64.3 % of the area that is operated by European Air Navigation Service Providers (ANSPs) [EUR24], as can also be seen in

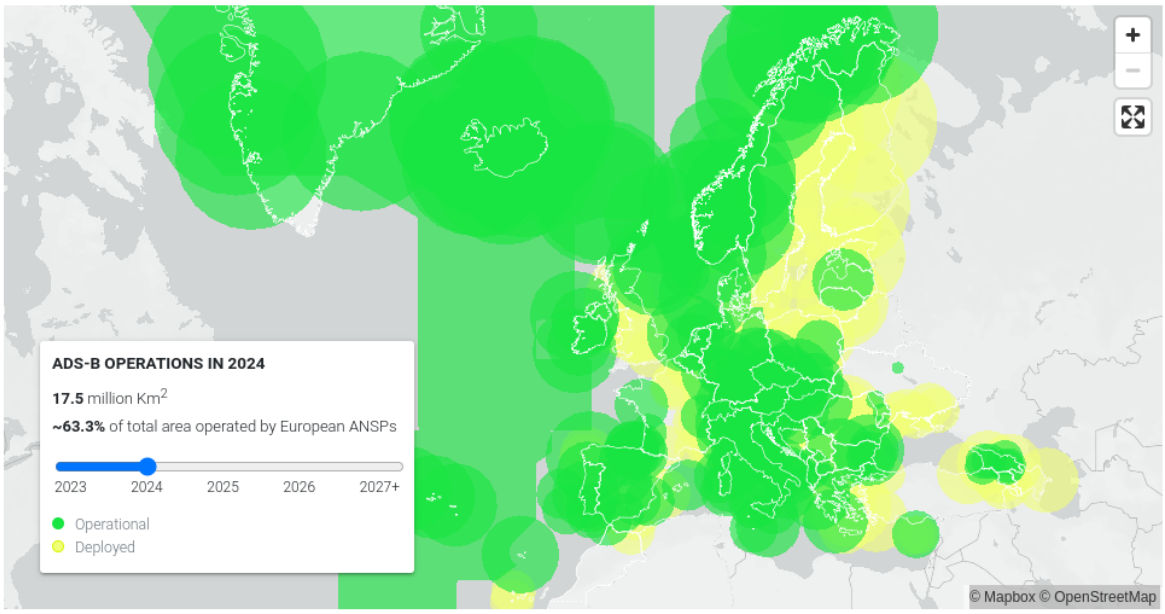


Figure 2.12: Distribution of deployed and operational ground stations for ADS-B signal reception by European Air Navigation Service Providers (ANSPs). Image source: [EUR24].

Fig. 2.12. The area of deployment is slowly, but steadily rising, with 65.6 % coverage expected by 2027.

ADS-B is, in contrast to other protocols used via the Mode S transponder, a broadcast technology that regularly transmits information **without prompting**. Other protocols used on the same frequency and via Mode S transponders are Elementary Surveillance (ES) and Enhanced Surveillance (EHS), both of which operate under an *interrogatory* environment: While ADS-B is regularly transmitted without prompting, ES and EHS are **transmitted on request by ground stations**.

2.4.1 ADS-B Packet Structure

ADS-B packets have a length of 112 bits for 1090ES, and 272 bits for UAT messages [RK17]. Since 1090ES is the protocol used for commercial aviation and also the one predominantly discussed in scientific research, the focus of the subsequent packet structure introduction are 1090ES ADS-B packets.

Before an ADS-B packet is sent, a **preamble** consisting of two synchronisation pulses is transmitted, as can be seen in Fig. 2.13. The preamble is transmitted over a duration of 8 µs, and the following ADS-B packet is then transferred using pulse position modulation (PPM), in which one bit can be transmitted every microsecond [SLM15b].

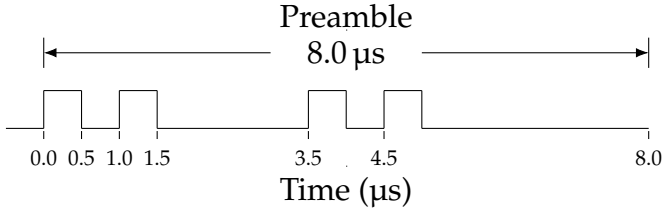


Figure 2.13: ADS-B Preamble. Adapted from: [SLM15b].

Downlink Format (DF)	Capability (CA)	Aircraft Addr. (AA)	ADS-B Data (ME)	Parity Check (PI)
5 bit	3 bit	24 bit	56 bit	24 bit

Table 2.1: Packet structure of 1090ES ADS-B packets.

ADS-B packets consist of the **five distinct fields** Downlink Format, Capability, Aircraft Address, ADS-B Data and the Parity Check, the length of each of which can be observed in Table 2.1. The **Downlink Format (DF)** field indicates the type of message being transmitted and helps receivers to determine how to interpret the message. Different DF values correspond to different message types. The **Capability** field specifies the equipment capabilities of the transmitting aircraft, such as its ability to use various surveillance technologies or features. The unique 24-bit **Aircraft Address** identifies the aircraft transmitting the ADS-B message. It is assigned by the aviation authority International Civil Aviation Organization (ICAO), and serves as a globally unique identifier for the aircraft. The **ADS-B Data** frame contains the specific information corresponding to the message type. This can include the airborne position, surface position, identification, the airborne velocity, the aircraft status and and aircraft operational status [Sun+21], as well as event driven information, e.g. concerning an emergency or priority information [Org14, p. 197]. The positional information are acquired via Global Navigation Satellite System (GNSS) satellites (cf. Section 2.4.3) and provided as latitudinal and longitudinal coordinates. The length of the data field is 56 bits. The subsequently transmitted 24 **parity bits** provide error-checking and redundancy. They are used to verify the integrity of the transmitted data [SLM13b] via a cyclic redundancy check (CRC) [SLM15b]. Additionally, up to 5 bit-errors can be corrected using the parity bits via a degree 24 fixed generator polynomial [SLM15b]. A message containing more than 5 bit-errors will be discarded [WSG20].

2.4.2 ADS-B Security and Privacy

ADS-B is a protocol build as an unauthenticated broadcast system with no encryption in place. A paper on the ADS-B link decision published by the Federal Aviation Administration (FAA) [Sca02] concerns only the frequency and link technologies taken into consideration during the design of ADS-B, and completely omits any security considerations. The focus seems to have been to construct a system that can support short range air-to-air communications of less than 40 nautical miles (74 km), as well as longer range air-to-ground surveillance up to 200 nautical miles (370 km), and possible attacks seem not to have been taken into consideration. The lack of security measures facilitates several attacks to disrupt the communication. This includes jamming, spoofing, bit-flipping and message injection; attacks on the protocol remain undetected as no mechanisms for attack detection exist [WSG20]. A comprehensive list of possible attacks can be found in Section 5.1.

While ADS-B messages are neither encrypted nor authenticated, the system contains two privacy features: One is *Limiting Aircraft Data Displayed* (LADD), by which the amount of aircraft data displayed in flight tracking services can be reduced. The other is *Privacy ICAO Aircraft* (PIA), through which a temporary identifier can be requested to increase the privacy of a specific aircraft [Ass22].

2.4.3 Global Navigation Satellite System and GPS-Spoofing

The locational information contained within ADS-B messages is acquired via the geopositioning system Global Navigation Satellite System (GNSS). It is a satellite navigation system in which a receiver can calculate its own position by using satellite signals. Satellites contain an atomic clock and are thereby synchronised with all other satellites. A receiver obtains the exact position of the satellite and then measures the time it takes a signal to be sent from the satellite to the receiver and from this can infer the distance between satellite and receiver. Using multiple signals, the receiver can then calculate its own coordinates [Tip+11]. While water-based localisation requires only 3 signals, for land-based localisation, the signals of at least 4 satellites are required to deliver an altitude estimate as well. GPS, Galileo, GLONASS and BeiDou are examples of GNSS systems deployed by the USA, Europe, Russia and China respective.

GNSS satellites broadcast both on the 1575.42 MHz and 1227.6 MHz frequencies. They transmit the data via two encodings: the Coarse/Acquisition (C/A), an unencrypted code used by civilian receivers and the precision (P(Y)) encoding [Eur21], which can be encrypted and can then only be used by military equipment containing an appropriate decryption key. The coordinates received using the C/A code have

a precision of more than 3 metres, and those of the P(Y) code have one of up to 0.3 metres. [GIS13]

Civilian GNSS signals can be spoofed, which has been explored both in academia [Tip+11], as well as in practice [Ebi15]: Using a Software Defined Radio (SDR) and the GPS Signal simulator GPS-SDR-SIM[3], it is possible to spoof a complete satellite constellation, and virtually transmit the signals of up to 12 satellites spoofing the time and positional information received in the vicinity of the signal [Ebi15; Wik19]. Albeit all GNSS systems are vulnerable to signal overlay attacks spoofing GNSS signals, the commonly used term is not GNSS spoofing but GPS spoofing.

As the ADS-B transmitter has no capability of verifying the accuracy of GNSS coordinates received by the aircraft's GNSS receiver, GPS spoofing attacks would result in the ADS-B messages containing the spoofed coordinates.

3. https://giters.com/pistoletpierre/gps-sdr-sim

3

Device Discovery - A Case Study on Probe Requests in Wi-Fi Networks

The two parties required to establish a wireless connection in a Wi-Fi network are a client and an Access Point (AP). The client could, for example, be a mobile device like a laptop or smartphone, and the AP is typically a router. In order for a client to discover known networks, it has to perform network discovery (cf. Section 2.1.2). In Wi-Fi networks, this can be done via two different approaches: On one hand, a client can perform active discovery, which is an active search for available access points. On the other hand, the AP can advertise itself, which is considered passive discovery. Passive discovery is the more privacy friendly approach, as it doesn't require for clients to disclose any information. But because passive discovery is slower and more energy consuming [Fre15], modern devices instead use active discovery. This chapter first focuses on the privacy implications when using active discovery of Wi-Fi networks by means of probe requests, and the pitfalls this can entail with respect to security and privacy in Section 3.2. An example given is the accidental disclosure of private information observed in the Service Set Identifier (SSID) field during a field study (cf. Section 3.2.3). But even without such unintentional information leaks, probe requests are a source of information that can be used for device fingerprinting and tracking (cf. Section 3.2.4). To counter this, four different approaches to prevent tracking are introduced in Section 3.3: The first dynamically hashes the SSID to allow for the use of hidden networks in a privacy friendly way (cf. Section 3.3.2), the second completely relies on passive scanning instead (cf. Section 3.3.4). The third and fourth approach introduce the use of a generic address (cf. Section 3.3.5)

and generic probe requests (cf. Section 3.3.6), to make active probing more privacy friendly and allow single users to disappear in a large anonymity set, thus making them as indistinguishable from one another as possible.

Relevant Publications The evaluation of the content of probe requests and the resulting privacy implications described in Section 3.2.3 have previously been published in [McD+22]. Generic probe requests are described in [McD+24b], while the proposal of the use of a generic address was published in [McD+24a].

3.1 Preliminary Studies and Related Work

The preliminary work on probe requests and their privacy implications is closely tied to the countermeasures that were implemented to mitigate them. The focus of the first published attacks were the **transmission of SSIDs**:

In 2007, Pang et al. [Pan+07] conducted a study on user tracking in Wi-Fi networks. They demonstrated that the combination of the Media Access Control (MAC) address used in probe requests, along with the transmitted SSID, contains sufficient information for tracking. In another publication, Pang et al. [PS07] analyse the privacy risks associated with both APs transmitting SSIDs and mobile devices transmitting probe requests. In an evaluation of the geoinformation contained on SSIDs collected in a data set in 2004, they discover that approximately a quarter of the devices transmit probe requests containing SSIDs unique to a single city. To mitigate this information leak, they propose the Tryst architecture, designed to conceal confidential information during network discovery. Tryst employs access control primitives with symmetric encryption to reveal information solely to the intended access point while concealing all other information. This, like all the following publications proposing to use cryptographic means to protect probe requests, did not end up being implemented in the standard. In fact, even though the IEEE 802.11w-2009 amendment included cryptographic protection of management frames, probe requests are not protected since they are transmitted before the key exchange in Robust Security Network Association (RSNA) (cf. Section 2.1.4) [IEE20].

In 2014, Cunche et al. presented a method for associating devices based on their transmitted SSIDs, inferring relationships between users [CKB14]. This was documented prior to the deployment of MAC address randomization, when the active discovery using directed probe requests was the prevalent method for network discovery. To enable mobile devices to locate known networks without having to resort

to active discovery, Cunche et al. instead suggest to replace the active approach by geolocation-based service discovery.

Since the transmission of SSIDs in probe requests was shown to be both a trivial identifier for tracking, as well as a source of information on the device's owner, it remains a concern even in newer publications: In 2019, Dagelić et al. [DPČ19] published the results of field studies conducted at music festivals between 2014 and 2018, and analysed the occurrence of SSIDs contained in the recorded probe requests. They highlight the ease with which devices are trackable via probe requests in case the device transmits a unique fingerprint. In a comparison of the data sets recorded over a period of four years, they observed that the number of MAC addresses increased over the years, while the amount of SSIDs decreased. The authors conclude that both the use of undirected probe requests as well as MAC address randomisation increase. Despite its decreasing use, Zhao et al. [Zha+19] were able to use transmitted SSIDs to localise criminal groups via probe requests, constructing a database of SSIDs similar to WiGLE[1] (cf. Section 3.2.3.3). They monitored probe requests in various locations to identify specific SSIDs, enabling the identification and tracking of devices belonging to a specific target group.

The susceptibility to **tracking via an unchanging MAC address** during wireless connections was discussed in the early stages of wireless communication: Gruteser et al. [GG03] warned in 2003 that long-lived interface identifiers can be used to triangulate and track users. To mitigate this, they suggest the use of *disposable MAC addresses*, essentially an early version of MAC address randomisation. They suggest to concatenate a legitimate Organisationally Unique Identifier (OUI), randomly chosen from an official assignment list, to a 24-bit long String that is part of a chain of MD5 hashes. As the focus of their work was not on probe requests, but on an ongoing wireless connection in an environment with multiple access points providing access to the same network, they suggested to use the next rotation of the MD5 hash chain, and thereby a new MAC address, for every new association.

Another early MAC address obfuscation strategy was proposed by Franklin et al. [Fra+06] in 2006: They detect driver-specific patterns in the transmission of probe requests, which they can pinpoint to a specific device by additionally taking the MAC address into account. To circumvent their fingerprinting technique, they suggest *MAC address masquerading*: This entails for one device to use the MAC address of another nearby device for sending probe requests. Two devices probing with the same MAC address, but exhibiting different driver fingerprints, would circumvent their fingerprinting technique.

1. https://wigle.net/

To impede tracking a device via an unchanging MAC address contained in its probe requests, large device manufacturers and Operating System (OS) designers introduced **MAC address randomisation** (cf. Section 2.1.3) starting in 2014. Although intended for privacy enhancement, the lack of standardisation rendered all implementations susceptible to attacks [Van+16]. Ever since, extensive research has been conducted on probe requests, MAC address randomization, and device fingerprinting: In 2015, Freudiger et al. [Fre15] provided insights into the amount of probe requests transmitted by different devices, assessing the effectiveness of MAC address randomization in various devices. Their analysis suggests that both Android and iOS devices transmit probe requests very frequently, up to 2000 per hour, and that if randomisation was used at all, it could be trivially circumvented.

Martin et al. [Mar+17] studied the use of MAC address randomisation in 2017 across a broad variety of devices. With respect to Android devices, they identified a significant amount of manufacturers that implemented MAC address randomisation in a way that persisted parts of the OUI, allowing inferences on the devices via the non-randomised identifiers. This was likely done to circumvent the illegitimate, temporary use of OUIs identifying other manufacturers when utilising 46-bit randomisation (cf. Section 2.1.3), but allowed attackers to identify Google, Motorola, Huawei, Sony, BlackBerry, HTC and LG devices via the OUI of the MAC address, and in some cases using additional information within the probe request. Martin et al. also found out that Samsung devices, at the time, never utilised MAC address randomisation, irrespective of the OS versions used. While the first tests they conducted on Apple devices revealed that it was difficult to infer information from their probe requests as they utilise 46-bit randomisation, an update to iOS 10 introduced vendor tags explicitly identifying their manufacturer. Providing best-practices for the implementation of MAC address randomisation, Martin et al. suggest for manufacturers to use 46-bit randomisation and to re-randomise the MAC address for every transmitted frame.

Three years later, Fenske et al. [Fen+21] reconstructed the study published by Martin et al. to observe the changes in MAC address randomisation and its implementations over time. They note that while in 2016, approximately 82 % of the tested devices were utilising their hardware address when transmitting probe requests, this number was significantly reduced to 56 % in 2020. While a certain OUI identified a large number of Google devices in 2017, Fenske et al. find in 2020 that this OUI was still in use, but only by non-Google Android manufacturers. Google-produced devices instead turned out to consistently use 46-bit randomisation. While they could show that the pervasiveness of MAC address randomisation had increased during the three years between the studies, it was, as of 2021, by no means consistently deployed yet. Gomez et al [GGP22] reached the same conclusion in 2022: After analysing data

recorded in public Wi-Fi networks across Latin America between 2016 and 2021, they demonstrated that its wide-spread use only began in 2020.

Supplemental to the previously mentioned findings, Freudiger et al. [Fre15] identified an additional vulnerability of probe requests: the possibility to **track users via their unrandomised, sequential sequence numbers**. While this introduces another attack vector, its mitigation is simple: to randomise the initial sequence number with every burst. In addition to these findings, Freudiger et al. also observed that recent mobile operating systems probe only for SSIDs of hidden networks. In theory, this development, in combination with randomised MAC addresses and sequence numbers, signified a large step towards privacy-friendly active discovery.

In practice, however, neither were implemented in a way that reliably protected user privacy: In 2016, Vanhoef et al. [Van+16] presented two **attacks capable of revealing a device's real MAC address**: A new version of the *Karma Attack*, and the *Hotspot 2.0 Honeypot*. In the original Karma Attack [DM05] presented in 2005, an attacker would record SSIDs transmitted within probe requests and dynamically open a fake AP transmitting these SSIDs to coax clients into connecting to such an AP. During the time of the experiment of Vanhoef et al., the omission of SSIDs to reduce tracking was already establishing among manufacturers as a privacy-preserving technique. The new revision of the Karma Attack therefore relied on a large number of users connecting to the same known public APs: By providing a fake AP transmitting 5 popular SSIDs, they were able to record connection attempts and therefore the real, or at least a per-network MAC address, of more than 17 % of the observed devices. Note that at this time, per-network MAC addresses were not overly common yet, with only Windows devices making use of it.

The other attack Vanhoef et al. devised to reveal a device's real MAC address was the Hotspot 2.0 Honeypot: Here, a service discovery mechanism contained within the 802.11u standard, which is commonly called Hotspot 2.0 (HS2.0), is utilised: To receive information on HS2.0 capabilities of APs, clients have to explicitly query for them using an ANQP query. Apart from Apple devices, all devices with at least one HS2.0 network configured transmitted ANQP requests using their real MAC address, amounting to up to 16 % of the devices. Given that the technology was very new at the time, the authors expected a steadily rising number of devices exhibiting this data leak, if not patched adequately.

In the aforementioned publication, Vanhoef et al. additionally revealed the possibility to **track devices by constructing a device fingerprint over all Information Element (IE) fields** (cf. Section 2.1.2) contained in probe requests. They conclude that MAC address randomisation alone is inadequate for thwarting tracking attempts. This publication also started the research into fingerprinting techniques using the IE, with the subsequent publications improving on association methods, choosing unique

IE fields and including emerging technologies into the association process: Gu et al. [Gu+20] demonstrated that elements in the frame body of probe requests can be utilised for device fingerprinting. They implement their frame association using deep learning methods. To protect from such an attack, they propose the encryption of probe requests via the symmetric stream cipher ChaCha20 as a means of protection against attackers. Tan et al. [TC21] model the association of probe request frames in a flow network and utilise minimum-cost flow optimization, thereby tracking users via their mobile devices in a shopping mall. They achieve an accuracy of over 80 %. [HTC23] extend the work of Tan et al., and improve the association in such a way so it does not require offline pre-calibration and is approximately 270 times faster, while reaching comparable discrimination accuracy and V-measure scores. By combining fingerprinting techniques with clustering approaches, Uras et al. [Ura+20] and Pintor et al. [PA22] successfully circumvent anti-tracking techniques. Their approaches at defeating MAC address randomisation reach an accuracy between 65.2 % and 91.3 %, respective up to 92 %. Since all of these attacks commonly use the IE content of the probe requests to track devices, the most straightforward means of reducing this attack surface is to **minimise the IE content** [Van+16; Mar+17; Fen+21], as will be later explored and analysed in depth in Section 3.3.6.

3.2 Extended Privacy Risks

The related work regarding active discovery shows that while privacy features are emerging, various identifiers contained within probe requests continue to be used to track devices. This section contextualises these privacy risks: First, a structured analysis of privacy risks and features in mobile operating systems for active discovery highlights which devices are prone to information disclosure in Section 3.2.1. Subsequently, an evaluation of the information leakage in the shape of transmitted SSIDs is presented in Section 3.2.3. Both are based on the results of the paper [McD+22], which are presented here in the context of this thesis. Then, Section 3.2.4 provides an evaluation of the risks of device tracking via additional features contained within probe requests through an analysis of the attacks presented in related work, a condensed version of which was also published in [McD+24b].

3.2.1 Active Discovery Privacy Features in Mobile Operating Systems

While the first implementations of probe requests were easily trackable due to persistent identifiers like the MAC address and SSID(s), the privacy risks entailing

	Apple iOS				Android				
	8	10	14	15	8	9	10	11	12
Release year	2014	2016	2020	2021	2017	2018	2019	2020	2021
Market Share in % in 2021	< 0.1	1.0	35.9	53.4	10.2	13.5	27.0	35.4	1.9
Randomised MAC ...									
- while probing	✓	✓	✓	✓	✓	✓	✓	✓	✓
- per connected SSID	-	-	✓	✓	-	(-)*	✓	✓	✓
- after resetting settings	-	-	✓	✓	-	-	(-)*	(-)*	(-)*
New random MAC after ...	-	-	-	6w	-	-	-	(-)†	(✓)‡
Private Address by default	-	-	✓	✓	-	-	✓	✓	✓
Modify distant Network	-	-	-	-	✓	✓	✓	✓	✓
Manually added == hidden	Automatic detection of hidden				✓	-	-	-	-
Probe with SSID	Only if hidden detected				if man. added	If explicitly declared hidden			

*: Only choosable via Developer Options
†: If use of non-persistent MAC is chosen via Developer Options, a new MAC is set (a) for every new connection establishment (b) every 24 hours, unless a connection is still established or (c) if both the DHCP lease has expired and the device has been disconnected for 4 hours
‡: Non-persistent randomisation used in case a network suggestion app specifies this via the API or in case of a connection to an open network without a captive portal. Otherwise, persistent randomisation is used.

Table 3.1: A comparison of privacy features when using probe requests in Android and iOS. Adapted from [McD+22].

the use of probe requests have been reduced over time. To quantify this reduction, this section compares privacy features of selected OS versions and the risks still inherent to their use. Table 3.1 shows a comparison of the privacy features introduced with different Android and Apple OS versions, which was previously published in [McD+22], and reflects the state of active discovery privacy features of devices running the mobile OS Android 12 and lower or iOS 15 and lower. The comparison was compiled in the context of the work for [McD+22] in 2021, and is required for the evaluation presented in Section 3.2.3.

iOS 8 was the first Apple OS that **introduced MAC address randomisation** in 2014, while Android introduced it in Android 6 in 2015. After this, the privacy features were continuously improved and new features implemented: While iOS 8 to 10 and Android 8 and 9 only used randomised MAC addresses while probing, Android 9 already allowed **setting a randomised Locally Administered Address (LAA) per network** in Developer Options, and starting with iOS 14 and Android 10, randomised LAA per network were used by default. And while Apple devices starting with iOS 14 additionally used **new randomised MAC addresses after resetting the settings** by default, Android devices starting with Android 10 persist the MAC address even after forgetting and re-adding a network, as the MAC address is dependant on

the network profile parameters. However, if explicitly directed via the Developer Options, devices running Android 10 and higher *can* employ this feature. As such features in Developer Options are difficult to find, they are typically only activated by technically sophisticated individuals explicitly interested in privacy enhancing technologies, and likely not widely adopted. This also applies to the **periodic renewal of randomised LAA**: While iOS 15 distributes new randomised MAC addresses per network every 6 weeks, this feature is only available in Android 11 and 12 when the use of a non-persistent MAC address is enabled via Developer Options. Starting with Android 12 and higher, non-persistent randomisation is used by default "for some networks" [Doc24], meaning either in case a network suggestion app specified its use via the `WifiNetworkSuggestion.Builder#setMacRandomizationSetting` API, or in case the network is an open network without a captive portal. In such cases, the LAA is randomised periodically, either if

(a) the DHCP lease expired and the last disconnect happened more than 4 hours ago, or

(b) the randomised MAC address is older than 24 hours.

If either case does *not* apply upon reconnection, the previous randomised MAC address is used.

A feature that is inherently useful to modify probing behaviour in case of added hidden networks is the possibility to **modify distant networks**. In Android, it is possible to modify all entries of the Preferred Network List (PNL) at any time. Since this allows to remove unused networks, and particularly hidden networks that the mobile device might be probing for using its SSID, this is immensely useful from a privacy perspective. Apple devices running iOS 15 and lower only allow modifications of networks in reach. In order to modify networks out of reach, a MacBook can be used to alter the iCloud Keychain that the networks are stored in. Without access to a MacBook, the modification without physical proximity is impossible, which means that a mobile device might still probe for hidden networks contained in the PNL, which are actually not in use anymore. This, in fact, changed with iOS 16, from which OS version onwards the modification of out-of-reach networks is possible [Myr22].

Another privacy feature of all tested iOS OS, as well as Android devices starting with Android 9 concerns hidden networks. While Android 8 devices still assumed that **manually added networks** should be treated like hidden networks, Android 9 and higher assumes they are *not* hidden networks, unless explicitly chosen in the network settings. When manually adding a network to iOS devices, the devices test whether the network is currently in reach, and whether it is set as a hidden network. If it is not in reach, it can not be added to the PNL. This directly influences whether

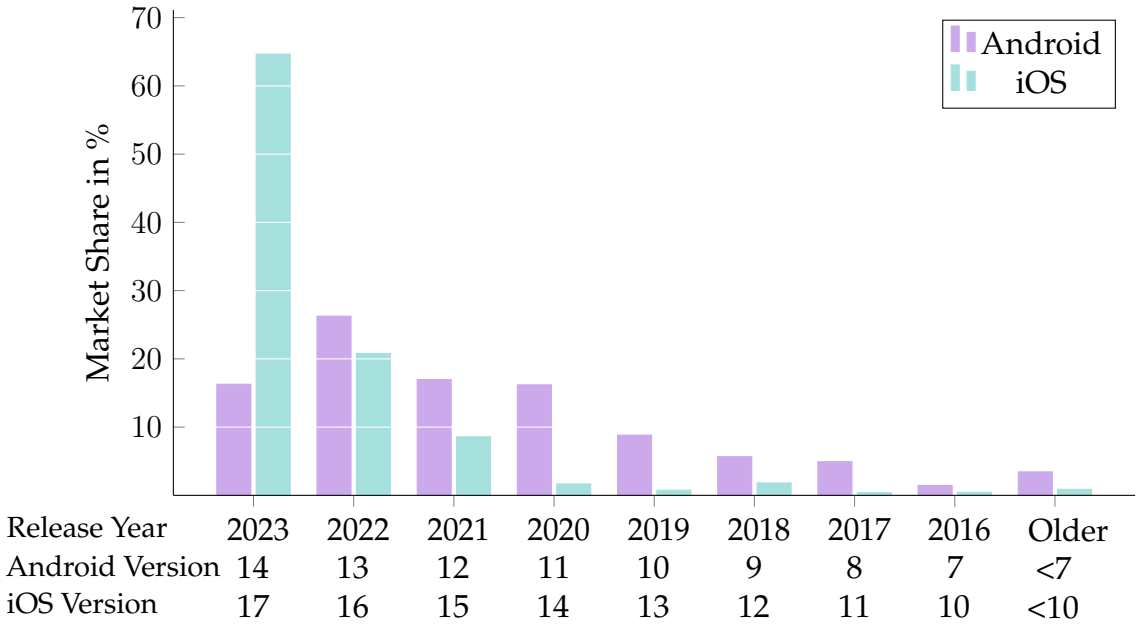


Figure 3.1: The market share of the OS distributions sorted by descending release year. The OSs released in 2023, Android 14 and iOS 17, are installed on 16.28 %, respective 64.66 % of the corresponding devices as of March 2024. The graph was accumulated using data published in [sta24a] and [sta24c].

or not devices **probe for networks using their SSIDs**: iOS devices running iOS 8 to 15 only do so if the network was detected to be a hidden one. Android 8 devices assumed that a manually added network was a hidden one and therefore probed for it using the SSID, but devices running iOS 9 and higher only send probe requests containing the network SSID when the network is explicitly declared to be a hidden network in network settings.

3.2.2 OS Support Lifespans

One contributor to the pervasiveness of OS that do not implement privacy-preserving techniques, and therefore a likely contributor to the transmission of the SSID content evaluated in the subsequent section, is the support span of mobile device OS. Fig. 3.1 shows the market shares of iOS and Android OS versions as of March 2024, accumulated from [sta24a] and [sta24c]. The figure underlines that while Apple devices tend to be used running the latest OS versions, the Android OS version distribution exhibits a flat curve, with Android 14, released in 2023, only reaching a market share

of 16.28 %, and the previous OS, Android 13, released in 2022, reaching a market share of 26.26 % as of March 2024 [sta24a].

The primary reason for this divergence in the use of latest operating systems is the willingness of device manufacturers to provide long-term support for their hardware. Apple strives to provide their devices with both security updates and general updates for as long as the hardware supports it: iOS 16 was released in September 2022, and iOS 17 in September 2023, and Fig. 3.1 shows that in March 2024, more than 85 % of the Apple devices were using these latest OS, with 64.66 % of the devices running iOS 17. All Apple iPhones produced between 2015 and 2018 were able to get at least 6 major version updates, and their support is only discontinued once the hardware fails to be able to sustain the latest update: The iPhone 8, for example, first released in 2017, received continuous major updates until iOS 16 in 2022, and while it does not receive the major update to iOS 17 anymore, iOS 16 is still supported and patched in case of vulnerabilities today [Wik24c]. A support span of seven years and more explain the very large market share of the latest iOS version in Fig. 3.1, and the guaranteed security updates during the whole time ensure a relatively safe use and considerably better privacy guarantees than the prominent Android manufacturers, which make up 71.54 % of the market share (see Fig. 3.2).

The very limited support span was a common point of criticism towards Android manufacturers: Devices like the Google-produced Pixel 4a and Pixel 5a had a support span of 3 years [Wik24d; Wik23]. However, the European Union recently adopted regulation C(2023)3538 "Designing mobile phones and tablets to be sustainable – ecodesign" [Eur23a], because of which manufacturers will be **required to provide their devices with at least five years of security and functionality updates**, starting on May 9th 2025 [Eur23b]. In reaction, several manufacturers of Android devices recently started to introduce longer support periods, with Samsung, the manufacturer with the biggest market share in Android devices of 23.85 %, guaranteeing four years of major updates [New22] and the latest Pixel 8 Pro, manufactured by Google with its market share of only 1.04 %, with a guaranteed support span of seven years [Gib23]. While this is a positive trend for new devices, devices produced and bought before these self-imposed commitments by manufacturers remain vulnerable to attacks once the support span ends. Additionally, the commitment is voluntary and not upheld by all manufacturers.

Except in the case of Apple, which has maintained considerably longer support spans of five years or more since 2013 [Wik24c], commitments by companies to uphold long support spans are a relatively new development. As a result, the majority of Android devices currently in use are running outdated OS: As can be seen in Fig. 3.1, 57.46 % of the Android devices run either the 2021 Android 12 or older OS. Only 42.54 % of the devices run either the latest Android 14 or the 2022 version Android

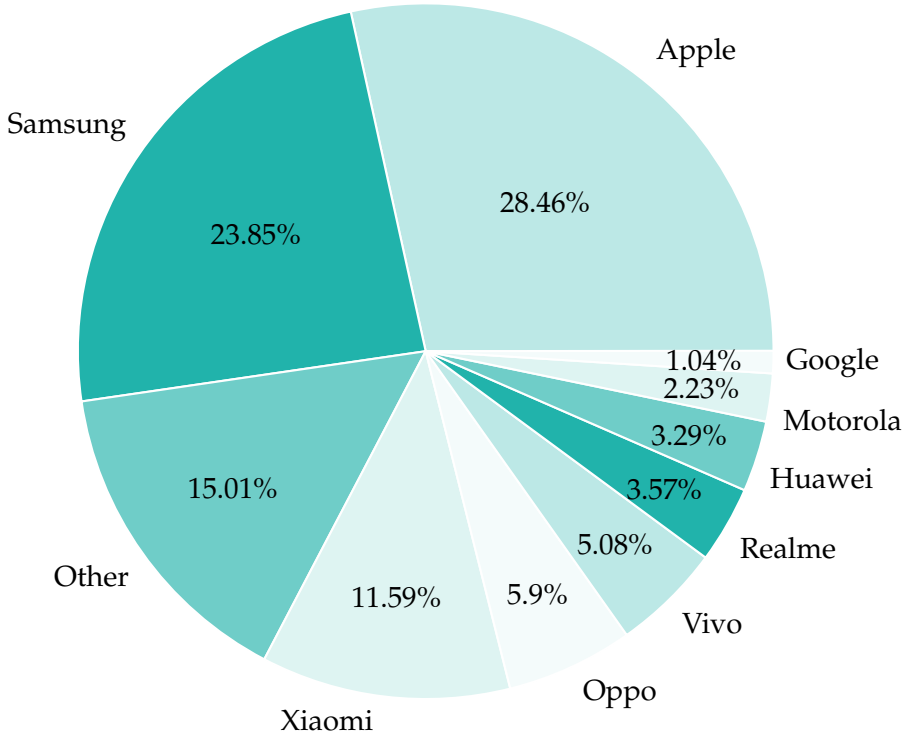


Figure 3.2: Market share of mobile device manufacturers in 2024. The data was accumulated from [sta24d].

13. This is a dangerous tendency, since older devices without security updates are open to attacks and prone to contain vulnerabilities. They are also very likely to leak more data than devices running recent OS, since many privacy features were only introduced in newer OS, as can also be observed in Table 3.1. Such data leaks, resulting from insufficient privacy guarantees, are examined in more detail in the subsequent section.

3.2.3 Evaluation of SSIDs transmitted in Probe Requests

Several publications have previously evaluated SSID content in probe requests: Some emphasise the potential for device tracking via the fingerprint that unchanging identifiers entail [Pan+07; DPČ19; Zha+19]. Others infer relationships between device users via their PNL [CKB14]. Still others quantify the information leakage caused by SSIDs in probe request [WK18]. The research is extended by [McD+22], in which a qualitative analysis on the SSID content was presented: This section is based on the field study performed during the aforementioned paper, and contextualises its results for this thesis. To gain insight into the propagation of SSIDs, probe requests

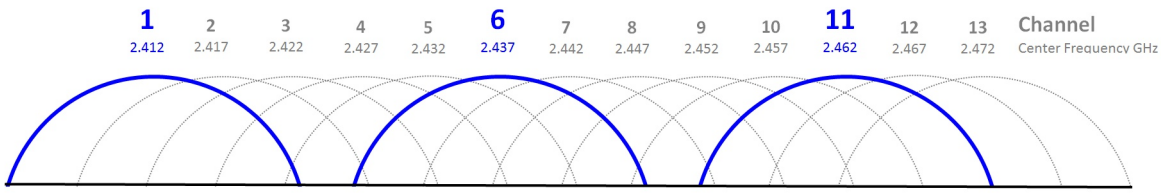


Figure 3.3: The channels of the 2.4 GHz spectrum, with the non-overlapping channels 1, 6 and 11 emphasised in bold. Image source: [Jonb].

were recorded in a field study, over the period of three hours on three different days. This was done using six off-the-shelf antennae, to monitor both channels 1, 6 and 11 of the 2.4 GHz spectrum, as well as channels 36, 40 and 48 of the 5 GHz spectrum. The first were chosen as they are the non-overlapping channels of the 2.4 GHz spectrum, as also illustrated in Fig. 3.3. The latter three were chosen as representatives of the 5 GHz spectrum, in which the non-overlapping channels are 36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157 and 161 [Jona]. The capture of the 5 GHz spectrum was limited to channels 36, 40 and 48, as surveilling more channels would have increased the complexity considerably.

To ensure ethical data collection and refute privacy concerns, the field study was performed in close consultation with the ethics committee of the University of Hamburg Informatics department. It received a vote of approval under case number 002/2021, with conditions and recommendation including the following:

- The data collection had to be restricted to allow for people to opt out, both in terms of content regulation as well as proximal limitation.
- The area of study had to be outfitted with warning signs so as to inform passersby of the data collection.
- The resulting data had to be additionally saved in encrypted form.

After the research was conducted, the resulting paper was resubmitted to the ethics commission for final approval. This resubmission received the confirmation of satisfying the commission's requirements.

3.2.3.1 General Evaluation and Contextualisation of the Data Set

The data set recorded in the field study contains 252 242 probe request, 23.2 % of which contain SSIDs. Fig. 3.4 puts this amount into perspective with other recordings of probe requests. It shows a comparison of probe request data sets recorded in various years, and the **percentage of probe requests contained in it that transmitted an SSID**: Barbera et al., who collected the Sapienza data set [Bar+22], published their

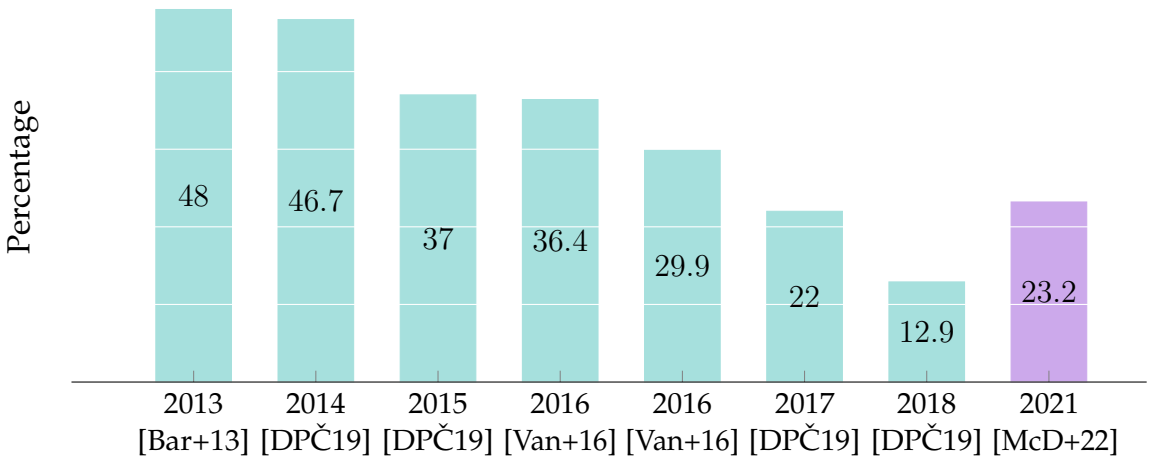


Figure 3.4: The amount of SSIDs recorded per year across various publications. The x-axis shows year and publication that the value was published in, and the y-axis the percentage of probe requests containing an SSID.

data analysis on the data set in [Bar+13], and found that while the probe requests contained an SSID in 34.1 % up to 55.5 % of the cases in the individual data sets, the overall amount of probe requests containing an SSID was 48 % in this collection. Dagelić et al. [DPČ19] recorded probe requests at music festivals between 2014 and 2018 and observed a steady decrease, with the 2014 data set containing 46.7 % of probe requests with an SSID, and the 2018 data set only 12.9 %. As they did not give exact values for 2015 and 2017, but only illustrated them in comparison to the 2014 and 2018 values, the numbers used in Fig. 3.4 had to be approximated from [DPČ19, Figure 4]. Vanhoef et al. also analysed two data sets they recorded, with one of them containing 36.4 % of directed probe requests, and the other 29.9 %. In direct comparison, the amount of 23.2 % of probe requests containing SSIDs recorded in [McD+22] stands out, since it is considerably higher than both values for 2017 and 2018 measured by Dagelić et al. On the other hand, their measurements were taken at a music festival. It is likely that most attendants were rather young and might therefore either have been tech-savvy enough to maintain an up-to-date OS, or a rather new mobile device. The recordings from the field study introduced in [McD+22] on the other hand, were taken around noon, in the touristic city centre of Lübeck, Germany. Since the study participants therefore differ immensely, their habits in maintaining a patched and recent OS might differ, too. The **numbers recorded in [McD+22] also correlate with the market share of Android devices** of the time [sta21a]: In December 2021, 10.2 % of the Android devices were still running Android 8, and 12 % were using OS older than Android 8. As mentioned in Section 3.2.1, such devices employed insufficient privacy enhancing technologies

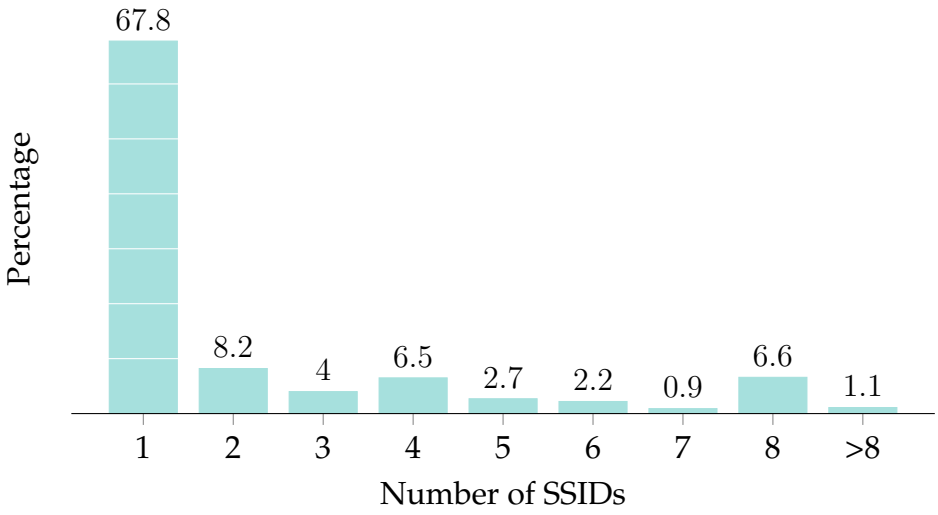


Figure 3.5: The distribution of specific amounts of SSIDs recorded per cluster.

if any, e.g. they still considered manually networks to be hidden networks, and therefore tended to probe using an SSID considerably more liberally than devices running newer OS. While the market share of Android devices was around 70 % at the time and iOS devices around 29 % [sta21b], the percentage of probe requests containing SSIDs was only slightly higher than expected based on these market shares and the privacy enhancing technologies deployed.

During the measurement of the 252 242 probes recorded in total, 46.4 % were recorded in the 2.4 GHz spectrum, 24.7 % of which contained an SSID. The captures of the 5 GHz spectrum encompassed 53.6 % of the probe requests, 21.9 % of which contained an SSID. The probe requests were first grouped into *bursts*, with one burst containing all requests transmitted within 4 seconds via the same MAC address. They were then partitioned into *clusters*: A cluster contains all requests with an exactly overlapping PNL. While this approach ensures that misclassification of two devices containing a slightly overlapping PNL can not occur, it simultaneously groups all devices transmitting just one overlapping SSID into the same cluster. As this publication makes no attempt to reassociate devices despite MAC address randomisation, this is irrelevant; grouping devices into clusters via their PNL was only a useful step for the following analysis, since it allowed to evaluate unique PNLs. Fig. 3.5 shows the **distribution of the number of SSIDs contained in probe requests**. Most devices transmitted only one SSID. However, 32.2 % of the devices transmitted two or more SSIDs, making their fingerprint unique enough for tracking: The more SSIDs a device transmits, the more unique its fingerprint becomes. This is a considerable decline in comparison to the 2016 findings of Vanhoef et al. [Van+16]: In their analysis, they identified that between 53 % and 64.8 % of the bursts entailed

a unique fingerprint due to containing two or more SSIDs. Barbera et al. [Bar+13] on the other hand, evaluated in their 2013 publication that of the directed probe requests, 50 % transmitted only one SSID, 30 % transmitted two to ten SSIDs and the remaining 20 % of the devices transmitted more than 10 SSIDs.

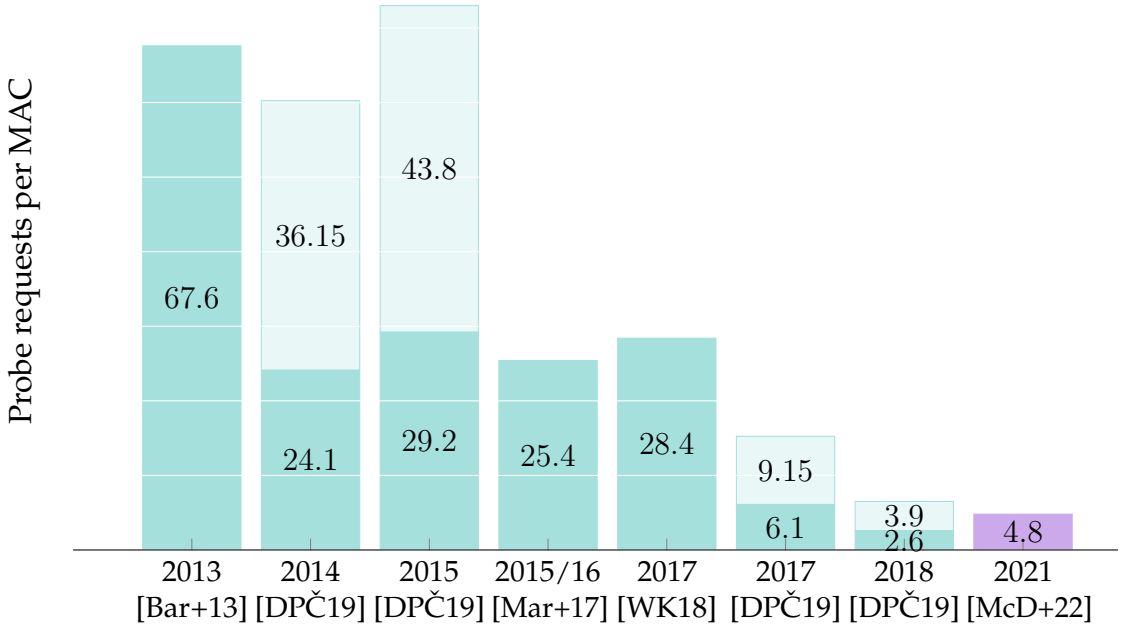


Figure 3.6: Visualisation of the decreasing number of probe requests transmitted per unique MAC address over the years 2013 to 2021, showing privacy mechanisms implemented after 2013 slowly taking effect. The green bars show the measurements from Barbera et al. [Bar+13], Martin et al. [Mar+17], Waltari et al. [WK18] and Dagelić et al. [DPČ19]: In 2013, Barbera et al. recorded 67.6 probe requests per MAC address in their Sapienza data set. Martin et al. recorded an average of 25.4 probe requests per MAC address in a long-term study spanning the beginning of 2015 to the end of 2016. Waltari et al. performed measurements across various locations and events. The event represented in the graph was the 2017 EuroSys conference, where 28.4 probe requests were recorded per MAC address. Dagelić et al. recorded between 29.3 and 2.6 probe requests per MAC address at music festivals between 2014 and 2018. As they only used one interface to record probe requests, their entries are additionally normalised with the light green stroke to reflect the probe requests that were likely missed in their capture. The purple bar shows that an average of 4.8 probe requests were captured per MAC address in the field study presented in [McD+22].

By dividing the total number of captured probe requests by the number of unique MAC addresses, one can determine the **average ratio of probe requests sent from**

each MAC address. This provides an overview of the **distribution of MAC address randomisation**: the more devices that use unchanging MAC addresses, i.e. their Universally Administered Address (UAA) instead of a randomised LAA, the lower the number of devices employing MAC address randomisation. The average amount of probe requests transmitted per MAC address in the field study published in [McD+22] was 4.8. This is also visualised in Fig. 3.6, where it is additionally compared to the results of Barbera et al. [Bar+13], Martin et al. [Mar+17], Waltari et al. [WK18] and Dagelić et al. [DPČ19]: As Barbera et al. captured a total amount of 11 136 711 probe requests transmitted via 164 740 MAC addresses in their 2013 data set, the ratio of probe requests sent per MAC address is 67.6. Martin et al. captured probe requests over a course of two years, from the beginning of 2015 to the end of 2016. Their data set contains a total of around 66 million probe requests, originating from 2.6 million different source MAC addresses. Each MAC address was therefore used on average to send 24.5 probe requests. Waltari et al. conducted a study capturing probe requests across six distinct events: the EuroSys 2017 conference, a pop concert, a workers' day celebration, a movie, a shopping mall, and a university campus. Since only the conference capture contained a remark concerning its year of origin, it's value of 28.4 probe requests per MAC address was the only one taken into account for Fig. 3.6. The other captures had a distribution of 24.8, 9.4, 18.5, 12.6 respective 30.1 recorded probe requests per MAC address, but since the Waltari et al. conference paper was published in 2018, it is safe to assume that all other values were also recorded in or prior to 2018. All values are considerably higher than those recorded by Dagelić et al. after 2015: The analysis Dagelić et al. performed between 2014 and 2018 shows a rapid decline of probe requests sent per MAC address. But since they only used one Wi-Fi interface to capture probe requests, they likely also only monitored one channel. Since devices typically cycle through a number of channels when transmitting a burst of probe requests from one randomised MAC address, they likely lost a significant amount of probe requests, possibly up to two thirds, as can be gleaned from Fig. 2.4, which was published in [WK16]. In this 2016 publication, Waltari et al. compared probing patterns of different devices and identified that most devices use nearly all channels within a burst, with the exception of channels 12 and 13, which are not used in North America. Since the channels are overlapping, monitoring one channel, e.g. channel 6, additionally provides the probes transmitted on the two channels below (4 and 5), and above (7 and 8), encompassing up to five channels. Multiplying the amount Dagelić et al. recorded by 2.5 can therefore provide an approximate estimate what a recording on all non-overlapping channels would have shown. This normalisation is reflected in the light-green overlapping bar stacked behind the entries of Dagelić et al. shown in Fig. 3.6.

Altogether, the graph shows a steady increase of the use of MAC address randomisation: Put into context, the amount recorded in the field study reflects the increasing

use of privacy enhancing technologies employed in active discovery. While the 2013 recording likely did not record the use of any privacy preserving techniques, the captures between 2014 and 2017 reflect a general increase in MAC address randomisation, while the remaining values recorded after 2017 indicate the almost-ubiquitous use of MAC address randomisation. An inspection of the *directed* probe requests reveals that the average number of probe requests transmitted from a single MAC address is 11.2. This finding supports the hypothesis that devices transmitting a PNL are **likely running older OS** which are less likely to employ MAC address randomisation.

In the following, the content of the clusters extracted as previously detailed are analysed as to their SSIDs.

3.2.3.2 Evaluation of SSID Content

Upon closer inspection of the SSID contents within the captured probe requests, it became evident that several patterns indicated the presence of passwords, typographical variations, and personal information within the PNL. The reason can likely be found in the behaviour of Android devices summarised in Section 3.2.1: Devices running Android 8 and below treat manually added networks as hidden networks. The assumption established in [McD+22] was that users trying to manually set up a network connection via the advanced network settings mistakenly entered wrong strings in the place of the SSID. Since the devices consider these manually added networks as part of the legitimate PNL and continue to probe for them using directed probe requests, their device's fingerprint is easily recognisable. In the following, the methodology to differ between the various types of entries and the resulting privacy implications are explained in more detail.

Password Leaks A significant portion – 11.8 % of the probe requests – contained numeric strings of 16 digits or longer, which are commonly used as initial passwords for home routers in Germany (e.g. Telekom home router or FritzBox). Some were entered in one consecutive string, while others were separated by spaces, dots or commas every four digits, e.g. `1234567812345678`, `1234 5678 1234 5678` and `1234.5678.1234.5678`. This is a typical typeset chosen to improve readability of initial passwords, and the "SSIDs" inserted like this were often not the only entry of the PNL, but rather one of several insertions containing the same numbers but various delimiters. Some of the entries were even prepended by strings like `PW:`, `WPA:` or `(WPA/WPA2:)`. Leaking passwords are not only critical because the users are likely oblivious to the fact, but also because the leaked passwords were, in most cases, part of a bigger PNL: only 2.8 % of the passwords were the only entry of the PNL, whereas

the rest contained other entries, and potentially even the SSID of the network the password belonged to. Such leaks are particularly critical since they could be easily exploited: If an attacker were to use the captured probe requests to set up a rogue access point using the SSIDs contained within the requests, and additionally the transmitted the password, the mobile device would be tricked into connecting to this network. This would grant the attacker full control over the device's network traffic. Since this constitutes an active attack that undermines user privacy and data security, its implementation was omitted and only its workings described.

Another way such password leaks could be exploited is by using the Wi-Fi mapping service WiGLE (cf. Section 3.2.3.3): An attacker with enough criminal energy could monitor probe requests to discover content reminiscent of passwords. They could then query WiGLE as to the whereabouts of other networks contained within the particular PNL or follow the victim around, as tracking them via their probe request fingerprint is trivial in case of such unique PNL entries. The attacker can then manually try out the potential passwords, either in the places the victim visits or in those they could identify via WiGLE.

Typographical Variations of SSIDs Another unexpected finding in the data set collected during the field study were multiple entries of the same SSID in various spelling variations. Such typographical variations to an SSID were likely entered manually in order to identify the correct spelling of a known SSID. Examples for such variations are groups of SSIDs such like `HomeNetwork`, `home network`, `HOME_NETWORK`, all contained in the same PNL. To extract all such occurrences, the **edit distance** over all SSIDs of a burst was calculated: The edit distance is defined by the amount of operations required to transform a string into another. The operations can be deletions, insertions or substitutions of characters. To account for variable string lengths, the result is **normalised** with respect to the string length of the longer string; the resulting edit distance is therefore divided by the length of the longer of the two SSIDs. A higher normalised edit distance indicates more differences between the strings, with a maximum possible value of 1 (completely different strings) and a minimum of 0 (identical strings). Additionally, as a comparison of a lower-case letter and the same upper-case letter would result in an incrementation of the edit distance and result in an edit distance of 1 between the two strings `network` and `NETWORK`, all strings are transformed to lowercase prior to comparison. The threshold is then set to 0.3: This way, two strings whose letters differ in less than 30 % are considered typographical variations of one another. The high threshold is required to accommodate for short SSIDs: While multiples of long SSIDs that differ in two or three positions can easily be recognised via their edit distance, a threshold of 0.3 would not consider the SSIDs `LOL` and `LEL` to be variations of one another. Simultaneously, setting the edit distance higher could result in very dissimilar SSIDs

still being considered multiples of one another, e.g. `NETWORK` and `WORKART` has an edit distance of 0.43. Setting the threshold to 0.3 allows for a compromise between false positive and false negative classification. To still ensure that distinct network names that likely belong to two different networks despite their low edit distance are not misclassified as variations of one another, a manual inspection additionally verified the findings. This could happen, for example in the case of manufacturer-set SSIDs like `Fritz!Box 7490` and `Fritz!Box 7590`, but was, however, not found in the dataset.

Using this evaluation technique, it was possible to identify that 19.9 % of the transmitted SSIDs can be considered a typographical variation of an SSID contained within the same PNL. While such variations might not pose an imminent security risk like exposed passwords do, they still increase the fingerprint of the transmitting device drastically and thereby facilitate tracking it.

Further (Personal) Data Contained in Probe Requests In addition to the potential passwords and typos, 106 unique names were identified - some first names, others last, some even a combination of both - transmitted 3339 times over the course of the three hour captures. Three e-mail addresses were altogether transmitted 36 times, and the names of 92 holiday homes or locations, which were transmitted 1257 times. A surprising find was a string that was likely the internal Wi-Fi password of a local store, as it was prepended by the string `PW:` and then contained the (very unique) store name. Additionally, the name of a local hospital in various typographical variations was transmitted 15 times. Particularly the names, e-mail addresses and the hospital name can reveal personal information on the device owners, ranging from their name and contact information to health-related sensitive information like a hospital stay. All of these findings can be considered at least confidential if not personal data, and are most likely transmitted without the device user's knowledge or explicit consent.

3.2.3.3 Verification via Geolocalisation

Devices that transmit probe requests containing SSIDs not only facilitate tracking by providing a fingerprint of the device, but also potentially violate the location privacy of their owners: using a wireless network map like WiGLE[2], users can input SSIDs into a web interface and receive coordinates of locations of these SSIDs in return. An example is visualised in Fig. 3.7: Here, a manual search via the web interface returns all locations of the network `eduroam` in a partial map of Hamburg. WiGLE

2. https://wigle.net/

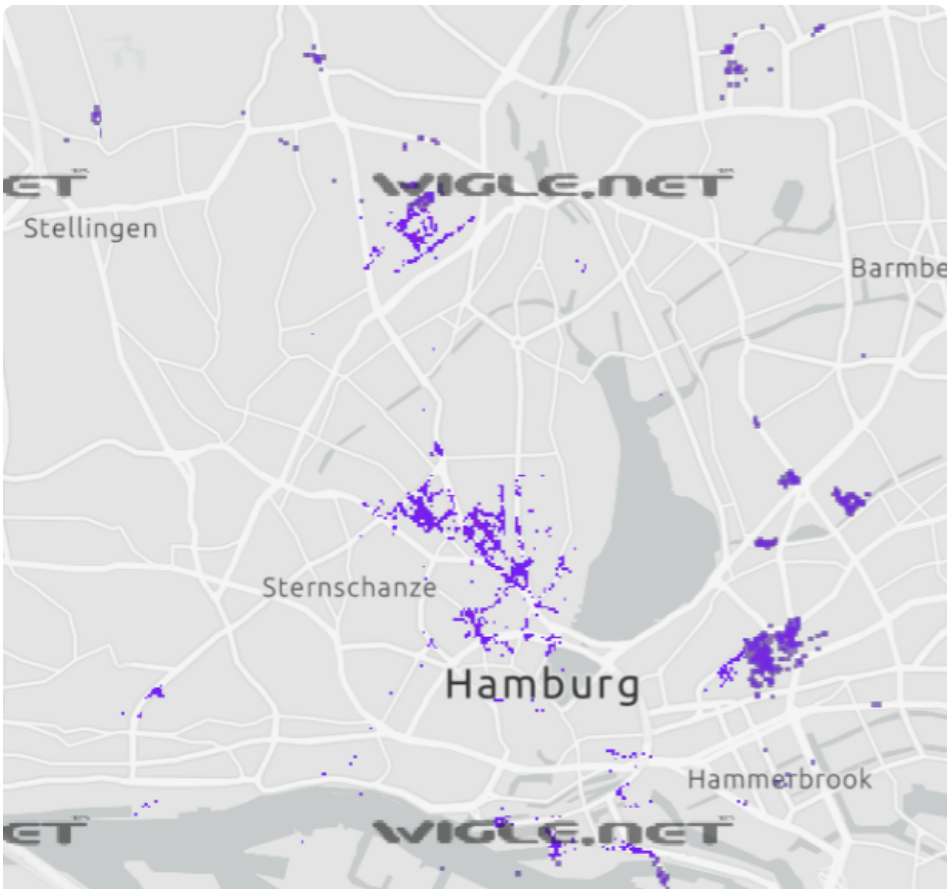


Figure 3.7: A partial map of Hamburg returned in a search using WiGLE, with purple marks highlighting all locations of the network eduroam. Image source: https://wigle.net.

also allows large-scale automated searches via an advanced search interface, which was the approach taken to verify the results of the previously introduced findings via geolocalisation. The attempt to localise all recorded SSIDs using WiGLE yielded the results visible in Fig. 3.8: Out of 1478 unique SSIDs, 38 were unresolvable as they contained special characters, 334 could be pinpointed to a unique location and 377 to multiple locations. 729 or 49.3 % were not localisable, which means that they are either hidden networks that hadn't been mapped in WiGLE or didn't exist. To verify the hypothesis that some of the recorded SSIDs were typos or passwords, their existence was additionally evaluated separately, as shown in the following.

Password Evaluation The strings taken into account to investigate the resolvability of potential passwords were selected from the set of SSIDs according to the following criteria:

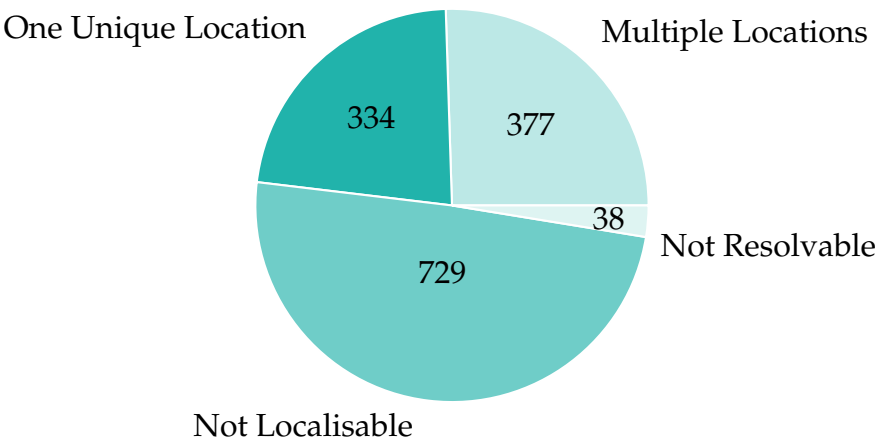


Figure 3.8: Resolving SSIDs in WiGLE: Out of the 1478 SSIDs captured during the field study, 729 were unresolvable, 334 resolved to one unique location and 377 to multiple locations. 38 were completely unresolvable as they contained special characters.

(a) They had to consist of 16 or more numeric digits, or

(b) contain trigger strings like `pass`, `pw`, `wpa` or `kennwort` (the German translation for password).

The hypothesis regarding the resolvability of the SSIDs contained in this subset is the following:

> **Hypothesis:** Most SSIDs contained in this data set are unresolvable because they are accidentally inserted passwords rather than legitimate network names.

The results of this evaluation largely support the hypothesis: As visualised in Fig. 3.9, out of the 78 strings identified as potential passwords, only one was resolvable to a unique location, while the rest could not be mapped to existing networks. This reinforces the idea that most were incorrectly inserted passwords and leads to the following conclusion:

> **Result:** Since most strings identified as potential passwords can not be mapped to actual networks, they are highly likely to be incorrectly inserted passwords.

Evaluation of Typographical Variations of SSIDs The same evaluation applied to the subset of SSIDs containing potential passwords was also conducted on identified SSID multiples, which may represent typographical variations of one another.

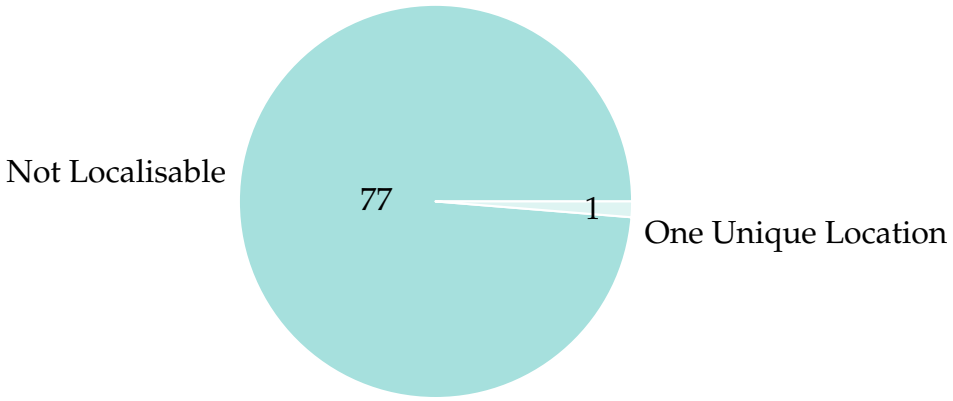


Figure 3.9: Resolving passwords in WiGLE: Out of the 78 SSIDs identified as potential passwords, only one could be resolved to a location using WiGLE.

All SSIDs within a PNL that had a normalised edit distance of 0.3 or lower (cf. the paragraph *Typographical Variations of SSIDs* of Section 3.2.3.2) were considered typographical variations of one another. In this context, the hypothesis as to the amount of resolvable SSIDs was the following:

> **Hypothesis:** More than 50 % of the SSIDs identified as typographical variations of one another are unresolvable, suggesting that many PNLs contain one legitimate SSID and one or more typographical variation.

The results are shown in Fig. 3.10: Out of the 296 strings identified as typographical variations, 38.2 % were resolvable, which supports the hypothesis. 47 of those could be pinpointed to one unique location, and 66 resolved to multiple locations. The remaining 183 SSIDs could not be localised using WiGLE. These results lead to the following conclusion:

> **Result:** While a number of the SSIDs correspond to real networks, the remainder can not be located as the networks likely do not exist. The unresolvable SSIDs are likely to be typos of the resolvable SSIDs.

Limitations While the verification via geolocalisation is a legitimate approach to ascertain the location of existing networks, the inability to localise a network via WiGLE does not automatically mean that the network does *not* exist: WiGLE does not map hidden networks, as these do not broadcast their SSID. As the main reason for using directed probe requests is to identify hidden networks, conversely, if the

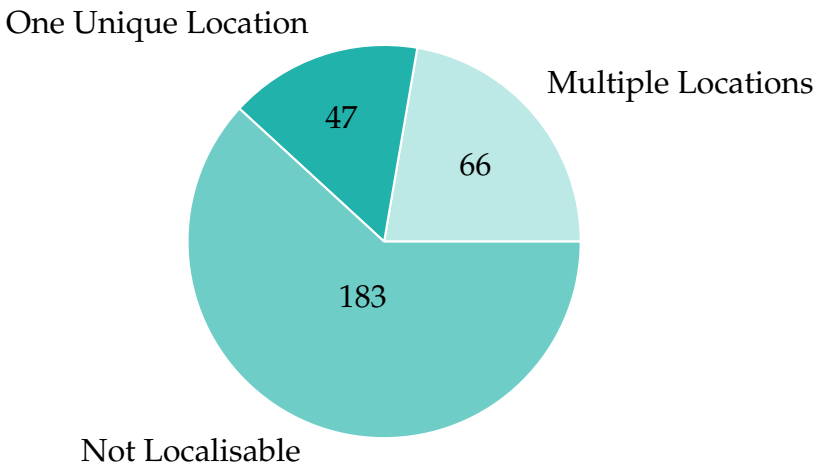


Figure 3.10: Resolving multiples of SSIDs using WiGLE: The potential typos identified in the evaluation were not localisable in 61.8 % of the cases. In 15.9 % of the cases, one location was resolved, and in 22.3 % of the cases, multiple locations were returned.

network is indeed an existing hidden network, WiGLE *cannot* localise it even though it exists. Nevertheless, a large percentage of the SSIDs recorded in the field study could be mapped to existing networks. One explanation might be that they do, in fact, exist, and that the real networks corresponding to the SSIDs in the recorded probe requests have been identified. However, other explanations are also possible:

i) While the SSID the device is probing for is, in fact, a hidden network not mapped by WiGLE, there might be one or multiple other networks with the identical SSID.

ii) The network has only been modified to be a hidden network recently, and the SSID shown in the map reflects the former existence of the non-hidden network and has not yet been updated.

iii) The network was added manually to WiGLE.

3.2.4 Device Tracking via Probe Requests

The privacy risks inherent to active discovery entail not only the transmission of SSIDs, and thereby possible disclosure of sensitive data: Over the years, various publications highlighted the possibility to track devices via probe requests. While the first attacks used the MAC address and SSIDs contained within probe requests, the introduction of MAC address randomisation also introduced the need for new attack vectors. This section compares the **attack vectors employed over time**, with

Authors	MAC Address	SSID	Supported Rates	Ext. Supported Rates	DS Parameter Set	HT Capabilities	VHT Capabilities	Extended Capabilities	Vendor Specific	all other fields	IE Field Sequence	BSS Membership	Transmission Frequency	Sequence Number	RSS
Pang et al. [Pan+07]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Cunche et al. [CKB14]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Freudiger et al. [Fre15]	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Vanhoef et al. [Van+16]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-
Robyns et al. [Rob+17]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	✓	-	-
Zhao et al. [Zha+19]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Dagelić et al. [DPČ19]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Gu et al [Gu+20]	-	-	✓	✓	✓	✓	✓	✓	-	✓	-	-	-	-	-
Uras et al. [Ura+20]	-	-	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-
Tan et al. [TC21]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	✓	✓
Pintor et al. [PA22]	-	-	-	-	-	✓	-	✓	✓	-	-	-	-	-	-
He et al. [HTC23]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓

Table 3.2: Comparison of the elements used for fingerprinting attacks across various publications. Adapted from [McD+24b].

particular regard for the separate fields of the IE that were considered distinguishing attributes. Parts of this section have previously been published in [McD+24b], and were extended for this thesis. An additional visual comparison is provided in Table 3.2, in which the elements used for fingerprinting throughout 12 influential publications are compared.

One of the first publications attempting to perform user tracking in 802.11 networks was published by Pang et al. [Pan+07]. They attempted to fingerprint devices using both probe requests as well as other wireless traffic. With respect to probe requests, they focussed on the **MAC address and SSID** contained in the request.

Cunche et al. [CKB14] used the **same set of identifiers** for fingerprinting, but additionally attempted to infer information on relationships between device owners by **comparing the PNLs**. The practical use of probe request tracking, particularly via the MAC address and SSID, was also demonstrated by Zhao et al. [Zha+19], who used probe requests to localise and track gang member via the **MAC addresses and SSIDs** their mobile devices transmit. Dagelić et al. [DPČ19] also demonstrated the possibility of tracking devices via MAC address and SSID, focussing less on localisation and more on analysing device behaviour during active discovery over a four-year period.

With the introduction of MAC address randomisation starting in 2014 (cf. Section 2.1.3) and the omission of the SSID field, other sources of information came into the focus of researchers analysing tracking via probe requests. Freudiger [Fre15] demonstrated that the **sequence number** can be used to track devices despite MAC address randomisation, as many devices randomise only their MAC address, but not the packet's sequence number. Alternative means of information were also used

by Vanhoef et al. [Van+16] to track devices via probe requests: they showed that the MAC address randomisation strategies were insufficient to inhibit tracking. For their attack, they used **all fields** transmitted in the probe request, and additionally the **sequence in which they were transmitted**: Vanhoef et al. found that the order of tags had a high entropy and therefore used it as an additional source of information.

For their fingerprinting technique, Robyns et al. [Rob+17] combined the **IE with the transmission frequency**. With respect to the content of the IE, they included all fields, but used a threshold λ that determines the part of the IE that is to be included. They determined that some fields have greater stability than others, and that even within fields, stability can vary on bit-level. By varying the threshold, they accommodated the variability with respect to the group size considered.

The attacks following in the subsequent years combined fingerprinting with other techniques: Gu et al. [Gu+20] used deep learning methods on **all fields except for the MAC address, the SSID, and vendor-specific values**. Tan et al. [TC21] model the probe requests in a flow network and use minimum-cost flow optimisation to track devices. Their approach entails **all fields, and additionally the sequence number and the received signal strength (RSS) field**. He et al. [HTC23] extend the work of Tan et al., and additionally employ even more identifiers like the **transmission order and the transmission time**. Uras et al. [Ura+20] and Pintor et al. [PA22] combine their device fingerprinting with **clustering approaches**, but while Pintor et al. chose **only a select few fields, including the HT Capabilities, the Extended Capabilities and Vendor Specific fields**, Uras et al. selected a considerably larger amount of fields, including **Extended Supported Rates, BSS Membership, HT Capabilities, Extended Capabilities, DS Parameters, VHT Capabilities and two additional reserved tags**.

While the introduction of MAC address randomisation and the omission of the SSID field has thwarted a lot of the early attacks, more recent publications introduce attacks that are still possible today: These mainly focus on other fields, and particularly the combination of fields and additional information that can be gleaned from probe requests. The following section introduces mitigations that aim to reduce the risk for individuals to be tracked.

3.3 Mitigations

The use of active discovery causes privacy risks to users: As described in the previous section, the SSIDs contained in probe requests can contain personal data. In case of misconfiguration, they might even contain passwords. To allow the use of hidden networks while simultaneously protecting the users from tracking via the PNL, a

hash-based scheme for covert SSID transmission is introduced in Section 3.3.2. While this scheme can protect users in case an AP is configured as a hidden network, the use of hidden networks is discouraged as they mainly pose a privacy risk via the required directed probe requests. It is therefore important to consider the current state of active discovery and question whether probe requests as they are today are in fact required.

In Section 3.3.3, **modified user settings for extended control** are proposed to enhance both Security-by-Design and Privacy-by-Design guarantees, while also providing users with greater control over their device's probing behaviour.

Section 3.3.4 concerns the question whether active discovery is in fact needed at all, or whether the use of **passive discovery only** would be sufficient. Subsequently, the use of a **generic MAC address** is introduced in Section 3.3.5. A generic MAC address defends against specific attacks targeting the MAC address: The lack of standardisation of MAC address randomisation drives manufacturers to implement randomisation as they see fit. This, in turn, can cause certain identifiers to still be contained in the MAC address, which then allows tracking despite the use of MAC address randomisation. A generic MAC address can help counter such attacks. Section 3.3.6 then introduces **generic probe requests**, in which the IE is reduced to the bare minimum. This way, probe requests lose their identifying features, allowing single devices to disappear in a large anonymity set.

3.3.1 Attacker Model

The considered attacker **passively monitors** the 2.4 and 5 GHz spectrum to record probe requests. They have an **unspecified amount of sensors** monitoring probe requests at their disposal, e.g. in a large area like a store, shopping mall or university campus. Since off-the-shelf equipment is sufficient for monitoring probe requests, the attacker is **not required to possess great financial means**. To perform real-time analysis, the sensors transmit the recorded probe requests to a server with **standard computational efficiency**, where all calculations are performed. By combining different fingerprinting techniques, the attacker is capable of tracking devices either using permanent identifiers such as the UAA or by calculating a fingerprint over elements contained in the IE tag. Simultaneously, the attacker has **bounded computational power**, and therefore a limited ability to find preimages of the SSIDs that the hashes introduced in Section 3.3.2 belong to.

3.3.2 Hash-Based Scheme for Covert SSID Transmission

The Wi-Fi Protected Access 3 (WPA3) standard (cf. Section 4.2.4) introduced the encryption of management frames through the 802.11w standard [Ebb20]. While probe requests *are* management frames, the encryption of management frames only applies to those exchanged *after* a 4-way handshake. Since probe request are transmitted *before* the 4-way handshake, the protection does not extend to them. Therefore, other means of protecting the content of probe requests, and particularly the SSID-field, are required: One such means is the omission of SSIDs in probe requests, as it reduces the attack surface probe requests provide. While some publications recommend to cease the use of probe requests altogether [Goo+19; Fra+06; WK18], a 2021 study [SRV21] revealed that the use of hidden networks is still prevalent, with up to 44 % of the networks found in certain areas being hidden networks. Discovering such hidden networks without active discovery is impossible, as they do not transmit beacons (cf. Section 2.1.2), and probe requests, and in particular *directed* probe requests, are therefore still required.

Given that the transmission of SSIDs in probe requests poses both a significant privacy leak of user data, and is crucial for connecting to hidden networks, a method for transmitting SSIDs in a non-trackable manner is needed. A hash-based scheme to conceal the transmitted SSIDs from attackers was therefore proposed in [McD+22], and is again presented here in the context of this thesis. The scheme preserves user privacy in a non-trackable way while simultaneously ensuring that a connection to hidden networks remains possible. The hash-based scheme is applied to the SSID contained within the probe request and concatenated ($||$) to the MAC address and the sequence number (SN) as a salt like so:

$$\mathsf{send}(\mathsf{hash}(\mathsf{MAC} \,||\, \mathsf{SN} \,||\, \mathsf{SSID}))$$

An AP receiving the probe request can then apply the sequence number and MAC address of the probe request to its own SSID and then compare the resulting hash to the hash stored in the probe request. If a salt were omitted, the mobile device could easily be tracked by its hashed SSID. The salt thus renders the resulting output less deterministic and, consequently, less trackable. Furthermore, by using the MAC address and sequence number, no additional salt has to be transmitted. The entropy introduced by this salt depends on the type of MAC address randomisation (cf. Section 2.1.3): Of the 48 bits used to construct a MAC address, two are typically fixed, as they contain the U/L bit and the I/G bit. For the remaining 46 bits, there are various randomisation strategies, some keeping the first 24 bits stable to denote the

OUI and only randomising the last 24 bits, while 46-bit randomisation randomises the whole address apart from the two aforementioned bits. By using the randomised MAC address as a salt for the SSID, the additional entropy is *at least* 24 bits, and *at most* 46 bits. The sequence number introduces an entropy of 12 bits. Together, both components of the salt introduce an entropy of a lower estimate of 36 bits, and a maximum entropy of 58 bits. To estimate the practical feasibility of the scheme, both the computational cost as well as the bandwidth requirements introduced by the scheme are evaluated in the following.

Computational Overhead To estimate the overhead the use of a hash function introduces to the comparison of two strings, a script was implemented. It uses the SHA-256 hash function from the `hashlib` library that Python provides. For baseline reference, a string comparison of two SSIDs is implemented in one function, and a hash-then-compare-function is implemented in another.

Both functions are tested in three test runs each, with each run calling the respective function a million times. On a Raspberry Pi, a device likely comparable in power to a home router but without cryptographic hardware acceleration, the hash-then-compare implementation increased the time required per comparison by 153.7 %, increasing the average time required for a single comparison from 4.6 microseconds to 11.7 microseconds. As the Raspberry Pi does not possess cryptographic hardware acceleration, which professional Wi-Fi routers on the other hand might have, the experiment was additionally run on a laptop that uses an Intel i5 processor. This reduced the overhead introduced by the hash-then-compare function to 53 %.

To put this into perspective, the amount of probe request captured in the field study performed in Section 3.2.3 can show the impact of hashed SSIDs on a real-world scenario: Over the course of the field study taking place in a busy pedestrian zone, around 23 probe requests were captured per second, 23.2 % of which contained an SSID, which is around 5.2 per second. A Raspberry Pi is capable of performing around 85 200 hashing operations every second. The overhead introduced by hashing the SSID before comparing it is therefore tolerable and well within the allocatable resources, also in even more frequented areas.

Bandwidth Overhead In terms of bandwidth overhead, an average probe request captured in the field study had a length of 133.3 bytes. A probe requests containing an SSID on the other hand, had an average length of 147 bytes, and the average length of the SSID was 11.4 bytes. Using the SHA-256 hashing algorithm, this average length would be increased to a fixed length of 32 byte, which corresponds to the maximum length of the SSID field. The SSID field length would therefore be increased by

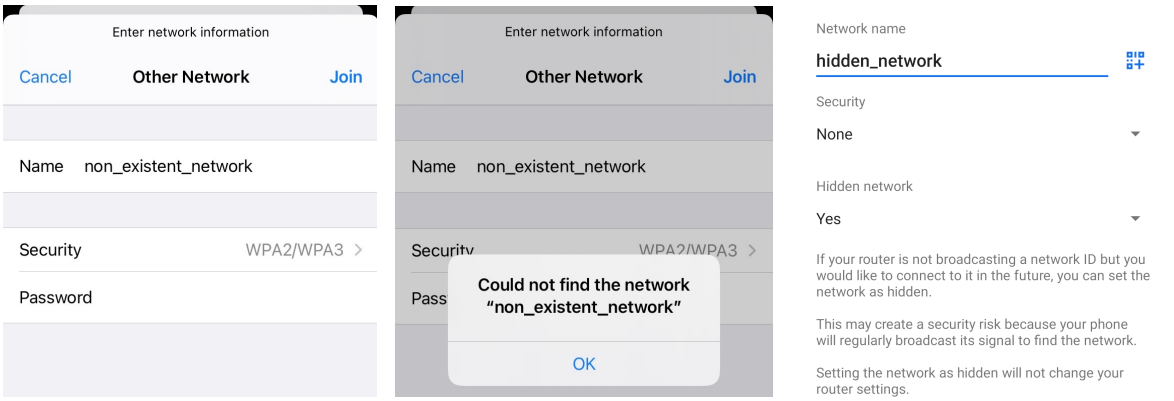
20.6 bytes, and the length of packets containing SSIDs would be increased to 167.6 bytes, which amounts to an increase of 14.07 %. Seeing that only 23.2 % of the probe requests captured contained an SSID, this trade-off can be considered acceptable with respect to the increase in privacy and the reduction of the ability to fingerprint individual devices via unchanging SSIDs.

A slight modification could result in a reduced bandwidth overhead: Instead of transmitting the whole 32 bytes of the hash, the string could be truncated to 16 bytes prior to submission. This way, directed probe requests would instead have an average packet length of 151.6 bytes, which is an increase of only 3.2 % compared to non-hashed SSID transmission. This increases the likelihood of hash collisions, with the consequence that there is a slightly higher likelihood to receive a probe response from a device whose hashed SSID resulted in hash with an overlapping first 16 bytes.

An additional reduction of overhead could be achieved by **limiting the devices** that respond to directed probe requests containing hashed SSIDs **to hidden networks only**.

Limitations This scheme holds against an attacker capable of monitoring all probe requests, but with **bounded computational power**, which makes the computation of the observed preimages of the hashes, namely the SSIDs, impractical. This holds provided that both MAC addresses and sequence numbers are randomly chosen and, together, provide a lower estimate of 36 bits of entropy. The scheme can defend users from attackers that want to learn the SSIDs contained in a user's PNL, possibly to locate them via WiGLE, and it also protects users from being tracked via an SSID unknown to the attacker. However, **it does not hold against an attacker with an a priori list of known SSIDs** who computes the hashes on the fly, made up of the current MAC address, sequence number and the known SSID(s), and compare said hashes with the hashes monitored in the vicinity.

This section showcased the hash-based SSID scheme as a privacy enhancing technique to reduce individual device fingerprints while simultaneously allowing privacy friendly reconnaissance with hidden networks. The subsequent section introduces additional control mechanisms that could be implemented to improve device privacy and provide users with more control over the probing behaviour of their mobile device.



(a) Screenshots of an attempt to connect to a hidden network on iOS 15. While Android devices allow to insert entries into the PNL even when the corresponding network is out of reach, iPhones only allow a manual extension of the PNL when a connection to the network can be established.

(b) A warning message displayed on devices running Android 9 or newer upon manually inserting a hidden network.

Figure 3.11: Examples of user dialogues to mitigate unwanted SSID entry.

3.3.3 User Interface Improvements to Reduce Privacy Risks

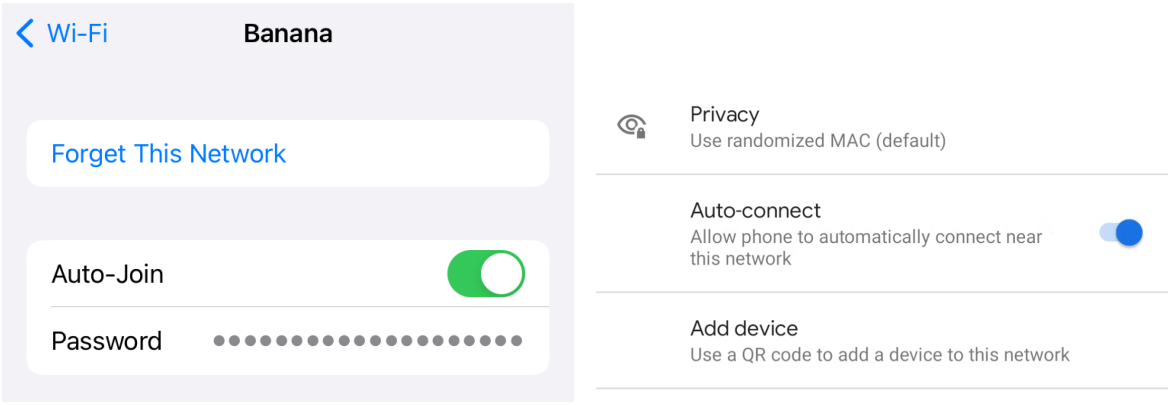
While the previously introduced approach permits privacy friendly directed probe requests, it is a solely protocol-based approach. Introducing it requires for both the manufacturers of routers as well as mobile device manufacturers to implement the protocol. Since the problematic SSIDs discovered in probe requests in the field study introduced in Section 3.2.3.2 are hypothesised to be primarily caused by users accidentally inserting SSIDs or passwords and not deleting them subsequently, modifications in the user interface (UI) design of mobile devices, including approaches like Privacy-by-Design and Security-by-Design, could improve device privacy even more, while simultaneously increasing user control. The herein introduced suggestions were published in [McD+22]. They encompass safeguards for manual SSID entry, the removal of known SSIDs, default expiry dates for PNL entries, manually adjustable auto-join functions and the possibility to silence probe requests and instead rely on passive discovery only. Since some of these suggestions are part of Android or iOS devices, the following paragraphs first outline the existing safeguards and then formalise the improvements that should be considered for Android, respective iOS devices.

SSID Entry Safeguards Both the recent iOS and Android OSs feature safeguards to prevent users from accidentally inserting hidden networks into their PNL, and

therefore introducing a network that would be used for subsequent directed probing: As can be seen in Fig. 3.11a, iOS devices **only allow the insertion of existing networks within range**, to which a connection can be established at the time of insertion. In case the entered password was incorrect or the network can not be discovered in the vicinity despite the use of directed probe requests, an acknowledgement of the warning message will lead users back to the general Wi-Fi setup page, and will not result in the network being added to the PNL. Whether a device is a hidden network, and will therefore be probed for using directed probe requests, or a non-hidden network, is determined automatically by iOS devices when first establishing a connection to the network. On Android devices, on the other hand, it is **possible to insert networks that are out of range**. Fig. 3.11b shows that an attempt to set the network status as hidden upon adding it displays a warning message. In it, the user is provided with explanations about the use of hidden networks, and is cautioned about the privacy risks introduced by the regular broadcast of directed probe requests. While **manually setting a network to hidden** could be considered unnecessary considering that iOS devices are capable of detecting their hidden status, it is required in case the added network is not within reach, and its hidden status can not be determined on the fly. To improve the security of Android devices to match that of iOS devices with respect to the manual insertion of (unwanted) hidden networks, the recommendation is for Android devices to adapt the iOS strategy of **adding only networks within reach**, and determining the hidden status automatically instead of requiring users their users to do so.

Known SSID Removal In the beginning of 2022, at the time of the publication of [McD+22], only Android allowed the **modification of the PNL via the network settings**, while iOS solely allowed the **modification of networks within range** of the mobile device (cf. Section 3.2.1). This changed in iOS 16: Starting with its release in September 2022, the modification of all PNL entries both within and not within reach was possible [Wik24b]. Albeit both Android as well as iOS devices now fulfil the recommendation published in [McD+22] to modify iOS to allow for the modification of distant networks, this safeguard is still mentioned here, for the sake of completion. It is unclear whether or not [McD+22] had any influence on the decision to include the feature in iOS, since despite a thorough search, it was impossible to obtain information or release notes introducing the feature from the official Apple website; only a manual on how to use it, published in October 2023 [Sup23].

PNL Entry Expiry An additional recommendation encompasses **expiry dates for entries of the PNL**: Whether a user visits a café in a different city once, stays in a holiday home for a week, attends a regular weekly event or streams series at home,



(a) The auto-join setting in iOS 17.

(b) The auto-join setting in Android 11, here called auto-connect. Image source: [Dav20].

Figure 3.12: Comparison of the auto-join functions in Android and iOS. Both OS allow to select the auto-join function on a per-network basis.

their needs to connect to the available networks vary. While the option to modify the PNL after use allows purging unused networks, a **switch to set an expiry date directly when connecting to a network** would alleviate the PNL sanitisation: A user connecting to a café network in a place they are unlikely to visit again could set for the network to be forgotten on the next day. A visitor of a holiday home could set the expiration date to a week, and the default expiration date that would also apply to the home network of a person or a regular event they're attending would be *never*. This **reduces the likelihood of attack using fake APs**, where attackers set up APs with common SSIDs to trick devices into connecting to them. The goal of such an attack could be to identify devices via their MAC address, since either the LAA or the UAA is disclosed upon connection establishment, or to capture their traffic. A strategy such as a predefined PNL entry expiry would assist in PNL sanitisation with minimal additional effort required from the user. Neither Android nor iOS devices allow users to set an expiration date for added networks.

Auto-Join Prompt While users would likely not remove irregularly used networks from their PNL to forgo the effort of adding them again, they might instead profit from an adjustable auto-join function. Especially for well-used public networks, the risk of falling victim to a fake AP attack is significantly reduced if users are prompted to decide whether they want to automatically join the network or instead manually choose the connection whenever they in physical proximity of the legitimate network. While both Android and iOS devices allow to disable auto-join per network, as can

also be seen in Figures 3.12, a **prompt upon first connection establishment** would increase the visibility of the option and thereby increase its privacy gain.

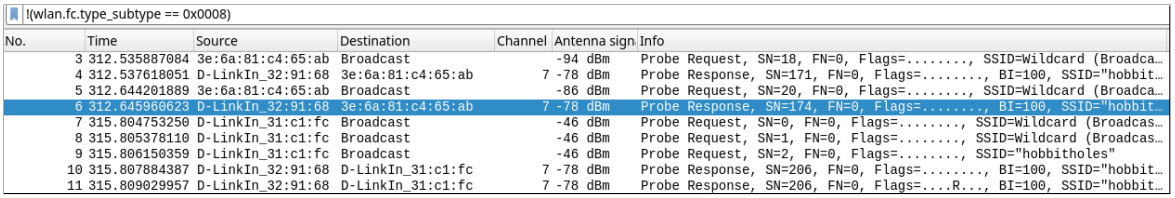
Probe Request Silencing The primary reason for the transmission of probe requests is to identify nearby known networks. As information on nearby networks not only serves to improve network connectivity, but also allows for coarse localisation of devices via network maps, turning off the Wi-Fi does not necessarily mean that no probe requests are used, as localisation processes can transmit probe requests even with Wi-Fi deactivated [MCT17]. While additional settings can allow users to prevent devices from using probe requests to localise the device, these are difficult to find. This could be remedied via a **reduced visibility mode**: To provide increased user controls and offer the possibility to remain undetectable via probe request, regardless of their content, the possibility to rely on passive discovery only would meet increased privacy demands that some users might have. This comes with **several trade-offs** that users have to be aware of:

- Connection establishment via passive discovery is **slightly slower** than active discovery [WK18].
- Passive discovery **increases the battery usage** [Fre15].
- **Connecting to hidden networks is impossible** via passive discovery, as hidden networks do not transmit beacons.

Nevertheless, some users might choose to accept those trade-offs knowing that the privacy gain is very large. Albeit the sole reliance on passive discovery is only a legitimate option to improve user privacy when using non-hidden networks, this option, as well as improvements to passive discovery discussed in the literature are investigated in more detail in the following section.

3.3.4 Beacons Only

While active discovery allows mobile devices to efficiently and quickly locate nearby known APs, it also entails inherent privacy risks, as shown in Section 3.2. Instead of trying to improve such an insecure scheme, another idea is to completely eliminate it. While probe requests are actually necessary to connect to hidden networks, all non-hidden networks could just be discovered using passive scanning. This approach has previously been suggested by Franklin et al. [Fra+06] in 2006, Waltari et al. [WK18] in 2018 and Goovaerts et al. [Goo+19] in 2019. The latter two go beyond just suggesting the idea: They analyse the temporal gain achieved by the use of active scanning



!(wlan.fc.type_subtype == 0x0008)

No.	Time	Source	Destination	Channel	Antenna sign.	Info
3	312.535887084	3e:6a:81:c4:65:ab	Broadcast		-94 dBm	Probe Request, SN=18, FN=0, Flags=........, SSID=Wildcard (Broadca…
4	312.537618051	D-LinkIn_32:91:68	3e:6a:81:c4:65:ab	7	-78 dBm	Probe Response, SN=171, FN=0, Flags=........, BI=100, SSID="hobbit…
5	312.644201889	3e:6a:81:c4:65:ab	Broadcast		-86 dBm	Probe Request, SN=20, FN=0, Flags=........, SSID=Wildcard (Broadca…
6	312.645960623	D-LinkIn_32:91:68	3e:6a:81:c4:65:ab	7	-78 dBm	Probe Response, SN=174, FN=0, Flags=........, BI=100, SSID="hobbit…
7	315.804753250	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=0, FN=0, Flags=........, SSID=Wildcard (Broadcas…
8	315.805378110	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=1, FN=0, Flags=........, SSID=Wildcard (Broadcas…
9	315.806150359	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=2, FN=0, Flags=........, SSID="hobbitholes"
10	315.807884387	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc	7	-78 dBm	Probe Response, SN=206, FN=0, Flags=........, BI=100, SSID="hobbit…
11	315.809029957	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc	7	-78 dBm	Probe Response, SN=206, FN=0, Flags=....R..., BI=100, SSID="hobbit…

Figure 3.13: A screenshot of probe requests and their responses captured in Wireshark to illustrate the response times in active discovery.

versus passive scanning, and develop methods to improve passive scanning even more.

During active discovery, a probe response is expected to be received immediately after the probe request was sent. Fig. 3.13, for example, shows two probe requests and their probe responses in packets 3 to 6. In these two instances, the probe response is received 1.73 and 1.76 ms after the probe request is transmitted. It is trivial to see that passive discovery, where every channel has to be monitored for beacons which are only sent every 100 to 120 ms [WK18], is slower than active discovery.

Waltari et al. [WK18] quantify this timing difference: They discover that when the beacon interval is set to 100 ms, passive network discovery is around 0.6 seconds slower than active discovery in 98 % of the cases. This behaviour can also be observed in Fig. 3.14.

In order to enhance passive network discovery and reduce the time difference between active and passive discovery, Goovaerts et al. [Goo+19] suggest several techniques to improve passive scanning: Their solution combines the following approaches:

(a) a prolonged *dwell time* while monitoring for beacons,

(b) prioritised scans and

(c) APs advertising their neighbouring SSIDs.

The **dwell time** denotes the time a device listens for beacons on a specific channel. A high dwell time ensures that devices can receive beacons from all surrounding APs on the first scan. The authors found out that a dwell time of 100 ms discovers around 90 % of the nearby APs, while a **dwell time of 120 ms or more returns 100 %**. In addition, instead of scanning every single channel, they propose to **prioritise the scans** to first monitor the channels 1, 6 and 11 in the 2.4 GHz spectrum, and in the next round channels 36, 40 and 44 in the 5 GHz spectrum. Additionally, they dynamically enhance this prioritised search by **always starting with the channel the last connection had been established on**, assuming this is the default channel for the network. These techniques deliver considerably improved results when measuring

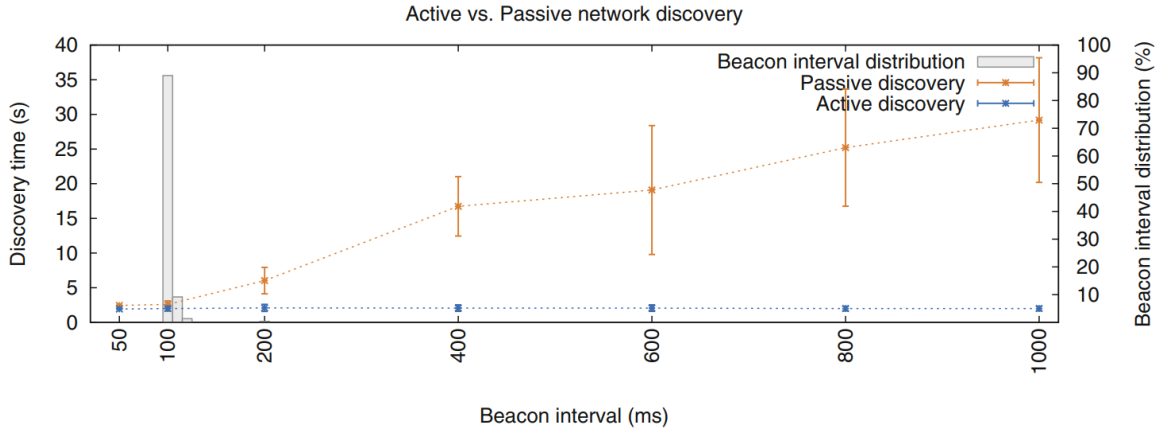
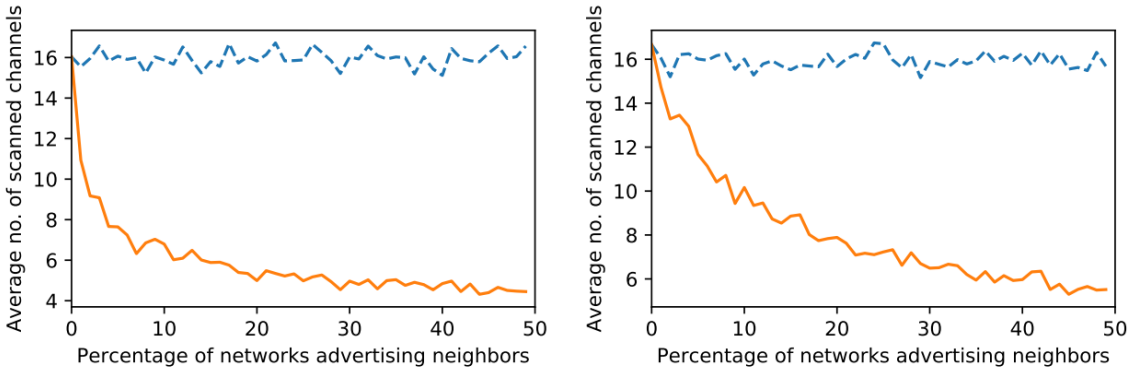


Figure 3.14: Network discovery using active and passive scanning, with discovery times measured with respect to beacon intervals. The right-most axis additionally shows the beacon interval distribution, which shows that around 90 % of the APs send beacons every 100 ms, while a few use slightly longer intervals. Image source: [WK18].

the time between the initial scan to the discovery of a known network: When the same dynamic priority scan is performed in both active, as well as passive discovery, active discovery is sped up from around 3.8 seconds to 0.049 seconds. Using a dwell time of 100 and 120 ms, dynamic prioritised passive scanning ranges around 0.91, respective 0.13 seconds.

Goovaerts et al. additionally propose that APs should **advertise neighbouring APs**. This reduces the amount of channels that have to be scanned before a known AP is discovered, and thereby accelerates connection establishment even more: In a typical search, an average of 16 channels is scanned; however, as the percentage of access points advertising neighbouring APs approaches 50 %, the number of scanned channels decreases to around 5. If in densely populated areas, around 12 % of the APs advertise their neighbours, mobile devices only have to scan 6 channels to discover a known network. In less densely populated areas, the same amount of channels is reached around 35 % of APs advertising neighbours. This behaviour can also be observed in Fig. 3.15.

While the last improvement requires changes in the implementation of routers, and is therefore less likely to be implemented, the first experiments on passive scanning showed that particularly a combination of a dwell time of 120 ms and prioritised dynamic scanning can improve passive network discovery to be faster, more efficient and more privacy friendly than the currently used implementations of active discovery. In summary, the results are as follows:



(a) San Francisco, California.

(b) Boulder, Colorado.

Figure 3.15: The average amount of channels that has to be scanned to discover a known network. The dashed blue line describes normal scanning, while the solid orange line denotes neighbour advertising. Figure (a) represents a densely populated area, and Figure (b) a sparsely populated one. Image source: [Goo+19].

Result: Passive discovery is more privacy friendly than active discovery, and by using prioritised dynamic scanning, it can be improved to be more efficient than currently used implementations of active discovery. However, active discovery remains the only option of discovering hidden networks.

3.3.5 Constant and Globally Equal MAC Address for Probe Requests

The problems of MAC address randomisation have been described in various publications. They range from lacking standardisation [ZBA24] to insufficient randomisation resulting in the possibility to associate frames [Van+16; Mar+17; Fen+21]. While it appears like a random string of bytes, the MAC address can still contain a variety of information: As described in Section 2.1.1, the first three bytes of the MAC address contain the OUI, as well as the U/L bit and the I/G bit. Martin et al. [Mar+17] and Fenske et al. [Fen+21] show that some implementations of MAC address randomisation are flawed, since they omit randomising the OUI, and can therefore still leak information on the devices. This way, devices can be identified and tracked despite the use of MAC address randomisation. This knowledge also characterises the **attacker model**: The use of a generic MAC address can protect users from an attacker who infers information on devices only by observing the MAC address. The trivial

```
1  from scapy.all import *
2
3  ssid = ""
4  interface = "wlan0"
5  sender = "22:22:22:22:22:22"
6  dest = "ff:ff:ff:ff:ff:ff"
7
8
9  def send_probe_req(senderaddr, destaddr, ssid, interface):
10         radiotap = RadioTap()
11         dot11 = Dot11(type=0,
12             subtype=0x04,
13             addr1=destaddr,
14             addr2=senderaddr,
15             addr3=destaddr)
16
17         rates_content = b'\x82\x84\x8b\x96'
18         ratestag  = Dot11Elt(ID='Rates', info=rates_content)
19         ssidtag = Dot11Elt(ID="SSID", info=ssid)
20
21         frame = radiotap / dot11 / Dot11ProbeReq() / ssidtag /
               ratestag
22
23         sendp(frame, iface=interface)
24
25 send_probe_req(sender, dest, ssid, interface)
```

Listing 3.1: The scapy script used to transmit probe requests via a USB antenna. The only IE tags contained are Supported Rates and the SSID, defined in lines 3 and 17 - 19. Adapted from [McD+24b].

case is a device using its UAA when sending probe requests; the more common case nowadays, with most modern devices using MAC address randomisation (cf. Section 3.2.1) are devices that utilize flawed randomisation schemes (cf. Section 2.1.3). The information on the device, which can be obtained using low-cost receivers like Wi-Fi-USB-antennae or an ESP32, can be used by the attacker as a device fingerprint with which they can track the device over time.

A solution that inhibits correlating frames due to flawed randomisation implementations was published in [McD+24a], the results of which are presented here in the context of this thesis. The paper proposed the use of a generic address during network discovery: Here, one generic address is used for all probing devices, and the device-specific MAC address is only used once the connection establishment with an AP is initiated. The subsequent switch back to the UAA/LAA is indispensable to avoid MAC address collisions during connection establishment and data transfer.

```
1 [device-wlan0]
2     wifi.scan-generate-mac-address-mask = FF:FF:FF:FF:FF:FF
          22:22:22:22:22:22
```

Listing 3.2: NetworkManager configuration to enable probing via the generic address 22:22:22:22:22:22. Listing Source: [McD+24b].

To first answer the question whether the use of a generic address still allows the transmitting devices to receive probe responses, the script shown in Listing 3.1 is used to transmit probe requests and record the resulting probe responses. It uses Scapy[3], an interactive packet manipulation program for Python. As can be seen in line 5, the sender MAC address is set to `22:22:22:22:22:22`. This is both a universally available and unreserved MAC address, and therefore suitable for the task [Aut23]. Lines 18 and 19 of the script define the IE content, namely the Supported Rates and the SSID tag. Line 21 combines the parts necessary to form a probe request to construct the packet in the Scapy synthax. Line 23 calls the send function that initiates frame transmission. In line 25, the function is called using the parameters set in the preamble, in lines 3-6. The probe requests are transmitted via a USB antenna, and the traffic simultaneously monitored using two USB antennae on channels 1 and 6, the primarily used channels in the vicinity of the experimental setup. The results are encouraging: Regardless of how many requests are sent from one MAC address, they always receive responses. It appears therefore that there is no mechanism to store previously seen MAC addresses. Hence, the proposed idea to use a generic address can subsequently be implemented and tested to verify its usability.

3.3.5.1 Implementation

The generic address is implemented on a Raspberry Pi running the linux-based operating system Raspberry Pi OS[4]. Using NetworkManager[5], a network configuration tool suite that also facilitates the use of MAC address randomisation, modifications to the address during network discovery can be implemented. The modifications are applied in the file `/etc/NetworkManager/conf.d/generate-mac-address.conf`, in which the setting `wifi.scan-generate-mac-address-mask` has to be included, as can be seen in line 2 of Listing 3.2. The setting modifies the MAC address only while probing, and switches to the individual MAC address for connection establishment. It determines a mask of bits to be replaced by a second value. The mask is set to

3. https://scapy.net/
4. https://www.raspberrypi.com/software/
5. https://networkmanager.dev/

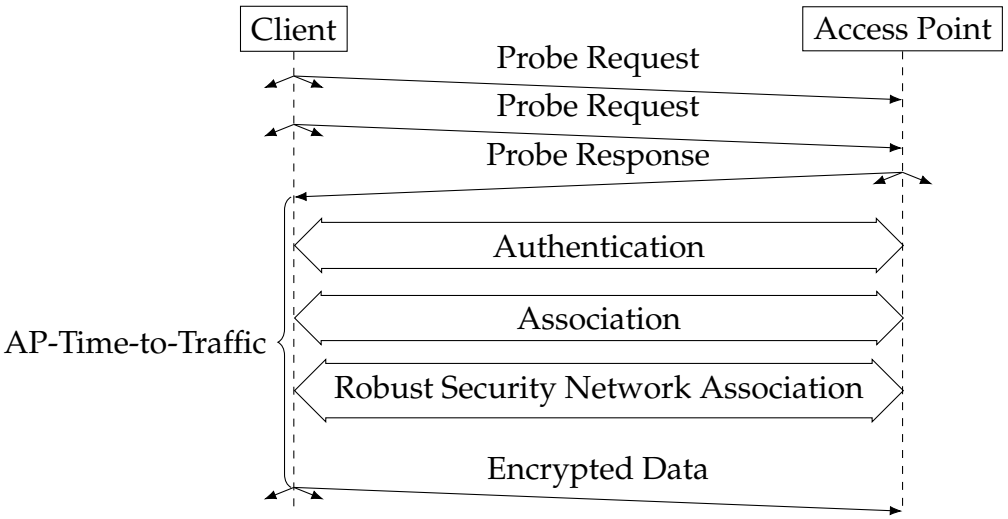


Figure 3.16: IEEE 802.11 connection establishment including the AP-Time-to-Traffic used in Section 3.3.5. Adapted from [McD+24a].

`FF:FF:FF:FF:FF:FF`, because of which every single bit is switched. The value used instead is the MAC address `22:22:22:22:22:22`, as in the previous example. MAC address randomisation, on the other hand, can be switched on in NetworkManager via the `wifi.scan-rand-mac-address-mask` configuration.

3.3.5.2 AP-Time-to-Traffic Metric

To measure the duration of connection establishment with respect to the use of a generic address, the AP-Time-to-Traffic (AP-TtT) metric is introduced. It measures the responsiveness of the AP with respect to the use of covert MAC addresses, or lack thereof. The establishment of a connection begins with the first transmission of probe requests by the client, and concludes with data transfer between the client and the access point (cf. Fig. 3.16). For the sake of reproducibility, instead of choosing the initial probe request as the starting point, the first probe response from the access point serves as the starting point to ensure server availability; otherwise, the waiting time would be included in the connection establishment time, which is undesirable. The AP-TtT concludes with the transmission of the first data frame.

3.3.5.3 Scalability Analysis

The above mentioned configuration of NetworkManager is tested using five Raspberry Pis. This is necessary to analyse the behaviour upon intentionally introducing MAC address collisions: Normally, nodes affected by MAC address collisions can

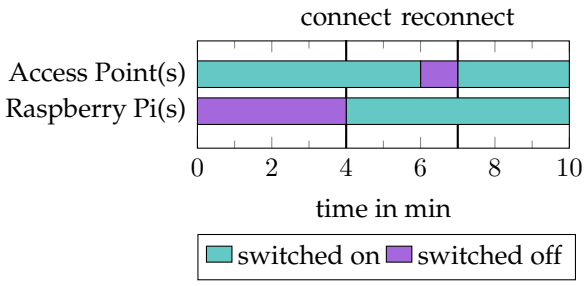


Figure 3.17: A test run produces two data points: The Raspberry Pis are powered on initially after 4 minutes, marking the recording of the first connection attempt with the AP. The AP is subsequently powered off after 6 minutes and switched on again after 7 minutes, where a second connection attempt is recorded. Adapted from [McD+24b].

no longer be distinguished on the data link layer; consequently, a frame intended for a single destination will also be received by nodes with colliding addresses. MAC address collisions are therefore undesirable within a network and during a connection; during network discovery, on the other hand, it is unnecessary for the AP to be able to differentiate between two devices sending probe requests, since it responds to every probe request with a probe response, regardless of the MAC address it was sent from. The test setup therefore includes five Raspberry Pis with the same modifications implemented in NetworkManager. To test whether connection establishment is impeded in case of more than one AP responding to the generic probe requests, two APs with distinct SSIDs are used: The APs are implemented via a Wi-Fi antenna connected to a laptop. The interface is split via hostapd[6], a user space daemon to set up access points, and a network bridge to provide two access points.

Three of the Raspberry Pis are equipped with the credentials for the first access point, and the other two have those of the second access point. The generic address is then tested in several test runs according to Fig. 3.17: The access points are switched on at minute zero, and the Raspberry Pis at minute four. A reconnection is recorded after the access points are switched off at minute six and turned on again at minute seven. With every connection attempt, the AP-TtT (cf. Section 3.3.5.2) is measured, resulting in two data points per test run.

Fig. 3.18 shows the results of these tests: The average AP-TtT with two devices is 9.31 seconds across 14 test runs, encompassing 53 data points. Three devices averaged an AP-TtT of 9.61 seconds across 12 test runs, encompassing 72 data points. Four devices had an average client-TtT of 9.63 across 5 test runs, encompassing 38 data points, and five devices an average of 9.77 across 5 test runs, encompassing 50 data

6. https://w1.fi/hostapd/

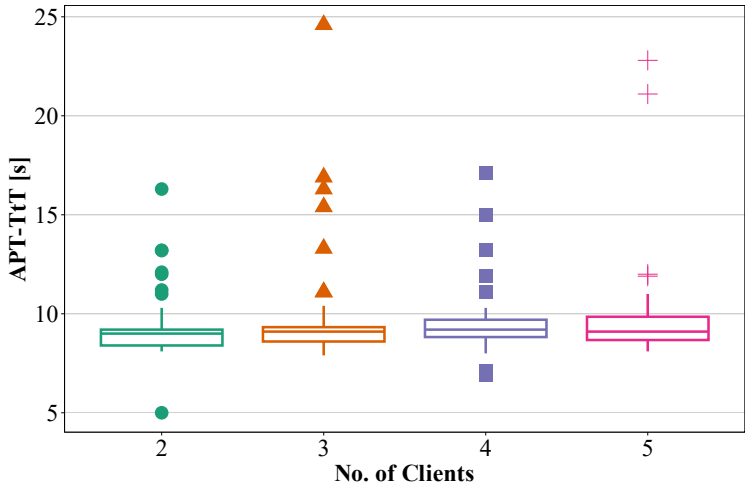


Figure 3.18: Comparative measurements of two to five devices employing a generic address and the respective AP-Time-to-Traffic. Image source: [McD+24b].

points. In a small number of instances, the obtained captures lacked specific essential measurement points for AP-TtT. In these cases, the required measurement points were either not recorded or not sent, which is why the number of data points is not necessarily a multiple of the possible number of connection establishments. Nevertheless, in every test run, the specified number of devices included in this comparison successfully established a connection.

While a slightly detectable increase of the AP-TtT can be recorded, the AP-TtT is rather comparable for two to five devices. Since this experiment serves to show that the use of a generic address is possible for multiple devices at the same time, it requires a large-scale simulation to establish whether the AP-TtT remains stable with a higher amount of devices, and whether the slight increase reflects a varying number of data points or an actual deceleration. While this would be required before actually implementing the scheme in devices, such a large-scale experiment is out of scope for this thesis.

3.3.5.4 Comparative Evaluation

To classify the use of a generic address in comparison to other conventional schemes, the AP-TtT is measured in four different settings:

(i) A Raspberry Pi using a generic address while probing,

(ii) one using MAC address randomisation,

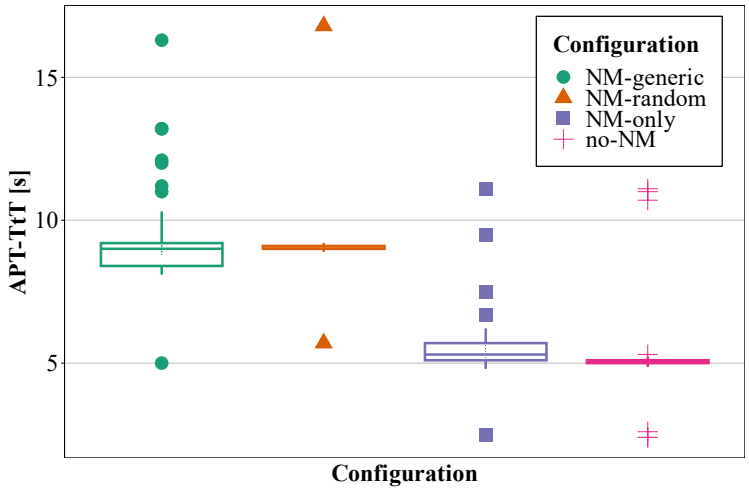


Figure 3.19: The AP-Time-to-Traffic required for connection establishment in four settings: without the use of NetworkManager (no-NM), using NetworkManager (NM-only), using a generic address (NM-generic) and using MAC address randomisation (NM-random). Image source: [McD+24b].

(iii) one connecting while using its built-in network configurator and

(iv) one connecting while using NetworkManager without additional configurations.

The results can also be observed in Fig. 3.19 and are clarified in the following:

Connection establishment using a generic address: The average AP-TtT required when using a generic address was 9.31 seconds. The time required for an increasing amount of connecting devices remains fairly stable (cf. Section 3.3.5.3).

Connection establishment using MAC address randomisation: To compare the use of a generic address to a scheme offering similar privacy protection, connection establishment while using MAC address randomisation is measured. Devices probing while using randomised MAC addresses require 9.27 seconds AP-TtT when establishing a connection.

Connection establishment without NetworkManager: To determine how long a connection attempt without the use of privacy enhancing tools takes, the connection establishment without the use of NetworkManager is measured. In this setting, the average AP-TtT is 5.36 seconds.

Connection establishment using NetworkManager without additional configuration: As the previous experiments showed that the use of privacy enhancing tools takes significantly longer than that without, this experiment determines whether the increased time is due to the use of NetworkManager or due to the additional configurations. Therefore, connection establishment while using NetworkManager without additional configuration is measured. The AP-TtT in this setting is 5.59 seconds on average.

The test results show that while the use of privacy enhancing techniques significantly slows down connection establishment, both privacy enhancing schemes cause a comparable AP-TtT of 9.27 vs. 9.31 seconds. The third experiment showed that connecting without privacy enhancing schemes and without NetworkManager caused an AP-TtT of 5.36 seconds. The last experiment revealed that the overhead is not caused by NetworkManager, as connection establishment using NetworkManager without additional configurations is comparable to that without NetworkManager at 5.59 seconds.

While both the use of MAC address randomisation as well as the use of a generic address cause a significant but comparable deceleration of connection establishment, they provide a significant improvement in privacy. Optimised implementations can likely reduce this overhead, and while MAC address randomisation is an already-established and pervasive scheme, the use of a generic address enhances privacy more, since no information can be inferred from inadequately implemented randomisation schemes. It additionally allows devices to disappear in a very large anonymity set, and should therefore be considered a replacement for MAC address randomisation.

3.3.5.5 Discussion

The proof-of-concepts introduced in this section demonstrated that it is possible for multiple devices to share the same MAC address while probing. A generic address does **not create a major overhead** in comparison to an established scheme like MAC address randomisation. Additionally, a scalability analysis showed that the connection establishment **took comparably long** between 2 and 5 devices. Simultaneously, the tests conducted in this experiment still leave some open-end questions, e.g. whether the AP-TtT is stable in a large-scale simulation or real-world scenario, and whether the minute tendencies observed in Section 3.3.5.3 were caused by the varying amount of data points.

Since the generic MAC address is only used while probing and the UAA/LAA used for subsequent connection establishment, there should be no obstacles in using it

on a larger scale. If implemented in the device driver, the overhead measured in Section 3.3.5.4 could be reduced, too.

With respect to the **security properties** the use of a generic MAC would introduce, it would ensure firstly that **no information on the manufacturer of the device could be leaked** via a persistent OUI in case of the use of 24-bit randomisation (cf. Section 2.1.3). It would additionally **hinder the distinction of bursts** transmitted from two different devices using the same MAC address for probing. The less distinguishing information is additionally contained within the IE (cf. Section 3.3.6), the harder it is to differentiate between all transmissions, both from different devices as well as within bursts. In summary, a generic MAC address would **distinctly improve the security in comparison to 24-bit randomisation**, and since the privacy protecting properties exceed those of MAC address randomisation, it should be considered a suitable replacement.

3.3.6 Generic Probe Requests

The use of a generic MAC address, as introduced in the previous section, can defend against attackers inferring information via the OUI given 24-bit randomisation, or flawed implementations of MAC address randomisation. This is not the only vantage from which probe requests are attacked: A much discussed and implemented attack vector is to fingerprint the IE tags. The particular fields often contained within the IE are introduced in Section 2.1.2, as well as the additional discriminating attributes used for tracking compared in Section 3.2.4 and visualised in Table 3.2. The latter also shows that the most distinguishing features used for attacks are contained within the IE of the probe requests: Some publications additionally use the sequence in which the tags are transmitted, the transmission frequency, sequence number or the received signal strength (RSS), but all recent research focusses on the IE as the main element, with the additional features only serving as a secondary distinguishing attribute. The logical consequence to reduce this attack surface is to **propose a minimisation of IE content**.

This has previously been proposed in various publications [Van+16; Mar+17; Fen+21], but its implications have never been scientifically analysed in detail. At least one manufacturer, Google, has started to implement minimised IE content [Hog17]: According to their development blog, the IE content transmitted by Pixel, Pixel XL and Nexus 5x devices running Android O+ is limited to transmit only the SSID and the DS parameter set (cf. Section 2.1.2). To test such a minimised IE in the context of this thesis, this behaviour is simulated using Scapy. The Scapy script shown in Listing 3.3 is used to transmit probe requests via an external USB antenna. Simultaneously, the non-overlapping channels 1 and 6 are monitored via two more USB antennae; in

```
1  from scapy.all import *
2
3  ssid = ""
4  interface = "wlan0"
5  sender = "22:22:22:22:22:22"
6  dest = "ff:ff:ff:ff:ff:ff"
7
8
9  def send_probe_req(senderaddr, destaddr, ssid, interface):
10         radiotap = RadioTap()
11         dot11 = Dot11(type=0,
12             subtype=0x04,
13             addr1=destaddr,
14             addr2=senderaddr,
15             addr3=destaddr)
16
17         dsset = Dot11Elt(ID='DSset',info='\x01')
18         ssidtag = Dot11Elt(ID="SSID", info=ssid)
19
20         frame = radiotap / dot11 / Dot11ProbeReq() / ssidtag /
               dsset
21
22         sendp(frame, iface=interface)
23
24 send_probe_req(sender, dest, ssid, interface)
```

Listing 3.3: The Scapy script used to emulate probe requests transmitted by Google devices running Android O+ and higher. Note the content of the IE to be transmitted: In line 17 and 18, the DS parameter set and the empty SSID tag are defined. They are subsequently joined to construct a packet in the Scapy synthax in line 20.

the vicinity of the experimental setup, channel 11 is used only infrequently, and therefore omitted in the setup.

The result of this experiment showed, that while several routers in the vicinity of the experimental setup responded with probe responses, a very commonly used router in Germany, **the FRITZ!Box 7490, failed to respond to probe requests containing only the SSID and the DS parameter set**. Despite using the standard configuration of this widely-spread router, it appeared to discard probe requests containing only the SSID and DS parameter set tags and failed to respond to them. Reducing the content to SSID and DS parameter set therefore seems to be a solution that is not universally applicable.

In the next test, the goal is to estimate how far the content of probe requests can be reduced to produce a smaller attack surface, while still receiving probe requests

```
1  rates_content = b'\x82\x84\x8b\x96'
2  ratestag  = Dot11Elt(ID='Rates', info=rates_content)
3  ssidtag = Dot11Elt(ID="SSID", info=ssid)
4
5  frame = radiotap / dot11 / Dot11ProbeReq() / ssidtag /
       ratestag
```

Listing 3.4: The part of the Scapy script required to set Supported Rates and SSID as the content of the IE. Line 1 defines the content of the Supported Rates, lines 2 and 3 convert them to the format required to construct a package in the Scapy synthax. The packet itself is constructed in Line 5.

from all routers in the vicinity of the experiment. Therefore, captured probe requests are re-transmitted and gradually reduced in content. The results of this experiment are introduced in the following, and were also published in [McD+24b].

This experimental setup provides the following observation:

> **Observation:** By retransmitting captured probe requests and reducing their IE content step by step, it can be deduced that the only fields required for probe requests to receive probe responses are the SSID field and Supported Rates (cf. Section 2.1.2). Both fields *can* be empty, but *have* to be included in the request.

From the gradually reduced content of the re-transmitted probe requests, another python script is created. The structure is the same as can be seen in Listing 3.3, but with lines 17-20 replaced by the the content shown in Listing 3.4. The resulting script is used to robustly determine and illustrate the necessary frame content: **The bare minimum required to receive probe responses of all surrounding networks is to include the SSID and Supported Rates tag into the IE.**

3.3.6.1 Proposition: IE Minimisation

The previously introduced results form the basis of the proposal of this section: the reduction of the IE content to the bare minimum. This can be done by the use of **undirected generic probe requests, containing a reduced IE field with only the SSID and Supported Rates tags**. This way, user privacy can be increased while simultaneously ensuring that the functionality of active discovery is maintained. The proposal additionally includes a limitation to *undirected* probe requests: These encompass the majority of the transmitted requests. *Undirected* probe requests are also the **main focus of the general research on probe request deanonymisation**,

and their protection should therefore be prioritised to ensure user privacy. This focus on undirected probe requests can also be explained by highlighting the scenarios in which *directed* probe requests are used:

(i) Directly preceding the connection establishment,

(ii) during a connection, to maintain connectivity with the AP providing the strongest signal and

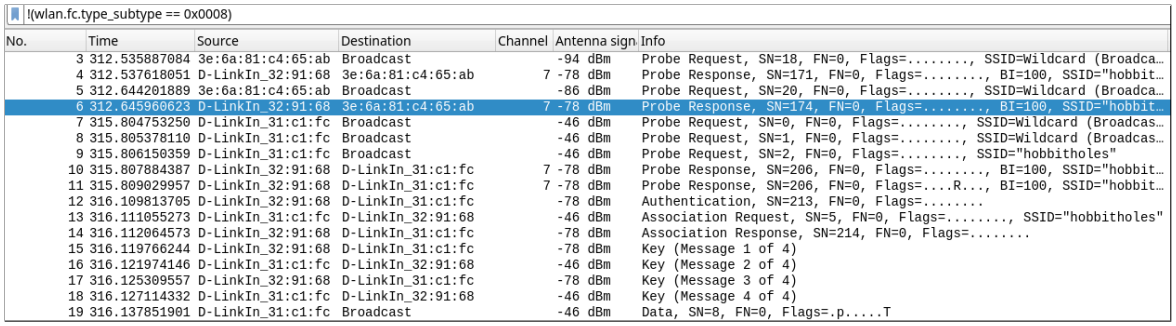
(iii) in search for hidden networks.

All three cases facilitate trivial device tracking via the identifiers contained in the probe requests: In the first two instances, the LAA is used as the source address, and is either an unchanging identifier or one that is stable for a predefined but typically extended amount of time (cf. Section 3.2.1). In the latter case, the SSID contained in the probe request serves as a trivial fingerprint. Extending the IE content minimisation to the protection of *directed* probe requests therefore does not improve privacy. Since neither connection establishment nor the connection itself require the use of active discovery, and can also be established by the use of passive discovery, the **reduction of IE content has neither influence on the robustness of the connection or the data transfer, nor on the connection speed or the reliability**.

To assess whether the proposal negatively impacts the time required for connection establishment, overall information exchange in network discovery, as well as security and privacy, the proposal is **analysed from various perspectives** in the following: Section 3.3.6.2 shows an evaluation of probe requests sent by five different devices to identify **whether a complex IE field enables faster connection establishment**. In Section 3.3.6.3, the impact of a reduced IE on the **functionality** of active discovery is evaluated. Section 3.3.6.4 then calculates the anonymity sets devices are contained within with respect to the IE content and thereby provides a **quantification of the privacy gain** achieved with a minimised IE content. Subsequently, Section 3.3.6.5 evaluates the **resistance to IE-Concerned Attacks** a reduced IE set induces.

3.3.6.2 Impact on Connection Establishment: IE Content Analysis

In theory, probe requests incorporate IEs to enable clients to convey their network discovery requirements, capabilities, and preferences to nearby APs. This exchange of information aids APs in delivering suitable and efficient responses, ensuring that clients can establish a connection in alignment with their specific needs and the network's capabilities. To find out whether the content of probe requests has an **actual impact on connection establishment,** the probe requests sent by five different devices are compared.



!(wlan.fc.type_subtype == 0x0008)

No.	Time	Source	Destination	Channel	Antenna sign.	Info
3	312.535887084	3e:6a:81:c4:65:ab	Broadcast		-94 dBm	Probe Request, SN=18, FN=0, Flags=........, SSID=Wildcard (Broadca…
4	312.537618051	D-LinkIn_32:91:68	3e:6a:81:c4:65:ab	7	-78 dBm	Probe Response, SN=171, FN=0, Flags=........, BI=100, SSID="hobbit…
5	312.644201889	3e:6a:81:c4:65:ab	Broadcast		-86 dBm	Probe Request, SN=20, FN=0, Flags=........, SSID=Wildcard (Broadca…
6	312.645960623	D-LinkIn_32:91:68	3e:6a:81:c4:65:ab	7	-78 dBm	Probe Response, SN=174, FN=0, Flags=........, BI=100, SSID="hobbit…
7	315.804753250	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=0, FN=0, Flags=........, SSID=Wildcard (Broadcas…
8	315.805378110	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=1, FN=0, Flags=........, SSID=Wildcard (Broadcas…
9	315.806150359	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Probe Request, SN=2, FN=0, Flags=........, SSID="hobbitholes"
10	315.807884387	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc	7	-78 dBm	Probe Response, SN=206, FN=0, Flags=........, BI=100, SSID="hobbit…
11	315.809029957	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc	7	-78 dBm	Probe Response, SN=206, FN=0, Flags=....R..., BI=100, SSID="hobbit…
12	316.109813705	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc		-78 dBm	Authentication, SN=213, FN=0, Flags=........
13	316.111055273	D-LinkIn_31:c1:fc	D-LinkIn_32:91:68		-46 dBm	Association Request, SN=5, FN=0, Flags=........, SSID="hobbitholes"
14	316.112064573	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc		-78 dBm	Association Response, SN=214, FN=0, Flags=........
15	316.119766244	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc		-78 dBm	Key (Message 1 of 4)
16	316.121974146	D-LinkIn_31:c1:fc	D-LinkIn_32:91:68		-46 dBm	Key (Message 2 of 4)
17	316.125309557	D-LinkIn_32:91:68	D-LinkIn_31:c1:fc		-78 dBm	Key (Message 3 of 4)
18	316.127114332	D-LinkIn_31:c1:fc	D-LinkIn_32:91:68		-46 dBm	Key (Message 4 of 4)
19	316.137851901	D-LinkIn_31:c1:fc	Broadcast		-46 dBm	Data, SN=8, FN=0, Flags=.p.....T

Figure 3.20: The client-TtT-metric measures connection establishment commencing with the first probe response to the last probe request sent via a randomised address (packet 6). The end point of the metric is the begin of data transmission in packet 19. Image source: [McD+24b].

The devices used for testing purposes represent common household appliances, including a Wi-Fi dongle, a laptop, a mobile phone, a tablet, and a single-board computer often utilized in IoT devices: They encompass a D-Link DWA-171 Nano Wi-Fi USB Adaptor, a laptop running an Intel AC 8265 wireless card, iPhone SE 2020 running iOS 17, an iPad Pro running iPadOS 15 and a Raspberry Pi Model 3B+. The experimental setup requires for the device to establish a connection with an access point, which is turned on at irregular intervals. Per device, each connection establishment is performed 7 times. The time required for connection establishment is measured using the client-TtT metric, introduced in the following.

Client-Time-to-Traffic Metric This experiment serves to measure the time it takes a device to switch from its temporary address used during probing to the LAA used in an established connection. To reduce the time required for transmission to include only this short interval, the client-Time-to-Traffic (client-TtT) is introduced: The starting point is the **last probe response received before the MAC address is changed to the LAA**. In Fig. 3.20, packet 6, marked in blue, shows this starting point. This last probe response before the MAC address change is chosen as the starting point, since its reception initiates the MAC address change to the locally used MAC address of the device, and thereby the connection establishment. The end point of the measurement is marked by the **first transmitted data frame**, which is visible in packet 19.

IE Content Evaluation Table 3.3 shows the results of the tests:

	Supported Rates (in Bytes)	Ext. Supported Rates (in Bytes)	DS Parameter Set (in Bytes)	HT Capabilities (in Bytes)	Vendor Specific (in Bytes)	Extended Capabilities (in Bytes)	HE Capabilities (in Bytes)	Interworking (in Bytes)	Frame Size undirected (in Bytes)	Frame Size directed (in Bytes)	client-TtT (in seconds)	Median	Standard Deviation
DWA-171	8	4	-	-	-	-	-	-	68	79	3.71	3.69	0.16
Intel 8265	8	4	1	26	7	-	-	-	108	277	1.92	1.67	0.73
iPhone SE	4	8	1	26	-	8	27	-	139	184	2.77	2.41	0.98
iPad Pro	4	8	1	26	28	8	-	7	152	163	2.54	2.8	1.17
Raspberry Pi	4	8	1	26	131	-	-	-	236	247	2.73	1.86	2.26

Table 3.3: Information Element tags of the undirected probe requests sent by five different devices, including frame sizes and client-Time-to-Traffic. Source: [McD+24b].

DWA-171 Undirected probe requests broadcast by the DWA-171 adaptor were limited in content and had a frame size of only 68 bytes. 12 of these construct the IE, containing 8 bytes of Supported Rates and 4 bytes for Extended Supported Rates. The DWA-171 adaptor required 3.71 seconds client-TtT on average.

Intel8265 The Intel 8265 transmitted undirected probe requests with a frame size of 108 bytes, with a client-TtT of 1.92 seconds on average and a median of 1.67. The IE contained Supported Rates, Extended Supported Rates, the DS Parameter Set, HT Capabilities, and vendor-specific information.

iPhone Undirected probe requests transmitted by the iPhone had a frame body size of 139 bytes and a client-TtT of 2.77 seconds. The IE tags included Supported Rates, Extended Supported Rates, the DS Parameter Set, HT Capabilities, Extended Capabilities and HE Capabilities.

iPad Pro The iPad Pro sent probe requests with a frame body size of 152 bytes and a client-TtT of 2.54 seconds. Its IE contained Supported Rates, Extended Supported Rates, the DS Parameter Set, HT Capabilities, Vendor Specific tags concerning Apple, Microsoft, and Broadcom, Extended Capabilities and the Interworking tag.

Raspberry Pi The Raspberry Pi transmitted undirected probe requests with the largest frame size of 236 bytes. Its client-TtT was 2.73 seconds and it contained the same IE tags as the Intel wireless card with the difference that its Vendor Specific tags concerned Broadcom, P2P and WPS, the last of which was 105 bytes long.

These results show that a **complex IE body does not necessarily imply fast connection establishment**. The largest frame body was transmitted by the Raspberry Pi, whose client-TtT of 2.73 seconds made it the third fastest. The fastest client-TtT

was achieved by the Intel 8265 wireless card, which had the second-smallest frame size. Hence, a **correlation between IE size and connection establishment could not be confirmed**. And while the DWA-171 Wi-Fi dongle had both the smallest IE and required the longest time for connection establishment, it is also the only device connected via a USB 2.0 Type A connector, while the other devices contained built-in Wi-Fi chips. Other possible explanations might therefore include optimised implementations and hardware.

Another interesting fact that could be explored in the future lies in the frame size of directed probe requests: directly before establishing a connection with a network, devices sometimes send additional *directed* probe requests using their own LAA to the network, including its SSID. The frame sizes of these directed probe requests are also compared in Table 3.3. The DWA-171 antenna, the iPad Pro and the Raspberry Pi only send an additional 11 bytes containing the SSID tag length and the cleartext SSID required for direct probing. In addition to the SSID, the iPhone sends three Vendor Specific tags, amounting to 184 instead of 139 bytes. The Intel 8265 with the fastest connection establishment sends the largest directed probe requests, with the extra information specifying Extended Capabilities, the Mesh ID, FILS request parameters, and additionally three Vendor Specific tags on WPS, P2P, and multi-band operation. While the larger frame body size of directed probe requests does not automatically imply correlation with faster connection establishment, it could still be interesting to investigate whether a very slim undirected probe request in combination with an extensive directed probe request during connection establishment would satisfy both the need for fast connection establishment while preserving the privacy of users in general probing.

In conclusion, the experiment shows that there **does not seem to be a direct correlation between extensive IE tags and fast connection establishment**. The proposed consequence to improve the privacy of users during active discovery remains to reduce the IE content to the bare minimum. This proposal is subsequently evaluated with respect to maintaining functionality of active discovery despite IE minimisation.

3.3.6.3 Impact on Functionality

As described in Section 2.1.4, connection establishment in Wi-Fi networks is done in four stages:

1) Active or passive device discovery

2) Authentication

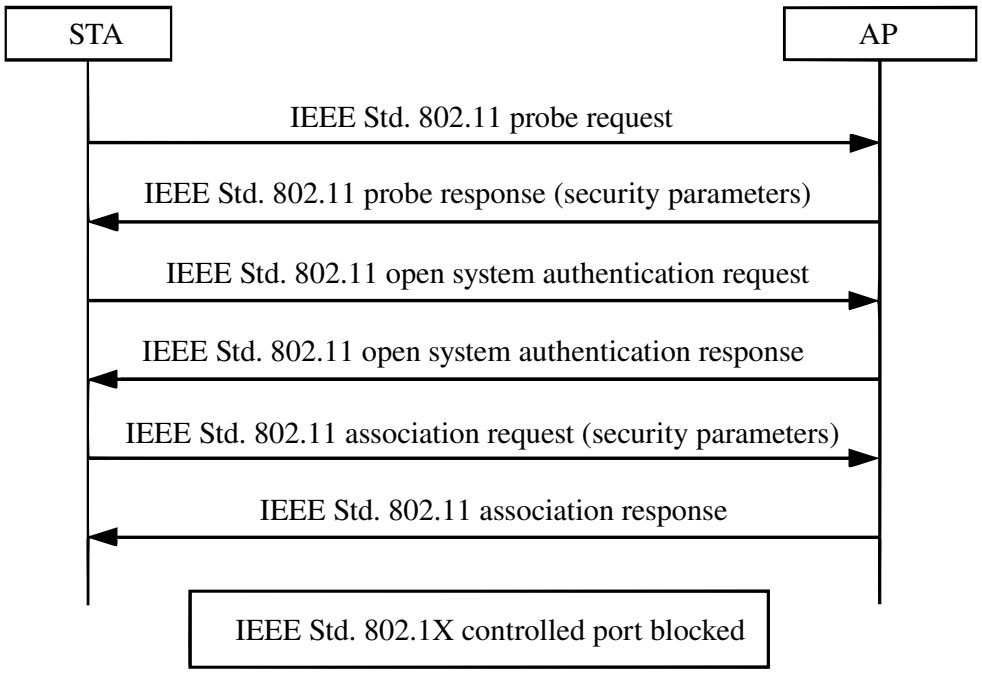


Figure 3.21: Association Establishment of IEEE 802.11 according to the standard. Adapted from [IEE20, Fig. 4-30].

3) Association

4) Robust Security Network Association (RSNA)

The IEEE Wi-Fi standard contains the depiction of association establishment shown in Fig. 3.21. In it, both the probe response sent from the AP, as well as the association request sent from the mobile device, are marked to transmit "security parameter". The fields contained in association requests are defined in [IEE20, Table 9-34] and those of probe requests in [IEE20, Table 9-38]. These findings suggests the following hypothesis:

Hypothesis: The content of the probe requests is, also according to Fig. 3.21, irrelevant to the exchange of security parameters between AP and the mobile device.

To prove or disprove this hypothesis, and therefore establish whether a reduction of IE content of probe requests impacts the functionality of active discovery, an in-depth analysis of the fields of probe requests, probe responses and association requests is therefore presented in the following.

A comparison shows that both probe requests and association requests have a large overlap of fields: While association requests are comprised of a maximum

Field	AP-CSN	Change Sequence	Channel Usage	Cluster Probe	DSSS Parameters	Estimated Service Parameter Inbound	Estimated Service Parameter Outbound	Extended Request	FILS Request Parameters	Mesh ID	PV1 Probe Response Option	Request	S1G Relay Discovery	Vendor Specific Request	SSID List
In Probe Requests	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
In Probe Responses	✓	✓	✓	||	✓	✓	✓		‡	✓	§		✓		**
In Association Requests									‡‡		§		*		**

||: Extended Cluster Report
‡: FILS Indication
‡‡: FILS Session, FILS Public Key, FILS Key Confirmation, FILS, FILS HLP Container, FILS IP Address Assignment
§: Substituted by a multitude of different fields
*: S1G Capabilities, S1G Relay, S1G Relay Activation
**: In practice, this is transmitted via several consecutive probe requests containing one SSID field each

Table 3.4: The fields that can be found in probe requests, but which are *not* in existence in association requests. Additional markers underline their existence or their substitution by different fields in both association requests and in probe responses. The fields Extended Request, Request and Vendor Specific Request are highlighted in red since they are the only fields not substituted or in existence in either probe responses or association request. Adapted from [McD+24b].

of 46 fields, probe request contain 34, 19 of which are overlapping with those of association requests. The remaining 15 fields, which are only contained in probe requests but not in association requests include the following:

- Request
- SSID List
- DSSS Parameters
- Channel Usage
- Extended Request
- Mesh ID
- Change Sequence
- AP-CSN
- Estimated Service Parameters Inbound
- Estimated Service Parameters Outbound
- FILS Request Parameters
- S1G Relay Discovery
- Vendor Specific Request
- PV1 Probe Response Option
- Cluster Probe

They can also be seen in Table 3.4.

The table also allows for an analysis of the fields contained in association requests: A closer inspection reveals that while association requests contain neither FILS Request Parameters nor S1G Relay Discovery (which are present in probe requests), instead they contain fields to negotiate FILS session activation and S1G Relays. In the case of passive discovery via beacons, their use must hence have been negotiated elsewhere: In inspection of the 84 possible fields contained in probe responses shown in [IEE20, Table 9-39] reveals that out of the 15 remaining elements that are present in probe requests, but not in association requests, eight are present in probe responses:

- DSSS Parameters
- Channel Usage
- Mesh ID
- AP-CSN
- Estimated Service Parameters Inbound
- Estimated Service Parameters Outbound
- Change Sequence
- S1G Relay Discovery

Additionally, both probe responses as well as beacons can contain a FILS Indication field; this explains why connections that were established using passive discovery can have exchanged FILS capabilities without the use of the FILS Request Parameters in probe requests.

The remaining seven fields that can be contained in probe requests but neither in association requests nor probe responses are:

- Request
- Extended Request
- Vendor Specific Request
- FILS Request Parameters
- SSID List
- Cluster Probe
- PV1 Probe Response Option

The lack of both Cluster probe and FILS Request parameters is substituted by the Extended Cluster Report field, respective the FILS indication contained in probe responses, and the subsequent exchange of FILS parameters contained in the association frame. The SSID List groups several known SSIDs from the PNL; in practice, separate probe requests are transmitted per SSID instead. The PV1 Probe Response Option [IEE20, Tables 9-305 - 9-310] element is a collection of capabilities and compatibility, bundled in several bitmaps. With it, a device can request to receive responses regarding the capabilities of the AP. Most of the fields whose inclusion can be requested via the bitmaps are already optionally present both in association requests and in probe responses. Additionally, connection establishment via passive discovery is possible without the PV1 Probe Response Option, just like the omission of the Request, Extended Request and Vendor Specific Request fields is tolerable in passive discovery. They therefore appear to be unnecessary when exchanging capabilities. In summary, the only fields of probe requests which are not in existence or substituted by another field in probe responses or association requests are Extended Request, Request and Vendor Specific Request, as can be seen in Table 3.4. Since they are neither contained in the beacons used in passive discovery, they are likely irrelevant for connection establishment, since passive discovery is possible without them as well. Altogether, these findings support the previously stated hypothesis, and lead to the following conclusion:

Result: The content of the probe requests is irrelevant to the exchange of security parameters between the AP and the mobile device.

Another test that can help to further support the hypothesis is a comparison of the probe responses transmitted in response to different probe requests. To do this, three different probe requests are transmitted, and the responses of nearby AP compared. The probe requests vary greatly in content: The first one is a scripted, minimised probe request containing only Supported Rates and the empty SSID field, as shown in Listing 3.1. The second and third are the Intel 8265 and the Raspberry Pi used in previous section. With this selection, the responses to both a bare-minimum probe request containing 64 bytes, a medium-sized one at 108 bytes as well as the most verbose probe request recorded in Section 3.3.6.2 with 236 bytes can be compared. The comparison reveals the following:

Result: Irrespective of the contents of the IE transmitted via probe requests, the AP always responds with one universal probe response.

An additional comparison of probe responses transmitted by 10 different APs within the transmission range of the experimental setup, collected over an extended period of time, further confirms this observation: Regardless of the content of the probe requests, an AP always responds with the same probe response. In conclusion, the hypothesis that the content of the probe requests is irrelevant to the exchange of security parameters between AP and the mobile device could be proven due to the following observations:

- All relevant information required for connection establishment are exchanged via the combination of probe response and association request.
- The content of the IE of probe requests is irrelevant with respect to the probe responses of the AP, since the probe responses always contain the same information.

This shows that a general reduction of IE content to contain only the bare minimum of information is **technically feasible**, would aid in **reducing complexity and redundancy in active discovery**, and would **increase user privacy in wireless communication**. In the next section, this impact on privacy is further evaluated with respect to quantifying the privacy gain achieved by the IE minimisation.

3.3.6.4 Impact on Privacy: Anonymity Set Determination

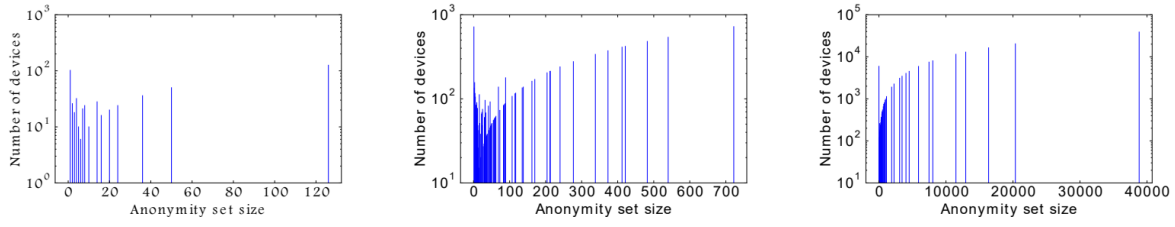
In order to gauge the impact a reduction of the IE content has on device privacy, a good estimator is the calculation of the anonymity sets that result from the reduction. An anonymity set defines how many users share the same identifiers and

are therefore indistinguishable from each other. Vanhoef et al. [Van+16] conducted an evaluation of the anonymity set size of probe requests in 2016, comparing the anonymity sets that devices that share the same IE fingerprint. It can be seen in Fig. 3.22a. Their data encompasses **two self-recorded data sets from their lab and a train station**, which contain 120 000, respective 110 000 probe requests, and additionally **the complete Sapienza data set** [Bar+22], a large data set encompassing 8 000 000 anonymised probe requests recorded in various scenarios, e.g. during large gatherings of people or in highly frequented areas like malls or train stations. The Sapienza data set is particularly useful for mapping MAC addresses to devices, since it was recorded in 2013, before the introduction of MAC address randomisation: iOS 8 was the first to introduce MAC address randomisation in 2014, and Android incorporated it in 2015, in Android 6.0 (cf. Section 2.1.3). It can therefore be assumed that **the amount of MAC addresses in the data set corresponds to the number of devices present**, which is necessary for assessing anonymity sets. To use a more recent data set with a large amount of devices using randomised MAC addresses the addresses would first need to be de-randomised to accurately determine the actual number of participating devices and map their probe requests. This requires to define an attack to correlate randomised MAC addresses to real devices, which is out of scope of both [McD+24b] as well as this thesis, but has been performed in several publications (cf. Section 3.1 and Section 3.2.4).

While the whole Sapienza data set contains a variety of subsets, encompassing political meetings, a pope's audience, a mall and a university campus, only the train station subset likely contains a large variety of probe requests from passersby and only a small amount of "lingering" devices. While one of the data sets Vanhoef et al. used in their analysis, which is also going to be presented in the following, encompassed the complete Sapienza data set, **only the train station subset of the Sapienza data set** was analysed in [McD+24b], as it is most likely to contain the **largest variety of probe requests**.

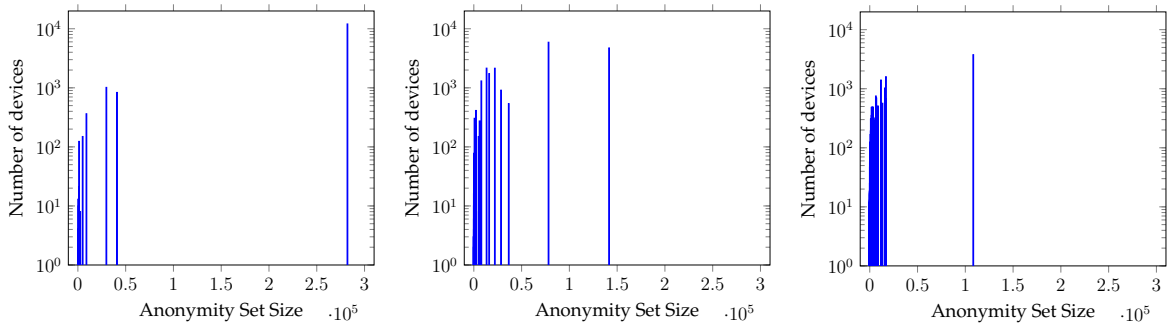
To estimate the usefulness of IEs for device identification, Vanhoef et al. calculated the anonymity sets across the devices within the previously described data sets. The device fingerprint they take into account spans the whole IE, including all transmitted tags. In all three data sets shown in Fig. 3.22a, it can be observed that one high spike on the right encompasses a large number of devices sharing the same anonymity set. Simultaneously, a large amount of devices are distributed over the left sides of the graphs, making up very small anonymity groups, which allow for easy fingerprinting of the devices contained within. In the lab data set shown in Fig. 3.22aa, it appears that around 105 share the same anonymity set, which amounts to 21 % of the devices. This dataset consists of a large amount of probe requests, and simultaneously a small amount of distinct MAC addresses, which points at only a small number of devices actually emitting the probe requests. For the train-station

data set shown in Fig. 3.22ab, the largest anonymity set encompasses 1.6 % of the devices, and in the Sapienza data set shown in Fig. 3.22ac, the largest anonymity set encompasses 0.75 % of the devices.



(a) `Lab` (b) `Train-station` (c) `Sapienza`

(a) The results of the analysis Vanhoef et al. presented on anonymity set size calculations on three different dats sets. The largest anonymity sets encompass 21 % of the devices in the lab data set (a), 1.6 % of the devices in the train station data set (b) and 0.75 % in the Sapienza data set (c). Image source: [Van+16].



(b) Sapienza data set reduced to only Supported Rates. The largest anonymity set encompasses 82.55 % of the devices.

(c) Sapienza data set reduced to Supported Rates and the DS Parameter Set. The largest anonymity set encompasses 40.49 % of the devices.

(d) Sapienza data set reduced to Supported Rates, the DS Parameter Set and HT Capabilities. The largest anonymity set encompasses 25.78 % of the devices.

Figure 3.22: Anonymity set size comparison and calculation with reduced IE content on a subset of the Sapienza data set. The IE content is constrained to include (b) Supported Rates only, (c) Supported Rates and the DS Parameter Set, and (d) Supported Rates, the DS Parameter Set, and HT Capabilities. Adapted from [McD+24b].

To evaluate the anonymity set sizes with reduced IE content, a subset of the Sapienza data set is used, the trainstation record. Since in 2013, only few devices used LAAs and most used their UAAs, the data set can be pruned to only contain devices transmitting their UAAs using the Wireshark filter `!(wlan.sa[0] & 0x02)`. This filter excludes all locally assigned addresses by displaying only packets in which

in the first byte of the source address, accessed via `wlan.sa[0]`, the second least significant bit (`0x02`) is not (`!`) set (cf. Section 2.1.1 for a background on the U/L bit). The thereby acquired subset encompasses 374 736 probe requests sent from 14622 distinct MAC addresses.

The results of this evaluation can be seen in Fig. 3.22. If **only Supported Rates** are included in the IEs (cf. Fig. 3.22b), the 14622 devices are divided into 19 anonymity sets. The largest anonymity set encompasses 82.55 % of the devices. These all share the same Supported Rates of 2, 4, 11, and 22.

Fig. 3.22c shows the results of including **Supported Rates and the DS Parameter Set**. Here, the largest of the resulting 61 anonymity sets encompasses 40.49 % or 5920 devices. 60.18 % of the devices did not include the DS Parameter Set, the others transmitted either channels 1, 2, 11, or 12.

In Fig. 3.22d, a reduction of the IE content to include Supported Rates and the DS Parameter Set is shown. In comparison to previous evaluations, the distribution is spread out significantly, encompassing 276 sets, the largest of which contains 25.78 % of the devices. In its HT Capabilities, it contains the information `0c 18 1b ff`, which constitutes disabled SM Power Save, maximum A-MSDU length of 7935 bytes, the use of DSSS/CCK in 40 MHz, A-MPDU Parameters, and supported RX Modulation and Coding Scheme Set.

The anonymity set sizes calculated in the previous paragraphs, particularly that with the IE content reduced to encompass only Supported Rates, places 82.55 % of the devices in the same anonymity set. Particularly in comparison with the evaluation of anonymity sets calculated by Vanhoef et al., which shows that a real-world data set places 0.75 to 1.6 % of the devices in the largest anonymity set, a reduction of IE content has a tremendous effect: **The less content an IE has, the larger the anonymity group single devices disappear in.** The analysis of IE content previously presented in this section additionally showed that the amount of information contained within the IE does not correlate with fast connection establishment. The calculation of anonymity sets additionally underlined that a reduced IE content is **vital to preserve the privacy** of users as well as possible. The next section emphasizes the importance of the IE for fingerprinting devices and thereby highlights how a reduced IE content can mitigate most attacks targetting probe requests.

3.3.6.5 Impact on Security: Resistance to IE-Concerned Attacks

In the previous sections, the requirements for the IE content of probe requests were analysed from several perspectives: The impact that the IE has on connection establishment was tested, and a subsequent analysis determined that probe request

Authors	MAC Address	SSID	Supported Rates	Ext. Supported Rates	DS Parameter Set	HT Capabilities	VHT Capabilities	Extended Capabilities	Vendor Specific	all other fields	IE Field Sequence	BSS Membership	Transmission Frequency	Sequence Number	RSS
Pang et al. [Pan+07]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Cunche et al. [CKB14]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Freudiger et al. [Fre15]	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
Vanhoef et al. [Van+16]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	-
Robyns et al. [Rob+17]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	✓	-	-
Zhao et al. [Zha+19]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Dagelić et al. [DPČ19]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
Gu et al [Gu+20]	-	-	✓	✓	✓	✓	✓	✓	-	✓	-	-	-	-	-
Uras et al. [Ura+20]	-	-	-	✓	✓	✓	✓	✓	-	-	-	✓	-	-	-
Tan et al. [TC21]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	-	-	✓	✓
Pintor et al. [PA22]	-	-	-	-	-	✓	-	✓	✓	-	-	-	-	-	-
He et al. [HTC23]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓

Table 3.5: Comparison of the elements used for fingerprinting attacks across various publications as listed in Section 3.2.4, with blue highlighting for Supported Rates, purple for DS Parameter Set and green for HT Capabilities. Publications that selected certain fields due to their distinguishing and unique features are marked in grey. Adapted from [McD+24b].

content is irrelevant for functionality, as all required parameters are exchanged via the combination of probe response and association request. Subsequently, the impact on privacy was determined via the calculation of anonymity sets, which showed that a reduction of IE content has a great positive impact on privacy. In this section, the impact of a reduced IE content on security is determined: It strives to answer the question how resistant to IE-concerned attacks devices are when their IE is limited to the maximum.

The basis of this comparative section is Table 3.2 constructed in Section 3.2.4, a coloured-in version of which can also be seen in Table 3.5. Section 3.2.4 showed that many publications have introduced attacks that can be used to track devices via identifiers contained in the probe request, and circumvent anonymisation strategies like MAC address randomisation, sequence number randomisation and SSID omission. The first few publications that showed the possibility to track devices via their probe requests commonly concentrated on the MAC address and SSID, e.g. Pang et al. [Pan+07], Cunche et al. [CKB14], Freudiger et al. [Fre15], and later Zhao et al. [Zha+19] and Dagelić et al. [DPČ19]. They chose selective fields with high entropy (at the time of their studies) to discriminate between devices, and their rows are shaded in gray. Two other publications subsequently concentrated on selective fields, with Uras et al. [Ura+20] concentrating on Extended Supported Rates, BSS Membership, HT Capabilities, Extended Capabilities, DS Parameters, VHT Capabilities and two reserved tags, and Pintor et al. [PA22] focussing only on Extended Capabilities, HT Capabilities, and vendor-specific values. All of these publications

that chose selective fields due to their discriminatory values have in common that **the Supported Rates have no relevance for their fingerprinting technique**. With respect to the DS Parameter Set, out of all of the publications mentioned, only Uras et al. decided to include it in their device fingerprint, while the HT Capabilities were included by both Uras et al. and Pintor et al.

On the other hand, many publications resorted to using the whole IE as a fingerprint, these are shown as unmarked (white) rows in the table. Of these, some, e.g. Gu et al. [Gu+20], exclude certain fields that they deem user-modifiable and therefore unfit as device identifier, others used even more identifiers in addition to the whole IE, like Vanhoef et al.[Van+16] and He et al. [HTC23], who additionally used the field sequence, Robyns et al. [Rob+17] who additionally focus on the transmission frequency or Tan et al. [TC21] and He et al. [HTC23], who additionally focus on the sequence number and the received signal strength (RSS). While all of these publications *do* include both Supported Rates, DS Parameter Set and the HT Capabilities, in their attacks, these fields are only one attribute among many. These results demonstrate that **by using the (empty) SSID field in combination with Supported Rates, all attacks targetting fields with high entropy can be thwarted**. While the other attacks include the Supported Rates for completeness, their content does not play a major role in distinguishing between individual devices. This also becomes apparent when observing the anonymity set determination previously presented: Here, it could be shown that the Supported Rates discriminate only 19 possible values. Altogether, this shows that the minimisation of IE content reduces the attack surface drastically and prevents existing attacks.

3.4 Conclusion

This chapter focuses on the process of device discovery in Wi-Fi networks, the privacy risks arising from it, and their mitigations. A means of active network discovery features the use of probe requests. Albeit their use is to discover nearby networks, they provide attackers with **several attack vectors**:

- The SSIDs contained in directed probe requests that are necessary to establish a connection with a hidden network are an easy means of fingerprinting and tracking a device.
- The transmission of SSIDs also renders devices susceptible to fake AP attacks.
- Transmitted SSIDs can disclose potentially sensitive information on users as shown in the evaluation of SSID content in Section 3.2.3.2.

- Using the SSIDs, attackers can **localise visited places** via network maps like WiGLE, as Section 3.2.3.3 demonstrates by verifying the existence and location of networks using the network map WiGLE.

Even if the device fingerprint is reduced by not including known SSIDs into the probe requests, their IE field still serves as an excellent means of tracking devices, as Section 3.2.4 shows. And albeit Section 3.2.1 shows a positive development in mobile OS to introduce privacy preserving techniques, it also displays their non-existence in older OS which are still widespread, particularly with respect to Android devices, as Section 3.2.2 underlines. Since the use of probe requests is the prevalent means of network discovery, the remaining sections strive to introduce means of improving the privacy friendliness of network discovery:

A proposed circumvention of the use of cleartext SSIDs is introduced in Section 3.3.2: A **hash-based scheme** that includes the SSID and, as a source of randomness, the currently used MAC address and the sequence number of the packet to hide the true SSID. To identify whether a directed probe request was directed at them, access points can then hash their own SSID, the MAC address and the sequence number of the received packet and compare it to the received hash. This way, the use of hidden networks is still possible while the possibility to track users via an unchanging identifier is mitigated.

To additionally provide users with more control over their PNL and probe requests, **UI improvements** were suggested in Section 3.3.3: These encompass SSID entry safeguards to protect users from accidentally storing non-existent networks in the PNL and thereby extending the attack surface via probe request tracking. Additionally, PNL entry expiration dates would allow for direct removal of one-time-use SSIDs after a set amount of time, and an auto-join prompt would ensure that users have better control over whether they want to automatically connect to a newly added network or not. Both controls, implemented as user prompts, would improve user security by rendering them less prone to fake AP attacks. While the suggestion to provide users of Apple devices with the same means of modifying their PNL at all times such as Android devices facilitate was still a necessary proposition in 2022 when [McD+22] was published, it was implemented in Apple devices later in 2022, with the introduction of iOS 16. Whether this was a reaction to the publication or not could not be established. The last suggestion proposed for increased user controls was **probe request silencing**: This would introduce users with means to turn off active discovery and rely on passive discovery only. While this solution contains certain draw-backs, as is renders connections with hidden networks impossible and might slow down connection establishment while increasing battery usage, it is an excellent approach to disable tracking via probe requests in general.

The possibility to instead **rely on beacons only** is subsequently explored in more technical detail in Section 3.3.4: In case of the connection with a non-hidden network, a privacy friendly device discovery can be performed using beacons only. Several publications suggested and tested this approach. They found that passive discovery is only marginally slower than active discovery. Additionally, they developed a scheme that improves passive discovery tremendously, introducing dynamic priority passive scanning as a privacy-friendly alternative to the currently prevalent active discovery. While it is unlikely that the current State-of-the-Art of using probe requests for network discovery is going to be completely discontinued presently, smaller system changes to improve user privacy are more likely to be implemented.

Such changes are introduced with generic probe requests and a generic address for probing: While the use of MAC address randomisation has been established in mobile devices starting with iOS in 2014 [Aar14] and Android in 2015 [And23], randomisation is, as of today, not standardised yet [ZBA24]. Manufacturers implement it using their own randomisation schemes; some of them use persistent OUIs and randomise only part of the MAC address. This can be used to infer information on the device by attackers [Mar+17; Van+16]. Such an attack can be countered with **a generic MAC** address as introduced in Section 3.3.5: One generic address used by all devices during active discovery. The approach is tested and evaluated, both to assess its scalability as well as compare it to established schemes. The results show that it performs comparable to MAC address randomisation while enhancing privacy even more.

A means of tracking devices despite the use of MAC address randomisation is by fingerprinting the IE element contained in the probe request. Various publications use either a subset or the full IE content to fingerprint devices, as shown in Section 3.2.4. An evaluation of the fields necessary for probe requests to receive probe responses shows that only Supported Rates and the SSID fields have to be included in the probe request, both of which can be empty. These results are then used to propose an efficient defence against fingerprinting attacks via the IE content in Section 3.3.6: **reducing the IE content to the bare minimum**, meaning including only the SSID and Supported Rates fields. Subsequently, the impact a reduced IE has on device functionality, privacy and security is analysed: First, the time required for connection establishment with respect to the IE content was determined, which showed that no correlation could be established. The device with the fastest client-TtT had the second-smallest frame size in undirected probe requests. The reason for the varying times required for connection establishment therefore most likely lies in other areas, e.g. optimised hardware or software. Subsequently, the impact of a reduced IE on the functionality was established by analysing all possible fields contained in the IE of probe requests, and determining whether their existence is required for connection establishment. The results showed, that the content of probe requests

is not required for connection establishment, since (a) all fields contained in the probe request are communicated also via the combination of probe response and association request and (b) probe requests can not even be used to request connection-specific parameters, since all recorded probe responses *always* contained the same content, independent of the request they responded to. The impact a reduced IE field has on the functionality was therefore determined to be negligible.

To analyse the benefits of this approach with respect to user privacy, the **anonymity set sizes** with reduced IE content were evaluated using the Sapienza train station data set. This evaluation showed that when including only Supported Rates, the probe requests can be grouped into 19 anonymity sets, the largest of which contains 82.55 % of the requests. This is a tremendous improvement compared to the calculations of the anonymity sets of unmodified probe requests done by Vanhoef et al. [Van+16], where the largest anonymity group on comparable data sets contained 0.75 % to 1.6 % of the probe requests.

Lastly, the impact on security with respect to **resistance to IE-concerned attacks** is evaluated by analysing which fields play a role in fingerprinting IE content: None of the publications that determined field entropy and selected only fields with high entropy included the Supported Rates in their analysis. The other publications either chose to use all fields and included or excluded certain tags, but none of them placed a focus on the discriminatory value of the Supported Rates field in the face of fingerprinting IE content: As shown in the anonymity set evaluation, a data set containing 374 736 probe requests only exhibits 19 different entries in Supported Rates. The significance for fingerprinting is therefore likely negligible.

In summary, this chapter compared and evaluated several schemes to protect the privacy of users while using probe requests. Probing for hidden networks can be implemented in a privacy-friendly way using hash-based SSID transmissions. A generic address protects the users in case of lacking MAC address randomisation implementations, and generic undirected probe requests containing only Supported Rates and the SSID tags in the IE field greatly improve user privacy during active discovery.

4

Connection Establishment - Using a VPN in a Captive Network

The use of Wi-Fi networks can be dangerous with respect to data confidentiality and privacy, as a multitude of attacks have previously shown [TB09; VP17; VR20]. In public Wi-Fi networks, this attack surface is widened even more, since they are often unencrypted or easily attackable, as Section 4.2 will show.

A common mitigation of these privacy risks arising from surfing in public Wi-Fi networks is the use of a Virtual Private Network (VPN): This way, all traffic is encrypted and tunnelled via a VPN server, and attackers attempting to attack and decrypt Wi-Fi traffic can neither infer information on the content of the transmitted packets, nor on the destination addresses. This chapter focuses on the use of VPNs in public Wi-Fi networks and their fulfilment of requirements for privacy-friendly and secure implementations, as well as their functioning in the face of captive portals contained within public Wi-Fi networks.

As this chapter shows, the use of a VPN does not automatically guarantee privacy, security and the desired functionality – their use can entail certain pitfalls with respect to connection establishment, which are highlighted via two examples: On one hand, when using VPNs, users have to trust the VPN provider to ensure that all packets are tunnelled via the VPN server. Packets transmitted outside the tunnel are denoted as leaks or leakage, and such leakage can expose potentially sensitive information on a user and reduces the trust placed in the VPN provider. Apart from the information required to establish the VPN connection, no traffic *should* be leaked

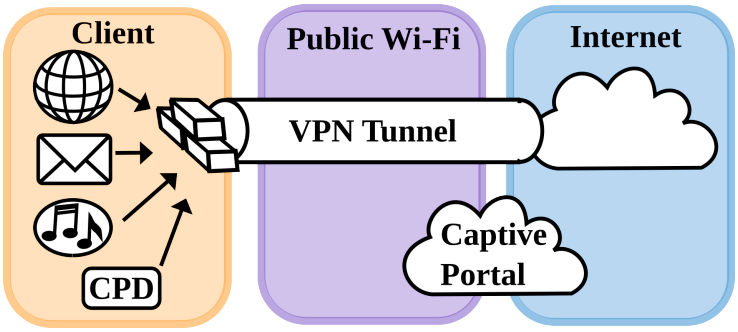


Figure 4.1: A captive deadlock: The VPN is active before the captive portal was remediated to reduce leakage that could occur via the network without the use of the VPN. Since the VPN blocks all traffic except for that necessary to establish the VPN, captive portal detection (CPD) is also blocked. In this situation, neither can the captive portal be remediated nor, due to the network block caused by the captive network, the VPN tunnel be established.

outside the VPN tunnel. The documented security features to ensure data security inherent to native VPN clients of Android, iOS, macOS, Windows and Ubuntu are summarised in Section 4.5. As Section 4.6 shows, however, data leakage *does* occur across all tested VPN applications and platforms.

Another problem arising with the use of VPNs in public Wi-Fi networks is the potential for *captive deadlocks*, which is also illustrated in Fig. 4.1: Public Wi-Fi networks often contain captive portals (cf. Section 2.2), which suspend the connection to the internet until the terms of the captive portal have been fulfilled. This can, for example, entail the agreement to policies inherent to the Wi-Fi provider or credential submission. To ensure that no leakage of data occurs surrounding the connection establishment and connection with a public Wi-Fi network, users can enable the VPN service before connecting to the Wi-Fi network, under the assumption that the VPN client withholds all traffic until the VPN tunnel is successfully established. In a *captive network*, a public Wi-Fi network containing a captive portal, this can cause a *captive deadlock*, in which the VPN blocks all traffic required to remediate the captive portal, and the captive portal simultaneously blocks the traffic required for VPN tunnel establishment. Therefore, requirements for secure bootstapping of VPNs as introduced in Section 4.4 have to be fulfilled to ensure functionality and privacy. Bearing these requirements in mind, both the susceptibility to captive deadlocks as well as the privacy-preserving techniques used in different VPN clients are tested and presented in Section 4.6. A proposed mitigation to avoid captive deadlocks discussed in Section 4.7 concerns a *selective VPN bypass*. Here, the VPN is extended to implement captive portal detection and remediation before VPN activation.

Relevant Publications Sections 4.4, 4.5, 4.6 and 4.7, are largely based on [Bur+21], to which I contributed as a co-author.

4.1 Preliminary Studies and Related Work

In their 2013 publication, Cheng et al. [Che+13] analyse the **privacy leakage users of public Wi-Fi hotspots** are exposed to. They differ between DNS queries, web browsing, online advertising and search engine queries. Bear in mind that, in 2013, the use of Transport Layer Security (TLS) encryption was considerably less common than it is nowadays, with Let's Encrypt[1], now being a major distributor of TLS certificates for both commercial and non-commercial use, commencing their certificate issuance in 2015 [Gro23]. Additionally, even though Wi-Fi Protected Access 2 (WPA2) had been introduced in 2004, it was, by no means, in ubiquitous use yet in 2013. This is be shown in Section 4.2 and particularly Fig. 4.3, which shows the types of protection used in public Wi-Fi networks in 2016, where technologies preceding WPA2 were used to provide almost 32 % of the hotspots. It is hence unsurprising that Cheng et al. discovered a significant information leakage, with up to 68 % of the packets captured from a device being traceable to a user name. Albeit TLS is now very widespread to secure communication with websites and online services, and insecure protection mechanisms contained in Wired Equivalent Privacy (WEP) and WPA are in considerably less use, privacy issues arising during the use of public Wi-Fi networks are still discussed now: In 2021, Lotfy et al. [Lot+21] published a meta data analysis of traffic captured at a university campus. They classify the internet activity into the categories search engine queries, web browsing and the use of social networks, the predominantly visited one being social networks. Among the recorded packages, the authors also discovered several user names, email addresses and phone numbers. Sangeen et al. [San+23] performed a similar analysis on packets captured via a public hotspot in 2023. Like in the analysis Lotfy et al. presented, most of the traffic captured was related to social networking and the use of search engines. Their experiment entails the provisioning of a publicly usable unencrypted access point which allows the users to surf for 15 minutes, and afterwards presents them with a screen showing their captured credentials. The experiment was continued on several consecutive days. The authors exhibited a considerable reduction of users on days following the presentation of credentials, which they attribute to the Protection Motivation Theory (PMT), according to which users take protective action to safeguard against a threat in case the threat is perceived as significant. From the reduced use of the hotspot on days following the presentation of sniffed credentials, the authors derive that users did in fact initiate protective action against credential leakage, meaning to

1. https://letsencrypt.org/

cease the use of unknown hotspots. Another examination of the use of public Wi-Fi networks was performed by Sombatruang et al. [Som+19] in 2019, but from a very different perspective: They convinced a number of 71 participants to install an app designed to monitor the connection to Wi-Fi networks, mobile data allowance and the battery power. Whenever a participant of the study would connect to an open and unsecured network, the app would record both battery power and the remaining data allowance. The study concludes that the participants were particularly driven to connect to unsecured hotspots whenever their allowance reached a stage of around 30 %. They also ascertained that the knowledge of risks inherent to the use of open Wi-Fi networks does not decrease the likelihood for participants to connect to an open network if the data allowance is below the threshold.

In their 2018 publication, Li et al. [LZM18] use leaked Wi-Fi traffic to conduct a demographics study. They describe various sources of traffic and related meta-data leakage, monitor the traffic and apply machine learning techniques to categorise users by gender and education. Their model classifies users' genders with an accuracy of up to 82 %, their education level with up to 78 % in unencrypted traffic, which makes up around 10 % of the surveilled data. In encrypted traffic, the gender can be predicted with a precision of up to 69 %, and the education level with up to 76 %. The authors suggest that the use of VPNs or the TOR network can mitigate the leakage and reduce the possibility for attackers to evaluate the meta data.

The problems arising from the use of public Wi-Fi networks not only extends to attackers sniffing traffic within the network but also to the parties provisioning the Wi-Fi network: Ali et al. [Ali+19] performed an analysis of **captive portals located within public Wi-Fi networks** in 2019 and found that most hotspots analysed utilise persistent third-party tracking via cookies stored during the interaction with the captive portal. This leads to users being trackable beyond the imminent reach of the access point; in fact, some of the cookies could be used to follow browsing behaviour for 20 years and longer.

While mechanisms like third-party tracking can *not* be circumvented by the use of a VPN, studies like those presented by Cheng et al., Lotfy et al. and Sangeen et al. are also partially possible, since within a network, both the metadata of the packets, as well as the destination address are visible to anyone sniffing the traffic. This can be mitigated by the **use of a VPN**, which disguises the destination address as its own address by encapsulating the packet, and, upon receiving a response, re-encapsulates it into a packet with the VPN server as the source address (cf. Section 2.3).

VPNs additionally encrypt the content of the packets, because of which their use enables privacy friendly and confidential internet use even in unsecured networks. Nevertheless, they also exhibit certain flaws, that have been investigated and reported on in much detail during the recent years:

In 2018, Khan et al. [Kha+18] performed an analysis of the security of 62 commercial VPNs. They found that several of the VPN clients they analysed leaked user data, particularly when the VPN tunnel fails to establish: Instead of blocking traffic while re-attempting to establish the tunnel, 58 % of the tested VPN services allowed traffic to leak outside the tunnel. Additionally, 19 % of the services failed to tunnel IPv6 traffic, transmitting it outside the tunnel instead. Two of the tested VPN clients also leaked DNS traffic, as they failed to modify the DNS servers the system was using. They also found that 10 % of the VPN providers hosted their VPN servers in countries other than those advertised.

Ikram et al. [Ikr+16] analysed **Android apps with VPN permissions**. They crawled the app store for apps containing keywords related to VPN services and decompiled around 1.5 Million apps to examine their `AndroidManifest` file for requests to access VPN permissions. This way, they identified 283 apps that utilise the VPN permissions of the Android OS. The VPN permissions allow the app to redirect all network traffic via a virtual interface, and thereby possibly intercept or manipulate it, or forward it to additional parties. Ikram et al. discovered that not all apps with VPN permissions are in fact VPN apps: among others, traffic optimisation and antivirus apps also require access to all network traffic to optimise routing, respective scan for malicious activity. While a lot of the use cases seem legitimate, the authors found that 72 % of free VPN apps and 35 % of premium services included third party trackers – some as many as 14 different tracking libraries – thereby delivering targeted advertisements to users. Ikram et al. additionally scanned the APKs for potential malware presence, and discovered that 38 % generate one or more positive malware reports from the categories adware, trojan, malicious advertising, riskware and spyware. The analysis shows that the Android VPN permissions are in fact maliciously misused by a surprisingly large percentage of app vendors, and that using VPN apps can, in itself, be an inherent risk to privacy and security, while appearing to be the exact opposite. Ikram et al. support the notion uttered by Khan et al. [Kha+18], that a lot of the VPN providers disregard IPv6 traffic and DNS requests. They found that 8 % of the analysed apps didn't tunnel IPv6 traffic and 55 % of the free apps and 60 % of the premium apps used Google to resolve DNS requests, and only 7 % respective 10 % used their own DNS resolvers.

A similar analysis was performed by Wilson et al. [WMB20] in 2020, but with a focus on iOS apps: They downloaded 57 VPN applications from the iOS store, encompassing either free VPN apps or ones offering a free trial. They found that 70 % of the VPN apps they tested used HTTP instead of HTTPS for package transmission, and 39 % leaked personally identifiable information (PII), such as usernames, passwords, email addresses, source IP addresses or GPS coordinates. Additionally, while only 44 of the applications could successfully connect to the VPN server and enable encrypted communication, 32 of them leaked DNS queries, primarily by forwarding

these queries to Google's DNS servers. Regarding IPv6 traffic leakage, Wilson et al. note that on iOS, using a tunnelling protocol that does not support IPv6 leads to IPv6 being disabled system-wide. Furthermore, the only means of implementing a VPN protocol supporting IPv6 on iOS is by the use of Internet Key Exchange version 2 (IKEv2): IKEv2 is a key exchange protocol that is part of the IPSec suite, which secures IP communication by providing encryption, integrity verification and authentication (cf. Section 2.3). It is a commonly used building block that supplies the key exchange for authentication within the IPSec suite. The authors found that 17 of the 57 VPN applications that were tested used IPSec with IKEv2, and 15 additionally employed Encapsulating Security Payload (ESP), a state-of-the-art component of the IPSec protocol suite used in conjunction with IKEv2, supplying encryption and authentication of the packets. While this shows that more than half of the VPNs had theoretical support of the IPv6 protocol, the authors did not explicitly mention testing for IPv6 traffic leakage. Nevertheless, they emphasise that only 15 of the tested VPN clients follow the best practice of using IPSec in conjunction with IKEv2 and ESP. They recommend that the other applications update their implementations to adhere to these best practices.

While many publications have highlighted insecure VPN applications, few have succeeded in offering users a secure alternative. However, Karlsson et al. [Kar17] present an implementation attempting to allow secure use of public Wi-Fi networks: They implemented a prototypical device capable of logging into public Wi-Fi networks and subsequently opening a VPN tunnel. The device then allows its user to connect to it via an encrypted hotspot, thereby ensuring that all user traffic is forwarded via the VPN tunnel established on the additional device. On one hand, startup leaks are avoided since the device presumably exposes only its own traffic, which is less sensitive than that of the user device. However, most people are unlikely to carry an additional device for VPN capability with them instead of using a VPN client directly on their mobile device.

4.2 Public Wi-Fi Networks and their Attack Surfaces

A public Wi-Fi network, also called hotspot, denotes an 802.11 Wi-Fi network that is usable by anyone and accepting connections from all clients, as can be seen in Fig. 4.2. A distinction can be made between the underlying technologies used to provide protection mechanisms for the hotspot, namely using WEP, WPA, WPA2 or WPA3. WEP and WPA, introduced in 1997 respective 2003, contain vulnerabilities [TB09] that make their use insecure, exploitable and inadvisable, and are therefore not taken into further consideration in this thesis. Even though WEP and WPA were superseded by WPA2 in 2004, almost 10 % of the networks observed in a 2016

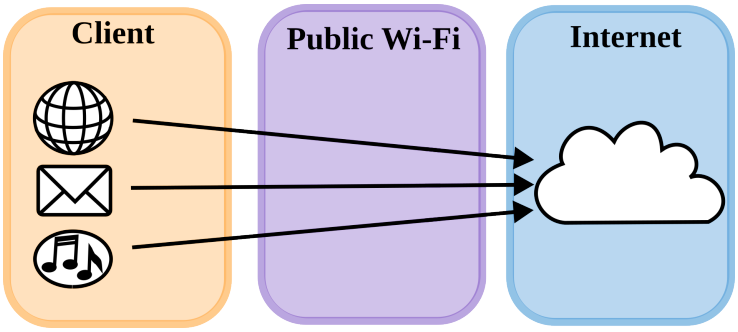


Figure 4.2: Normal operation in a Wi-Fi network: The traffic generated by various applications the client is using are sent via the public Wi-Fi network, each upholding a separate connection visible within the network.

study, covering almost 32 million networks and shown in Fig. 4.3, still used these insecure technologies [Leg16], and they are likely, if less, still in use today. The same 2016 study also exhibited that around 22 % of the networks were *open* networks, meaning that accessing them was possible without entering passwords, and in turn also without the exchange of key material used for subsequent encryption. In such open networks, attackers monitoring the traffic can read the content of unencrypted data packets and the metadata on traffic using transport layer encryption via TLS. As this section will show, most public Wi-Fi networks can be attacked due to their inherent attributes. This also underlines the need to protect oneself further when using a public Wi-Fi network, for example by the use of a VPN. In the following, the different types of public Wi-Fi networks and their attack surfaces are discussed in further detail.

4.2.1 Wi-Fi Protected Access 2 (WPA2)

A popular way of providing a public hotspot is to make a WPA2-protected network accessible via a publicly announced password. Connection establishment in WPA2-protected networks works as shown in Fig. 2.7, with the legacy steps of authentication and association, which are still maintained to allow for compatibility with WEP networks, performed first, and Robust Security Network Association (RSNA) performed subsequently to exchange the key material in a 4-way handshake. Each client connected to the network exchanges a different Pairwise Transient Key (PTK) during RSNA, which is subsequently used to encrypt the traffic. However, WPA2-protected connections can still be monitored by attackers, e.g. if an attacker within the network manages to capture the exchange of the 4-way handshake used to exchange the PTK, they can use the airdecap-ng tool[2] contained within the aircrack-

2. https://www.aircrack-ng.org/doku.php?id=airdecap-ng

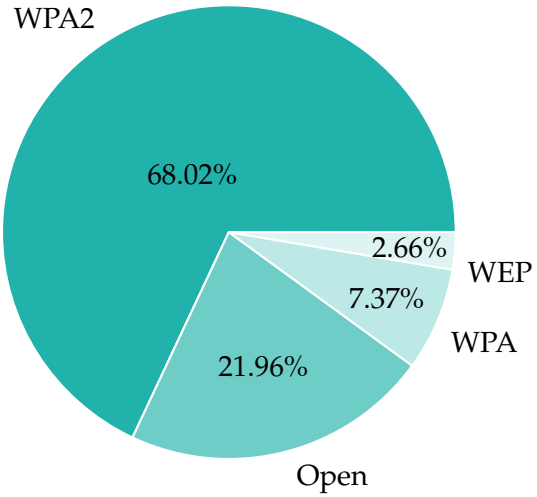


Figure 4.3: Encryption types utilised in public Wi-Fi networks according to a 2016 study [Leg16]. Newer data would include a significant amount of WPA3-protected networks and ones using Opportunistic Wireless Encryption (Opportunistic Wireless Encryption (OWE)), both of which were introduced in 2018.

ng suite[3], a suite of programs usable for attacks on Wi-Fi networks. Airdecap-ng requires at either the packets 2 and 3 or packets 3 and 4 of the 4-way-handshake to derive the PTK, using which the encrypted traffic can be decrypted.

An alternative to sniffing traffic in a public hotspot is to conduct a Man-in-the-Middle (MitM) attack using a fake Access Point (AP) with the same credentials as the public hotspot. If the fake AP has a stronger signal than the real hotspot, connected devices will disconnect from the hotspot and connect instead to the fake AP, on which the attacker can then monitor (and modify) all the traffic. This attack is not only possible in the vicinity of the public hotspot; the attacker can set up the fake AP anywhere, and client devices will assume it is a legitimate network and connect to it automatically, which immensely enlarges the attack surface.

If the attacker has no knowledge of the Pre-Shared Key (PSK) of the network, e.g. the Wi-Fi password, they can use attacks such as the Key Reinstallation AttaCKs (KRACK) attack [VP17]: The KRACK attack forces clients to reinstall keys and reuse previously used nonces by replaying handshake messages. Using the captured handshakes, it is possible for attackers to decrypt and replay packets in case of AES-CCMP encryption. AES-CCMP denotes the use of AES in Counter Mode with the Cipher Block Chaining Message Authentication Code Protocol, which is part of the WPA2 standard and replaces the WPA-TKIP protocol.

3. https://www.aircrack-ng.org/doku.php

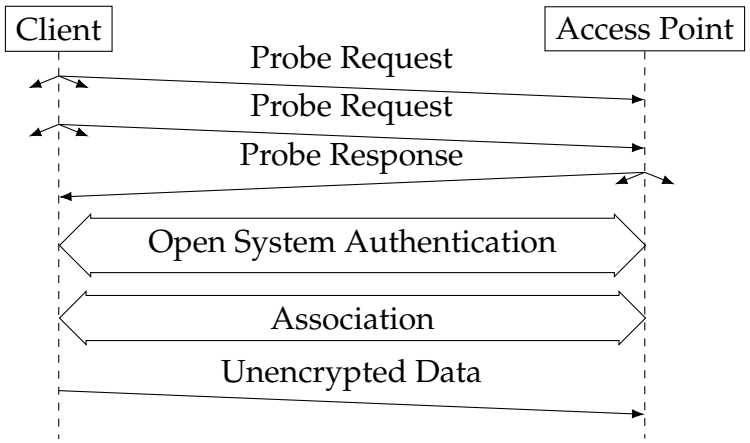


Figure 4.4: IEEE 802.11 connection establishment for open networks. In comparison to the IEEE 802.11 connection establishment for WPA/WPA2/WPA3-protected networks (cf. Fig. 2.7), the authentication step is replaced by Open System Authentication, a dummy step. Additionally, the exchange of key material in the Robust Security Network Association (RSNA) is omitted.

Additionally, it is possible to decrypt, replay and forge packets when WPA-TKIP and GCMP are used: WPA-TKIP denotes the utilisation of WPA using the Temporal Key Integrity Protocol (TKIP), which was replaced by AES-based encryption in WPA2 and WPA3. GCMP, on the other hand, is the Galois/Counter Mode Protocol, a mode that AES can be used in, and which is used in both WPA2 and WPA3. This shows that the KRACK attack can be used to attack both legacy networks, as well as the encryption used in state-of-the-art technologies.

In case an attacker does not have access to the Wi-Fi password of a public WPA2-protected network, which might be the case, for example, if they do not have access to the premises on which the providers of the hotspot show the credentials, they can use tools like hashcat[4] to either brute-force the password, use a dictionary attack to retrieve it or run a rule-based attack.

4.2.2 Open Networks

An alternative to providing a WPA2/WPA3-protected access point is the setup of an open network. Here, an unencrypted hotspot is opened and connecting to it requires no knowledge of a password. The connection establishment is shown in Fig. 4.4: In comparison to WPA2/WPA3-protected networks, it also consists of device discovery via beacons or probe request and probe responses, with the subsequent

4. https://hashcat.net/wiki/doku.php?id=cracking_wpawpa2

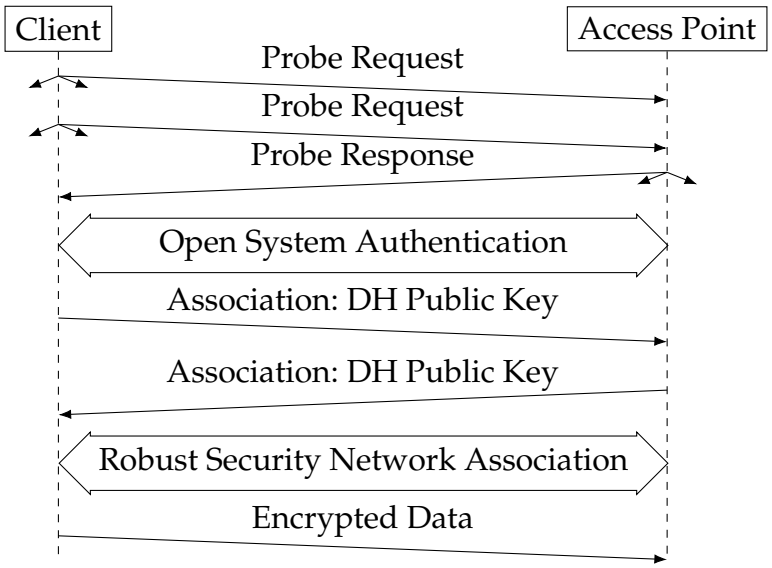


Figure 4.5: IEEE 802.11 connection establishment for OWE networks. In comparison to the IEEE 802.11 connection establishment for WPA/WPA2/WPA3-protected networks (cf. Fig. 2.7), the authentication step is replaced by Open System Authentication, a dummy step. During the 802.11 association, DH public keys are exchanged and then used to create the Pairwise Master Key (PMK). The key material derived in the 4-way handshake during Robust Security Network Association (RSNA) is based on the PMK.

authentication step being replaced by Open System Authentication, a dummy step denoted as a *null authentication algorithm* in the specification [IEE20, Sec. 12.3.3.2.1]. The subsequently exchanged messages are not encrypted on the data link layer.

4.2.3 Opportunistic Wireless Encryption (OWE)

A more secure protocol than both providing an open network and using WPA2 with a publicly advertised key, but unfortunately not yet widely spread, is the use of OWE [All20]. OWE describes the unauthenticated but encrypted connection between two devices, and is contained in the Wi-Fi specification under the name *Wi-Fi CERTIFIED Enhanced Open*™ [Dan]. In OWE, the client sends its Diffie Hellman public key in the 802.11 association request, to which the AP replies with its own Diffie Hellman public key in the association response. Using the previously exchanged public keys, the Diffie Hellman key exchange is completed and the resulting secret value used by both parties to derive a Pairwise Master Key (PMK). This is then used in the 4-way handshake performed in RSNA to derive encryption keys for the encryption of both unicast and broadcast data. [HK17]

OWE protects users from passive attacks, since the calculation of the encryption keys is not based on the known PSK as in WPA2, which enables attackers to calculate the PTK after capturing a handshake. Instead, the calculation of the encryption keys uses the PMK as a basis for key derivation, which is unknown to an attacker. However, this **only ensures protection from a passive attacker** trying to sniff a connection.

An **active attacker**, on the other hand, can perform a **fake AP attack**: They would first issue a deauthenticate-command. The victim's device would terminate it's connection to the OWE-protected network and attempt a reconnect, during which the attacker can open up an AP with a stronger signal and the same SSID. The mobile device will then connect this AP instead.

Just like with WPA2-protected public Wi-Fi networks, while not in the vicinity of an OWE-protected access point, the attacker can still open an access point with the same credentials (or SSID and lack of credentials) as an often-used network, e.g. of a popular coffee place. A device that isn't actively used would connect to it and continue using the network in the background. Since in this case, the attacker participates in the exchange of key material, they are subsequently able to decrypt the encrypted content.

4.2.4 Wi-Fi Protected Access 3 (WPA3)

Just like in WPA2, it is possible to provide a public access point with a publicly announced password, the pre-shared key. In WPA3, however, the PMK from which the encryption keys are derived, is not based on the pre-shared key. Instead, WPA3 uses the Simultaneous Authentication of Equals (SAE): A password-authenticated key exchange [IEE20] based on the Dragonfly handshake [VR20].

Following the 802.11 association and authentication steps required to provide backwards compatibility to WEP, the RSNA is performed. In WPA2, this phase consists of a 4-way handshake to establish key material. In WPA3, RSNA additionally employs SAE for password-authenticated key exchange. While the implementation of SAE in WPA3 was vulnerable to attacks in its original version [VR20] using the Dragonblood attacks[5], the authors provided defence mechanisms which were subsequently incorporated into the standard. In its latest version, WPA3 has not yet been shown to be vulnerable to further attacks that compromise confidentiality using passive attacks. The **feasible attacks are therefore limited to the following *active* attacks**:

5. https://wpa3.mathyvanhoef.com/#tools

- **Downgrade attack**: WPA3-protected networks can be configured to support mixed-mode, meaning primarily WPA3 and additionally WPA2 to allow connectivity for devices supporting only WPA2. In this case, an attacker can force a device attempting to connect using WPA3 to use WPA2 instead and calculate the PTK or use a KRACK attack to decrypt the communication.
- **Fake WPA2-AP attack**: In this attack, an attacker sets up a fake AP offering WPA2 protection, in which the previously mentioned attacks in the context of WPA2 can be used to passively monitor the encrypted content,
- **Fake WPA3-AP attack**: By setting up a fake WPA3 protected network, an attacker can actively control the key exchange and therefore decrypt the content of the subsequent encrypted communication.

The passive PTK derivation attacks which were possible in the context of WPA2 are, however, largely infeasible due to the security guarantees SAE offers.

In summary, this section shows the susceptibility to attacks on public Wi-Fi networks depending on the different protection mechanisms: In an open network, all traffic is transmitted without encryption and monitoring the content is trivial. For WPA2 protected networks, passive attacks allow attackers to decrypt encrypted traffic. Both OWE and WPA3-protected access points can be attacked via active attacks. This shows that to maintain confidentiality and integrity while using a public access point, an additional layer of protection is required, and the security guarantees a VPN can provide meet those needs. But as Section 4.1 underlines, even a protective layer like a VPN can leak confidential information outside the tunnel by failing to forward certain protocols inside the tunnel. Another critical point regarding the use of VPNs is the process of bootstrapping the VPN, meaning the phase of preparation and initialisation of the tunnel until its successful establishment. In the subsequent section, first an attacker model is introduced, and then requirements for secure bootstrapping to guarantee a leakage-free VPN connection establishment are described.

4.3 Attacker Model

The attacker considered in this scenario has **access to a public Wi-Fi network**, and, depending on the protection in place, can **read or decrypt** the content. This is the case for all protection schemes preceding and including WPA2, but due to the attacker only having **bounded computational resources** infeasible for WPA3 and OWE. Their objective is to **monitor the content and metadata of the traffic** and to infer as much information on the content of encrypted traffic as possible. For the

purposes of monitoring the traffic, the attacker employs Commercial Off-The-Shelf (COTS) equipment, which **does not require great financial means**. It is possible for the attacker to perform their analysis in **real time**. However, attacks on the VPN infrastructure are out of scope for this attacker.

4.4 Requirements for Secure Bootstrapping of a VPN

The requirements introduced in this section can help to ensure that VPN connections are established in a privacy-preserving and secure way. They were previously proposed in [Bur+21] and are presented here in the context of this thesis. They encompass the following requirements:

R1: Always-On Functionality. The first requirement asserts that upon establishing a network connection, a VPN tunnel is directly established. If this requirement is not enabled by design, it must at least be possible for the user to activate the functionality prior to establishing a network connection, so as to ensure no traffic is leaked before the VPN tunnel is established.

R2: Captive Network Support. The requirement for captive network support ensures that connection establishment is possible while using a VPN, even if a captive portal is contained within the network. The VPN client must support both captive portal detection and remediation of the block. A VPN client not supporting captive portal detection would otherwise cause a captive deadlock in conjunction with the use of a VPN.

R3: Minimal Startup Traffic. The third requirement for secure bootstrapping is the minimisation of startup traffic. This requirement asserts that traffic that is unnecessary for captive portal remediation or VPN establishment is held back until both the captive portal is remediated and the VPN tunnel established.

R4: Blocking Fail State. In case a VPN tunnel can not be established successfully, e.g. due to VPN server unavailability, outbound traffic remains blocked.

R5: No Tunnel Bypass. The last requirement specifies that no traffic bypasses the tunnel. Particularly Section 4.1 showed that a significant number of VPNs tested in prior work exhibited DNS- and IPv6 leakage. Additionally, implementation flaws can cause previously established TCP connections to be forwarded outside the tunnel if they were in use prior to VPN establishment [Pro20]. The only traffic exempt from this requirement are periodic requests regarding captive portal detection, since these are required to maintain the connection.

	macOS/iOS	Windows	Android	Ubuntu
R1: Always-on	✓	✓	✓	✓
R2: CPD	-	-	-	-
R3: Minimal Traffic	✓	-	-	-
R4: Blocking	-	✓	✓	-
R5: No Bypass	(✓)	✓	✓	-

Table 4.1: An overview over the intrinsic platform capabilities of VPNs as mentioned in the documentation. While all platforms provide means of implementing an always-on VPN (R1), captive portal remediation (R2) is not mentioned in the documentation of any platform. The table reflects the intrinsic capabilities as mentioned in the documentation, but other requirements might be satisfied programmatically. With respect to R4 and R5, VPNs under Windows and Android *can* be implemented to support fail-state blocking and to disallow tunnel bypassing, but are not necessarily set to implement this behaviour by default. Under macOS/iOS, requirement R5 is satisfied in case a device under supervision utilises an always-on VPN. Since device supervision is typically only used within companies, the feature is unlikely to be natively used within apps for leisure use, unless implemented manually by the developers.

Together, these requirements ensure that no traffic is leaked, neither during the establishment of the VPN connection nor during its use. Because of the requirement to only employ minimal startup traffic, the possibility of leakage is further reduced. Additionally, the use of captive networks remains possible and the process of their remediation free of further leakage. Lastly, the requirement to construct a blocking fail state ensures that an unsuccessful VPN tunnel establishment does not result in unexpected traffic leakage.

4.5 VPN APIs: Status Quo

In order to understand the functionality of system APIs providing VPN services on major platforms, this section provides an overview over the status quo as previously published in [Bur+21], and presented here in the context of this dissertation. All findings are visually summarised in Table 4.1.

4.5.1 Apple macOS and iOS

Within the framework providing network extensions, Apple provides two APIs that allow integration of VPN functionality into apps. The first is the **Personal VPN**

[Appd], using which a VPN connection on the basis of IPSec can be established. For the key exchange, either Internet Key Exchange version 1 (IKEv1) or version 2 (IKEv2) can be used, both of which are part of the IPSec protocol suite (cf. Section 2.3). The Personal VPN API allows developers to make use of on-demand capabilities, which trigger when certain conditions are met [Appe]. An example could be to commence the VPN upon connection establishment to a Wi-Fi network and terminate it upon the switch to mobile data.

The second VPN API Apple provides is the **Packet tunnel provider** [Appc]. Here, developers can implement a custom VPN protocol. Both approaches differ with respect to the application entitlements that are required in order to perform respective API calls. Entitlements are privileges within the system that grant apps the rights to execute particular capabilities [Appa]. Network Extensions Entitlements in particular are a set of APIs that can be used to customise the networking features of an app [Appb].

With respect to the **requirements for secure bootstrappin**g of VPNs specified in Section 4.4, the API provides developers with the possibility to offer always-on functionality, which satisfies the requirement R1. The always-on functionality can be implemented via on-demand rules. These can be configured to trigger upon certain events, for example upon establishing a Wi-Fi connection [Appe]. These on-demand connection rules additionally satisfy requirement R3, since they block outgoing traffic prior to successful VPN tunnel establishment. Organisations can choose to provide devices under device supervision with Always On VPNs. These also support separate tunnels per interface, with a distinct VPN tunnel for each active interface capable of transmitting IP traffic. In this case, requirement R5 is additionally satisfied, since all traffic outside the tunnel is continuously blocked [Appf]. However, since this feature requires device supervision, it is likely not used in many VPN implementations for leisure use.

4.5.1.1 Distribution of Entitlements in macOS apps

In their 2019 publication [Blo+19], Blochberger et al. presented an analysis of Apple's sandboxing mechanism and its adoption in macOS apps. By crawling the Mac App Store daily for 11 months, they created a snapshot of all available apps and their updates. In total, the snapshot comprises 8366 macOS applications. These served as a basis to evaluate the distribution of use of the distinct VPN API entitlements in macOS apps in the context of [Bur+21]: 92 of the apps used the **Personal VPN entitlement** `.vpn.api`, and 75 used the **packet tunnel provider entitlement**, which is enabled by setting the `.networkextension` to `packet-tunnel-provider`. It is additionally possible to provide per-app VPN tunnels via an app-proxy provider. Since

the research for [Bur+21] focussed on system-wide tunnels, app-proxy providers were out of scope of the analysed VPN APIs. Nevertheless, in the analysis mentioned above, one single app was found to use the **app-proxy provider entitlement**, which is enabled by setting the `.networkextension` to `app-proxy-provider`. This shows that among app manufacturers, there does not seem to be a clear preference for either the Personal VPN or the Packet tunnel provider entitlement, but rather that both are considered valid implementation options.

4.5.2 Android

VPN apps under Android require the use of both the system API, as well as the `BIND_VPN_SERVICE` permission. Permissions within the Android system are comparable to *entitlements* in the Apple system, prompting users for explicit consent to the use of certain execution capabilities. VPN apps under Android can be implemented to either start upon user request, or to be started by the system, which can be done via the *always-on* feature. The use of the latter satisfies requirement R1. To block traffic forwarding outside the VPN tunnel, users can manually switch on the *Block connections without VPN* option via their settings, the use of which would satisfy requirement R4. However, since this is a user setting, app providers can not specify it themselves. Users can additionally set per-app VPN settings via an allowed list or a disallowed list, in which they can specify the apps whose traffic is either forwarded via the VPN or outside the VPN tunnel. The traffic of apps that are not in the lists is blocked by default. Developers have similar control via the per-app VPN settings. However, the documentation states that a missing allowed or disallowed list or an empty allowed list causes all network traffic to be sent via the VPN, and additionally, an always-on VPN can be set to block all connections not using the VPN both of which satisfy requirement R5 [And].

4.5.3 Windows 10

Windows 10 is an operating system developed by Microsoft and released in 2015. It was superseded by Windows 11 in 2021 and remains in support until the end of 2025 [Kar23], with extended security update support ending in 2028 [Ste23]. During the time of writing the publication on VPN leakage across various platforms in 2020 [Bur+21], it was the latest available Microsoft operating system, and it remains supported during the writing of this thesis, which is why it is introduced here despite a later release – Windows 11 – being available as of 2024. Additionally, as Fig. 4.6 shows, Windows 10 remains in considerable use as of today, with a market share encompassing more than 68 % of the devices [sta24b].

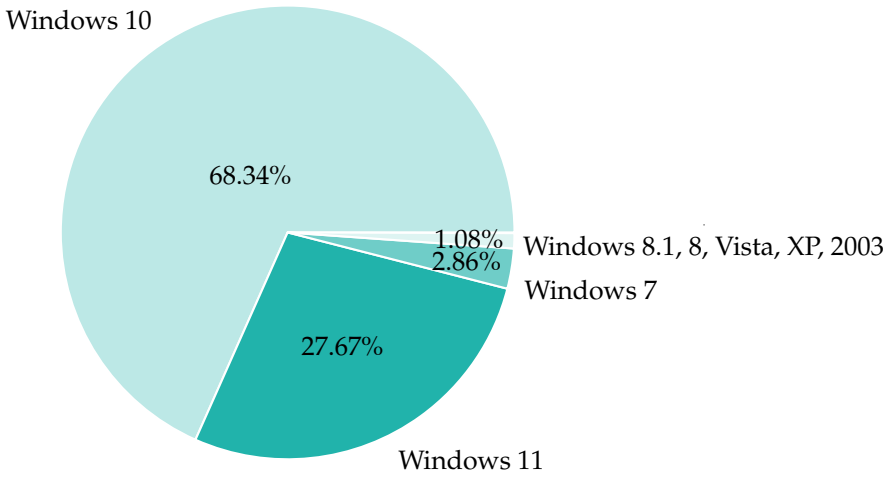


Figure 4.6: Market share of Windows operating system versions as of May 2024. The figure was created using data from [sta24b]

The native VPN client on Windows 10 can be set to support several of the requirements introduced in Section 4.4: Besides triggering the VPN connection establishment upon opening an application or upon visiting a specific domain, a VPN can be set to always-on, the latter satisfying requirement R1. The always-on feature can be triggered upon sign-in to the device, upon network change or whenever the screen is turned on. [Mata]

When using the native Windows VPN, it is possible for administrators to configure a VPN profile that enforces the use of the VPN to the point where users are unable to turn it off. This satisfies requirements R1, R4 and R5, since the outbound traffic is blocked completely in case of VPN connection unavailability, the VPN is set to always on and all traffic *has* to be transmitted via the tunnel [Matb]. However, this profile setting can only be configured by administrators and can *not* be disabled by non-admin users.

It is possible for Windows developers to develop VPN applications using the `IVpnProfile` Interface[6] contained within the Windows Runtime API. All settings mentioned above can be integrated into those apps. Additionally, mechanisms for mobile device management can be utilised to enable clients to remote-join to a domain [Ser23].

6. https://learn.microsoft.com/en-us/uwp/api/windows.networking.vpn.ivpnprofile?view=winrt-19041

4.5.4 Ubuntu GNU/Linux

In the GNU/Linux environment, the landscape of distributions is very diverse. Therefore, the analysis was focussed on Ubuntu, a popular desktop distribution. Ubuntu utilises NetworkManager[7], a network configuration tool suite and high-level daemon providing networking that can also provide VPNs. NetworkManager provides an API that can be used to offer VPN services via libnm[8] and DBus[9]. If a VPN service is declared persistent [Man23], it will attempt connection maintenance across link changes and connection disruptions, in which case the VPN would fulfil requirement R1. It is additionally possible when using NetworkManager to implement VPNs as so-called *secondaries*[10]: In this case, a VPN is automatically activated upon connection establishment via another connection.

In summary, the documentation of native VPN implementations provides detailed insight into several areas. It showed, for example, that all platform providers supply the means to implement an Always-On VPN. However, the documentation is thoroughly lacking with respect to captive portal detection, which is completely omitted, but an important requirement for the use of VPNs in public Wi-Fi networks. The subsequent section therefore provides insight into the captive portal detection mechanisms of the native clients via an experimental analysis.

4.6 Captive Portal Detection in VPNs - An Experimental Analysis

In the previous sections, the dangers of using public Wi-Fi networks were shown, and the need for the use of VPNs in such vulnerable surroundings emphasised. Since traffic leakage occurring despite the use of a VPN would decrease privacy guarantees and user trust, VPNs should fulfil certain requirements as summarised in Section 4.4: They should be set to always-on (R1), allow for captive portal detection (R2), and reduce the traffic during connection establishment to the minimum to block all traffic not required for VPN tunnel establishment (R3). Additionally, they should contain a blocking fail-state, meaning that upon failed VPN tunnel establishment, traffic continues to be blocked (R4). Additionally, all traffic should be routed via the VPN and no traffic should bypass it (R5). The previous Section 4.5 then examined

7. https://networkmanager.dev/
8. https://networkmanager.dev/docs/libnm/latest/
9. https://networkmanager.dev/docs/api/latest/
10. https://networkmanager.dev/docs/libnm/latest/NMSettingConnection.html#nm-setting-connection-add-secondary

the status quo of system VPNs and VPN APIs, and summarised the findings in Table 4.1.

The requirement R2 to fulfil captive portal detection is essential to both VPN establishment as well as captive portal remediation: VPNs attempt to block all traffic except that required for VPN establishment, and simultaneously, a connection to the Wi-Fi network can not be established without captive portal remediation. If the traffic to the captive portal is therefore also blocked, captive portal remediation is impossible, resulting in a captive deadlock.

The previous section showed that no mechanisms for captive portal detection are documented for either of the platforms. This section therefore aspires to document the existence of captive portal detection mechanisms both in native VPN applications as well as VPN apps. This is done by analysing the mechanisms from within a testbed as described in Section 4.6.1. The section additionally describes leakage classification and specifies the test procedure. Subsequently, the captive portal detection mechanisms inherent to the different platforms are explored in Section 4.6.2. Section 4.6.3 then analyses whether the native VPN clients fulfil the requirements for secure bootstrapping. As the tests will show, albeit the Apple VPN API can *theoretically* be used to provide an always-on VPN, their native VPN clients do not provide these capabilities in practice. Therefore, to establish whether the requirement *can* be met in practice, a VPN API demo is implemented and tested in Section 4.6.4. Section 4.6.5 then shows the results of the tests of third-party (commercial) VPN clients.

4.6.1 Testbed Setup and Test Procedure

The testbed used to analyse captive portal detection mechanisms comprises a Raspberry Pi 3 Model B+, running Raspbian GNU/Linux 10. The Raspberry Pi provides an access point via hostapd[11] 2.2.7, a user space daemon that implements IEEE 802.11 access point management and authentication servers. The configuration used for hostapd can be seen in Listing 4.1: It provides an access point via its interface wlan0, with the SSID FREEWIFI. Lines 10 and 11 show that the use of WPA protection and a password are commented out, making it in essence an open, unencrypted access point as often found in public places.

The captive portal is provided via Nodogsplash[12] 4.5.1 beta. Nodogsplash is an open source project providing a captive portal and restricting internet access prior to captive portal remediation. It is set to redirect plain HTTP request via a 307 status

11. https://w1.fi/hostapd/
12. https://github.com/nodogsplash/nodogsplash

```
1  interface=wlan0
2  driver=nl80211
3  ssid=FREEWIFI
4  hw_mode=g
5  channel=7
6  wmm_enabled=0
7  macaddr_acl=0
8  auth_algs=1
9  ignore_broadcast_ssid=0
10 #wpa=2
11 #wpa_passphrase=insecure_password
```

Listing 4.1: The hostapd configuration file to set up an open access point with the SSID FREEWIFI. Note that both variables `wpa` and `wpa_passphrase` in lines 10 and 11 are commented out, resulting in an open access point without WPA2 protection.

code, which provides a temporary redirect. The redirect forwards the user to the captive portal page running on the Raspberry Pi. By clicking a *continue*-button on the sign-in page provided, users gain internet access. The traffic is captured on the Raspberry Pi using Wireshark/tshark.

The devices used for testing VPN behaviour encompass the following:

- Google Nexus 5X, running an Android 10.0 custom ROM (PixelExperience ROM version 10.0-20200912-1735),
- An iPad running iOS 13.7,
- A MacBook Pro running macOS 10.15.7,
- A Dell laptop running Ubuntu 20.04 LTS and NetworkManager version 1.22.10,
- The same Dell laptop running Windows Education 10.19041.

4.6.1.1 Leakage Classification

In the experiment, **all outgoing traffic *not* required for VPN setup is considered leakage**. However, the following protocols are exempt from this leakage classification:

- Address Resolution Protocol (ARP),
- Dynamic Host Configuration Protocol (DHCP),
- Extensible Authentication Protocol over LANs (EAPOL),

- Internet Control Message Protocol (ICMP),
- Link-Local Multicast Name Resolution (LLMNR),
- Internet Group Management Protocol (IGMP) and
- Multicast DNS (mDNS).

Packets sent via these protocols are typically used for control and configuration within the local network, exchange of communication parameters or diagnostic purposes. Requiring for them to be tunnelled via VPN would defeat their purpose and could negatively influence the network connection.

4.6.1.2 Test Procedure

Each VPN client is tested in an unrestricted network, in the following called **open mode**, and a captive network, denoted **captive mode**. To test the behaviour in case of a network block, VPN clients are additionally tested in **block mode**, where the network is configured to drop the traffic destined for the respective VPN endpoint. Tests in block mode are necessary to test the requirement R4 of blocking in fail state. This mode also increases the potential to spot race conditions during VPN startup. A client not offering to auto-connect is instead manually activated before joining the respective test network.

For each VPN client, at least **three tests are performed per mode**. The test procedure encompasses the following steps:

1. Disconnect the Wi-Fi connection on the client device,
2. Activate the VPN client (if the VPN client does not support always-on),
3. Clear the captive portal remediation state,
4. Start the Wi-Fi capture,
5. Connect the client device to the Wi-Fi network,
6. Complete the Captive Portal sign-in in case the VPN employs Captive Portal Detection (CPD),
7. Capture the traffic for 20 seconds.

The captured traffic is subsequently classified according to the leakage definition presented previously.

	macOS	iOS	Windows	Android	Ubuntu
System employs CPD	✓	✓	✓	✓	✓
Blocking of platform traffic	✓	✗	✗	✗	✗
Blocking of third-party traffic	✓	✓	✗	✓	✗

Table 4.2: An overview over the platform behaviour during the remediation of a captive network. Source: [Bur+21].

4.6.2 Inherent Captive Portal Detection Mechanisms

Since captive portal remediation during the use of a VPN client can only be successfully performed by the system in case the device has mechanisms for captive portal detection, a test to establish a baseline for each platform is required. The test are conducted in two scenarios, with the first one (a) successfully completing captive portal sign-in, and the second one (b) omitting captive portal sign-in, resulting in the client remaining captured. The results are shown in Table 4.2: All platforms employ captive portal detection and prompt their users to fulfil the terms for captive portal remediation. **MacOS and iOS** perform captive portal detection before allowing the rest of the system to utilise Wi-Fi connectivity. Unless the captive portal is remediated, all third-party traffic remains blocked. This applies both to scenarios (a) and (b). In scenario (b), however, DNS queries originating from the operating system but unrelated to captive portal remediation are leaked on iOS, followed by outgoing traffic towards those hosts. **Android**, on the other hand, leaks DNS lookups and TCP traffic to platform services unrelated to captive portal detection prior to remediation. However, third-party traffic remains blocked on Android. On both **Windows** and **Ubuntu**, captive portal detection blocks neither platform nor third-party traffic.

4.6.3 Analysis on Native VPN Clients

All operating systems tested in this analysis are equipped with built-in clients for basic VPN functionality. To gauge whether the native VPN clients fulfil the requirements summarised in Section 4.4, they were tested and analysed. However, only the functionality exposed via the OS's GUI, for example via the dialogue used to enter network settings, was analysed, not additional functionality only exposed by use of APIs (cf. Section 4.5) or special configuration files. The results of the tests are summarised in the following and can also be observed in Table 4.3.

Platform	Client	R1: Always-on	R2: CPD	R3: Minimal	R4: Blocking	R5: No Bypass
macOS	Native	✗	–	–	–	–
	Demo	✓	✓	✗	✗	✗
	EncryptMe	✓	✓	✗	✗	✗
	ExpressVPN	✓	✗	(✓)	✓	?
	Mullvad	✓	✗	✓	✓	✓
	ProtonVPN	✓	✓	✗	✓	✗
iOS	Native	✗	–	–	–	–
	EncryptMe	✓	✓	✳	✓	✗
	ExpressVPN	✓	✓	✳	?	✗
	Mullvad	✓	✓	✓	✓	✓
	ProtonVPN	✓	✓	✳	✳	✗
Windows	Native	✗	(✗)	–	–	–
	EncryptMe	✓	✓	✗	✗	✗
	ExpressVPN	✓	✓	✗	✗	✓
	Mullvad	✓	✗	✓	✓	✓
	ProtonVPN	✓	✗	✓	✓	✓
Android	Native	✓	✗	✓	✓	✓
	EncryptMe	✓	➠	✗	✗	✗
	ExpressVPN	✓	✗	✳	✓	✓
	Mullvad	✓	✗	✳	✓	✓
	ProtonVPN	✓	➠	✗	✗	✳
	ProtonVPN (always-on)	✓	✗	✓	✓	–
Ubuntu	Native	✓	✗	✗	✗	✓
	ExpressVPN	✓	✗	✗	?	✓
	Mullvad	✓	✗	✓	✓	✓

Table 4.3: Overview of the analysis of various VPN clients. The symbols used in addition to ✓and ✗ imply a race condition (➠), traffic leak caused by the platform (✳), features that were not testable (**?**), and features that were inapplicable (–). Tests for Ubuntu could only be conducted using the native client, ExpressVPN and Mullvad as the other providers either do not offer Linux clients, or provide clients that did not work. Adapted from: [Bur+21].

4.6.3.1 macOS and iOS

Even though the system API provides the means of implementing VPNs with **always-on capability**, the native VPN clients on macOS version 10.15.7 and iOS version 14.0.1 contain no means of using it. On both platforms, it is **not possible to start a**

VPN prior to connection establishment to a Wi-Fi network; VPNs *have* to be started manually. Additionally, in case of an interrupted network connection, the VPN clients **do not always automatically reconnect** once the network connection is restored. While iOS supports the remote deployment of always-on VPN provides, this is only possible via device supervision, for example when using mobile device management [Appf]. Since the devices used in this example did not fulfil the prerequisites required for device supervision, an **always-on VPN option could not be tested**.

4.6.3.2 Windows

Windows 10 has a graphical built-in VPN configurator, which **does not** support the configuration of an **always-on VPN profile**. Nevertheless, network interfaces going offline do not immediately cause VPN disconnections. This has an impact on captive portal detection, since reconnecting after a short disconnect causes for **captive portal detection to be suppressed until the VPN timeout**, resulting in a captive deadlock. VPN timeouts, in turn, result in **unrestricted outbound traffic outside the VPN tunnel**. Additionally, tests in open mode and in block mode showed that **DNS lookups to Microsoft hosts bypass the VPN tunnel**, querying addresses like `www.bing.com`.

4.6.3.3 Android

The VPN client natively provided on Android devices [And] allows its users to set the native VPN into **always-on mode.** This requires for the VPN server address to be provided numerically and for a DNS server to be set. When using always-on, the Android device enters a captive deadlock: Since the **native VPN does not perform captive portal detection**, the captive portal can not be remediated. The issue has been reported to Google[13]. In the issue tracker, it was first pronounced to be a product feature issue and not a developer issue, and then marked a duplicate of another issue, access to which is denied for normal users.

In the following tests, the captive portal was disabled, which resulted in the native Android VPN client being able to establish a VPN connection. Apart from the traffic required for CPD remediation, **TLS traffic destined to `www.google.com` was transmitted before tunnel establishment**. Apart from this, no other traffic was leaked.

13. https://issuetracker.google.com/issues/170461560

4.6.3.4 Ubuntu GNU/Linux

The native VPN application provided on Ubuntu is NetworkManager, which is the networking client with additional in-built VPN functionality. With a VPN profile configured, Network Manager allows its users to employ **always-on functionality for specific connections**, e.g in conjunction with a certain SSID. The in-built VPN client has **no support for captive portal detection**, and connection establishment in a captive network therefore results in a captive deadlock. In open mode, however, the VPN client can be used as expected. Upon receiving ICMP notifications on VPN destination unreachability, **internet connectivity is set to offline**, preventing additional leaks. However, during the use of the auto-connect functionality that is in place before receiving such an ICMP notification, NetworkManager employs **neither fail-state blocking nor minimisation of outbound traffic**, during both VPN establishment and in case of VPN establishment failure. Therefore, during the time NetworkManager attempts to establish a connection, **traffic is leaked outside the VPN tunnel**. In block mode, ICMP notifications are dropped in conjunction with the remaining VPN traffic. NetworkManager therefore retains connectivity until receiving a VPN time-out, thereby increasing the time during which other processes leak traffic.

4.6.4 VPN API Demo

As shown in Section 4.5, dedicated APIs are provided for macOS and iOS to support traffic blocking during VPN connection establishment. The native iOS and macOS VPN clients do not integrate the blocking functionality. A custom API demo for macOS was therefore implemented in [Bur+21] to assert the documented properties. It uses the Personal VPN API introduced in Section 4.5. The service registers an auto-connecting on-demand VPN via the `NEOnDemandRuleConnect` rule whenever the device is connected to a Wi-Fi network.

Tests of the demo showed that the VPN tunnel is started reliably upon connection establishment to the test Wi-Fi. Traffic captures recorded during the tests exhibited **platform traffic towards Apple hosts and third-party traffic occurring between captive portal remediation and tunnel establishment**, both in open and in captive mode. The `NEOnDemandRuleConnect` rule therefore appears to have a less strict blocking feature than the documentation suggests.

Another finding made while monitoring the traffic with an established VPN tunnel was that **TCP streams originating from before the establishment of the VPN connection bypass the VPN tunnel**. This behaviour has been previously reported

[Pro20], and remained unfixed at the time of the analysis performed for the paper [Bur+21], and furthermore still existed in iOS 16, which was released in 2022. The behaviour also applies to TCP re-transmission attempts initiated before VPN establishment. In block mode, no fail-state blocking could be observed: Both platform traffic and third-party traffic commenced upon VPN connection establishment failure.

Result: The demo testing the implementation of the on-demand connection rule showed that it appears to be **insufficient to ensure fail-state blocking.**

4.6.5 Third-Party VPN Clients

In addition to the inspection of native VPN clients, several third-party clients were analysed with respect to the fulfilment of the requirements summarised in Section 4.4: ExpressVPN, EncryptMe, ProtonVPN and Mullvad VPN. ExpressVPN was chosen since it is the market leader among the VPN providers, and EncryptMe since it was the benchmark used by [Kar17]. ProtonVPN is another market leader, additionally providing open source clients and publishing research into leak detection and prevention [Pro20]. Mullvad VPN is another open source VPN provider. The results of all tests can also be observed in Table 4.3.

4.6.5.1 ExpressVPN

When using the ExpressVPN client version 3.0.2.12 on **Ubuntu**, it deadlocked in captive mode. In open mode, no leaks were observed besides local traffic. Tests in block mode were impractical because of unpredictable address switches on the endpoint. Furthermore, upon entering a blocking state in which the app is unable to connect to the VPN server, re-establishing the VPN connection could only be done with full leakage.

On **iOS**, version 8.3.5 of the ExpressVPN app was tested. It uses the on-demand API described in Section 4.5 to automatically re-establish the VPN tunnel if activated. In these tests, platform traffic not required for captive portal detection or remediation was observed both in open and in captive mode. Just as observed in Section 4.6.4, resets of TCP connections transmitting platform traffic, which had been established prior to VPN tunnel establishment, could be observed. Just like on Ubuntu, block testing had to be skipped. Additionally, the iOS client provided by ExpressVPN was capable of captive portal detection.

Subsequent testing of the **macOS** client version 7.11.6(6) showed that it was not using Apple's on-demand API, according to the network settings. In captive mode, the client causes a captive deadlock since it interrupts captive portal detection. However, no leakage could be observed during the captive deadlock. In open mode, packets for captive portal detection could be observed, but the subsequent tunnel establishment repeatedly failed to complete within the 20 seconds determined as capture time, but again, without leakage. The ExpressVPN client under macOS additionally fulfilled the requirement of fail-state blocking of traffic when tested in block mode.

A test of the **Windows** client version 9.1.0(258) in both open and captive mode exhibited leakage of platform traffic unrelated to captive portal detection, as well as third-party traffic, during both the captive portal remediation phase and during tunnel establishment. When testing in block mode, the client exhibited fail-state blocking, showing no additional occurrence of leakage. However, the client does not appear to sufficiently block leaks occurring during startup.

The **Android** client version 9.0.40 caused a captive deadlock when tested in captive mode. Except for platform traffic to `www.google.com`, no other leaks originating from third parties could be observed. The same leaks were observed in open mode, and captive portal detection caused no deadlocks. When tested in block mode, no additional leaks were exhibited.

4.6.5.2 EncryptMe

EncryptMe offers VPN-activation dependent on network trustworthiness. Client version 4.2.3 was tested with activated auto-start on **macOS**. EncryptMe additionally contains a feature called OverCloak[14], which provides leak protection in untrusted networks. This includes connection lock-down prior to VPN tunnel establishment, allowing only critical DNS, SSH and HTTPS packets to be transmitted. The client allows packets required for captive portal detection and successful remediation to pass. However, in both captive, open and block mode, traffic originating from the platform and third-parties is leaked prior to connection establishment and afterwards.

On **iOS**, version 4.4.4 was analysed. On this client, it is possible to enable a setting called auto-protect, which makes use of the on-demand functionality iOS provides. Both in open and captive mode, platform traffic unrelated to captive portal remediation could be observed. In block mode, no additional leakage occurred.

The **Windows** client tested was version 1.1.0. Captive mode exhibited platform and third-party leaks but no deadlocks. Open mode allowed the client to complete VPN

14. https://blog.encrypt.me/2019/09/26/what-is-encryptme-overcloak/

tunnel establishment faster, therefore fewer leaks were found, originating only from platform traffic. In block mode, both platform and third-party traffic was observed again.

The tests of the **Android** client version 4.2.0.1.81964 indicate a race condition occurring between captive portal detection and VPN tunnel establishment: both indeterministic captive deadlocks and leaks of third-party traffic could be observed. In captive mode, captive portal remediation is impossible due to the occurrence of captive deadlocks, which are, however, leakage free. Tests in open mode exhibited no leakage, however, block mode caused for third-party traffic to be leaked.

Tests of a **Linux** version could not be conducted since EncryptMe has no Linux client.

4.6.5.3 Mullvad VPN

The Mullvad VPN client for **iOS**, version 2020.4, first attempts VPN connection establishment, upon failure performs captive portal detection, and upon successful captive portal remediation performs leak-free VPN connection establishment. Both open mode and block mode allowed for leakage free connection establishment. An inspection of the source code confirms that the on-demand VPN API is used by the iOS app.

The clients available for **macOS, Windows and Ubuntu**, version 2020.5, caused captive deadlocks due to blockage of packets related to captive portal detection. Both in open mode and block mode, connection establishment was possible and no leaks were recorded.

The **Android** client version 2020.6-beta2 was released in beta version. Tests in captive mode showed that packets for captive portal detection are transmitted and redirected. However, the request to the captive portal was lost in all tests, thereby causing a captive deadlock.

4.6.5.4 ProtonVPN

The **iOS** client version 2.2.4 offered by ProtonVPN provides always-on functionality, which can not be deactivated. DNS and IP packets originating from the platform and unrelated to captive portal detection could be observed between the captive portal remediation and the establishment of the VPN tunnel. Additionally, traffic addressed to the global IP address that the testbed gateway has could be found throughout the

capture. Tests in block mode revealed the same behaviour, with additional leakage observed towards to Akamai servers, presumably servicing Apple.

When tested in open and captive mode, the **macOS** client ProtonVPN 1.7.2 exhibited neither leaks during captive portal detection nor during remediation. However, after successful remediation and before tunnel establishment, both platform traffic unrelated to captive portal remediation, as well as third-party traffic encompassing DNS and IP traffic could be observed. TCP streams encompassing platform and third-party traffic originating from before tunnel establishment, as well as local traffic is leaked outside the VPN tunnel. Additionally, reverse DNS lookups to the local IP address of the test client could be observed. The same leakage as previously stated could be observed in block mode. Here, however, no further leaks occurred after unsuccessful tunnel establishment.

The **Windows** client version 1.17.3 does not feature an auto-connect setting, and has to be started manually prior to Wi-Fi connection establishment. In a captive network, the Windows client causes a captive deadlock, as packets required for captive portal detection are blocked. When tested in block mode and open mode, no leakage was observed apart from traffic to `api.protonvpn.ch`.

Android version 2.3.54.0 exhibited inconsistent behaviour when tested in captive mode: Either the request to the captive portal is suppressed, causing a captive deadlock, or it is transmitted prior to tunnel establishment, as well as other platform traffic. This indicates a race condition occurring between VPN handling and captive portal detection. In block mode, the Android client exhibits increased leaks of platform and third-party traffic in comparison to captive mode. In open mode, platform traffic is leaked throughout the whole VPN connection establishment.

In addition to the VPN functionality provided by the ProtonVPN Android app, the settings provide instruction for the activation of the **Always-On** functionality as can be seen in Fig. 4.7. Since this functionality extension via the system settings was described in the app, it was also included in the tests. By manually enabling *Always-On VPN* in the android settings and additionally enabling the kill-switch to block all communication outside the VPN tunnel, system functionality can be used to enhance the inherent VPN properties. Tests comparing ProtonVPN with and without Always-On enabled as shown in Table 4.3 indicate that the fulfilment of traffic minimisation and fail-state blocking, which are not met by ProtonVPN, are in fact met if used with Always-On turned on in the Android system. However, while captive portal detection was faced by the race condition described above without Always-On enabled, using Always-On results in complete blocking of packets required for captive portal detection and therefore a captive deadlock.

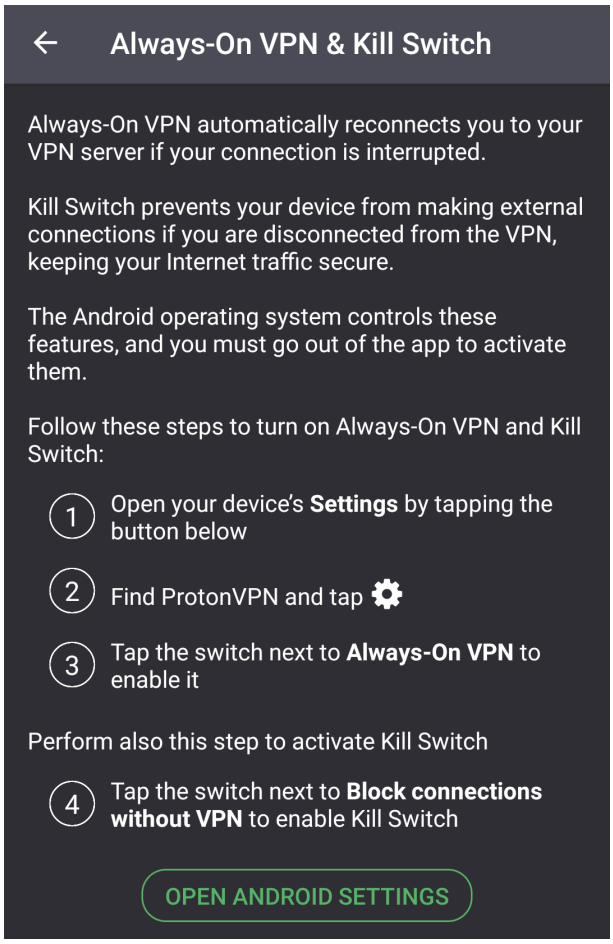


Figure 4.7: Screenshot of the steps described in the Android ProtonVPN client to turn on Always-On functionality. While the ProtonVPN app has no features for traffic minimisation and fail-state blocking, when setting the Always-On features, both traffic minimisation and fail-state blocking are in place. However, without Always-On, the app exhibits a race condition during captive portal detection, and with Always-On, the use of a captive network results in a deadlock.

While ProtonVPN offers the use of a command-line client for **Linux** devices, testing its behaviour was impossible since the credentials were continuously rejected by the system.

4.6.6 Summary

A VPN used in a captive network should fulfil certain requirements to ensure both functionality as well as leakage suppression. The leakage suppression requirements

entail for the VPN functionality to be always-on (R1), for traffic prior to VPN establishment to be minimised (R3), for the application to entail fail-state blocking in case of failed connection establishment (R4) and for the elimination of bypassing traffic (R5). To additionally ensure functionality in captive networks, packets required for captive portal detection have to be exempt from the traffic blockage (R2). In the comparison of both native and third-party clients, the only application fulfilling all requirements was the Mullvad VPN app for iOS. On Windows, Android and Ubuntu, effective leak protection automatically caused captive deadlocks due to suppressed captive portal detection (R2). Particularly on iOS and Android, the requirement to minimise startup traffic (R3) was often not fulfilled, since exceptions to traffic blocking were made by the system APIs that caused leakage of platform traffic. Both EncryptMe and ProtonVPN exhibit race conditions appearing during the connection to a captive network, indicating imprecisions in the prioritisation of VPN startup or in the developers guide for secure API use.

Altogether, several bugs were found and reported, including leaks of third-party traffic during the use of the on-demand connection handling macOS offers via it's API, which is meant to provide blocking functionality and VPN setup in specified networks.

Result: While unprotected surfing in public Wi-Fi networks makes eavesdropping possible – and, in some cases, easy – the security guarantees promised by the use of a VPN are not always met in practice.

In fact, only one of the tests conducted on clients across various platforms successfully met all security requirements. This highlights that the current implementation of VPNs is less secure than advertised, and significant improvements are necessary to enhance both their security and functionality.

4.7 Mitigation: Selective VPN Bypass to Mitigate Captive Deadlocks

In the previous section, 25 different test configurations of VPN clients on multiple platforms showed that in only 9 of the scenarios, captive portal detection was successfully performed by the system. The situation is potentially exacerbated by the fact that *none* of the native clients support captive portal detection. If they had mechanisms for intrinsic captive portal detection, commercial implementations could either employ the same mechanisms or improve on them in their applications.

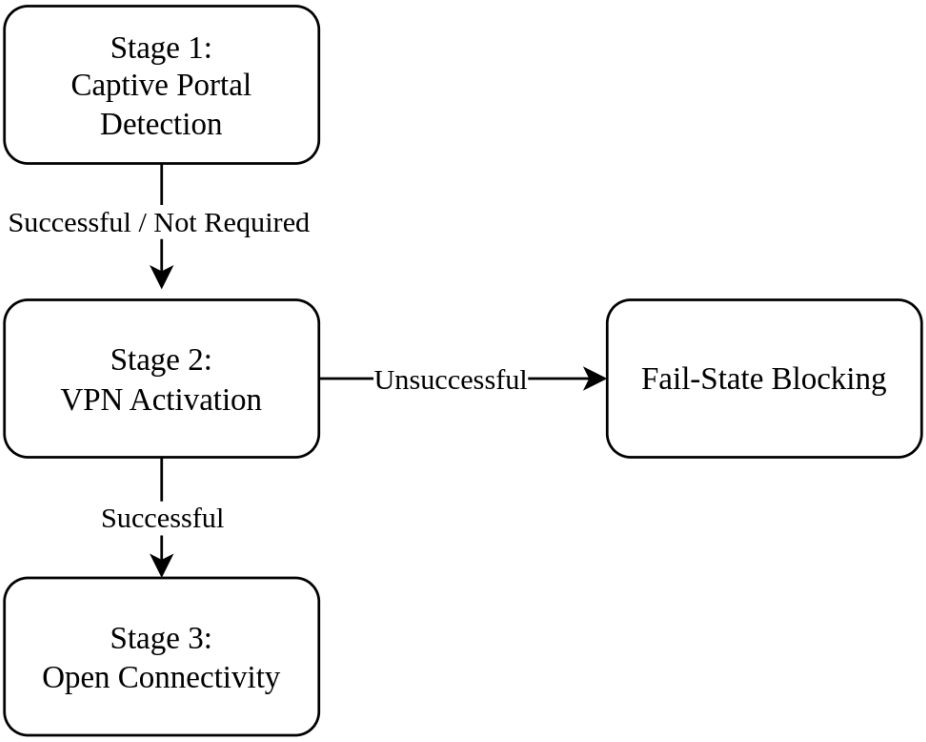


Figure 4.8: A flow diagram illustrating the stages of the selective VPN bypass.

To enhance the captive portal detection capabilities of devices during the use of a VPN, a mechanism for selective VPN bypass is required. This section is, again, based on the paper [Bur+21], which was co-authored within the scope of this dissertation.

A selective VPN bypass that allows access to captive portals during VPN connection establishment consists of the following three stages, which are also illustrated in Fig. 4.8:

Stage 1: Captive Portal Detection

Stage 2: VPN Activation

Stage 3: Open Connectivity

In the **first stage**, depicted in Fig. 4.9, only requests required for captive portal detection with the predefined destination of a platform's detection server are allowed. All other traffic, including unrelated platform services and third-party communication should be blocked completely. This can be implemented using restricted networking capabilities dedicated only to the process responsible for captive portal detection. For sign-in with the captive portal, a minimal, isolated web browser is required. An alternative means of implementing captive portal detection from within a VPN

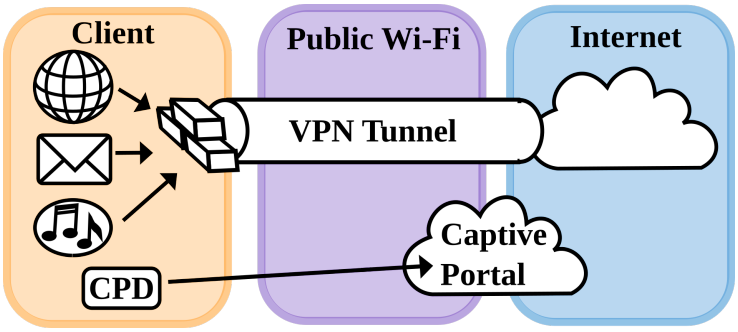


Figure 4.9: Stage 1 of the *selective VPN bypass*: Only traffic required for captive portal detection allowed. Image source: [Bur+21].

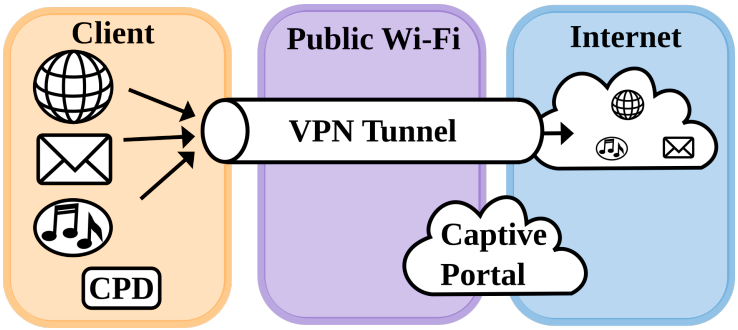


Figure 4.10: Stage 3 of the *selective VPN bypass*: Traffic is only allowed to go via the VPN provider. Image source: [Bur+21].

application is the use of dedicated firewall rules that allow only for the traffic of services related to captive portal detection to be transmitted.

The **second stage** commences after successful captive portal remediation. The system should then commence VPN connection establishment, granting all networking capabilities required for VPN connection establishment. In case the connection establishment is **unsuccessful, it should result in fail-state blocking**, with all network connections suspended until manual intervention by the user determines the subsequent course of action. On the other hand, **successful completion of this stage results in the transition to stage three**:

In **stage three**, depicted in Fig. 4.10, all networking capabilities are granted to platform services, applications and third-parties, and all traffic is tunnelled via the VPN provider without bypassing the VPN tunnel.

In case of a connectivity failure at any stage, connection establishment commences again with stage one. Unsuccessful connection establishment always results in fail-state blocking. This design results in VPN tunnel establishment that adheres to all

the requirements set out in Section 4.4, offering stronger privacy guarantees and more robust functionality than almost all of the applications tested in Section 4.6.

4.8 Conclusion

Ever since Wi-Fi capable devices became easily portable, the use of public access points has made mobile communication both accessible and convenient. However, this ease of movement comes with inherent risks as shown in Section 4.2: The widespread use of open Wi-Fi networks makes monitoring traffic trivial, since open networks do not employ any form of traffic encryption. However, a popular alternative is only slightly better: A WPA2-protected access point with a publicly available password. In WPA2-protected public networks, connection establishment is a security critical process, since WPA2 requires a handshake procedure, which can, if monitored by an attacker, be used to decrypt the following communication. Newer protocols like WPA3 and OWE feature a better handshake protocol, which makes key derivation attacks infeasible. In the face of these protocols, attackers would instead have to resort to active attacks when attempting to monitor and decrypt user traffic. However, these protocols are not in wide use yet.

Users nevertheless wanting to use public Wi-Fi networks therefore have to resort to other means of protecting their traffic. A popular approach is the use of a VPN: In theory, all traffic is encrypted and then transmitted via a VPN tunnel, ensuring that both the content as well as the metadata of the traffic is concealed from eavesdroppers. To effectively fulfil the privacy protection guarantees and perform as expected in public Wi-Fi networks, VPNs must meet certain requirements, as summarised in Section 4.4: They should be able to remain active at all times, particularly during startup, and additionally minimise startup traffic as much as possible to reduce the risk of leakage. In case of connection establishment failure, they should enter a blocking fail-state, in which no traffic is transmitted at all. Also, no traffic should bypass the tunnel. To ensure functionality in public Wi-Fi networks, they should additionally support the detection of captive portals: Without this, the use of an always-on VPN will otherwise result in a captive deadlock. However, tests conducted and presented in Section 4.6 showed that these requirements are neither fulfilled by any of the native VPN clients provided by the specific platforms, nor by most of the popular VPN services tested. In fact, only the iOS client that Mullvad provides fulfils all criteria. This underlines the urgent need to improve both the API provided by the platforms, as none of them contain mechanisms for captive portal detection, as well as the commercial implementations supplied by the VPN providers.

To indeed fulfil all requirements, a selective VPN bypass upon connection establishment with a Wi-Fi network is proposed in Section 4.7. Here, the traffic required for captive portal detection is allowed outside the VPN tunnel, and all other traffic blocked. If the captive portal remediation is successful, only the traffic required to establish the VPN tunnel is allowed, and only if the tunnel establishment is successful can all other traffic be routed through the tunnel. This design ensures that all requirements for privacy and functionality of VPNs are met, protecting both connection establishment and subsequent communication from passive and active attackers within public Wi-Fi networks.

In conclusion, the stage of connection establishment entails various security challenges. In public Wi-Fi networks, connection establishment is particularly critical, since in this stage, key material for the subsequent encrypted communication is exchanged. In public Wi-Fi networks, passive attacks are possible in both open and WPA2-protected networks, and active attacks in both OWE and WPA3. A VPN *can* conceal both traffic and metadata from eavesdropping, but depending on the implementation, it might also leak sensitive data, particularly during connection establishment. However, the use of a VPN employing the selective bypass introduced in Section 4.7, thereby meeting all the requirements summarised in Section 4.4 would ensure a secure and privacy-friendly connection establishment even in insecure public Wi-Fi networks.

5

Data Transfer - Unencrypted Broadcast Communication using the Example of ADS-B

In the previous chapters, the stages of device discovery and connection establishment were examined in detail, and typical pitfalls and their mitigations were analysed using the example of probe requests for device discovery in Wi-Fi networks, and the use of a VPN client to safeguard against eavesdropping during connection establishment in public Wi-Fi networks. Once a connection has been successfully established, the next stage is data transfer. This is the stage of communication where information is exchanged between two (or more) parties. A trivial example for data transfer is an unencrypted and unauthenticated broadcast message, which can be received by anyone with the correct equipment in place. Here, the question is whether and how the legitimacy of such broadcast messages can be tested and verified. Additionally, data transfer can be disrupted due to various causes: Unintentional causes could for example be devices leaving the radius within which a connection can be maintained, or that the quality of service degrades due to various reasons. An intentional cause for a disrupted connection, on the other hand, can be an attack on the system - either via fine-grained attacks tackling single connections or by jamming a whole frequency spectrum.

In this chapter, the focus lies on the Automatic Dependant Surveillance-Broadcast (ADS-B) protocol, an example of **broadcast communication for positional information on aircraft**. The protocol is particularly interesting from a security perspective

as it was built without any security or authentication mechanisms. It is susceptible to active and passive attacks that disrupt the connection or violate the integrity or confidentiality of the content: The lack of encryption allows eavesdropping on the content, and the lack of authentication enables attackers to inject or modify messages. As no mechanism exists to estimate packet loss, fine-grained message deletion attacks or all-out jamming can remain undetected in the system. Section 5.1 provides a thorough overview over the possible attacks vectors in the context of ADS-B messages.

The lack of security is a particularly interesting research topic, and academic discussions on retrofitting security mechanisms have introduced various solutions. These are detailed in Section 5.2, where preliminary studies and related work are reviewed. They can be classified into three fields: The first concerns the **redesign of the ADS-B protocol to include cryptographic mechanisms** for message encryption, authentication and integrity protection. It is unlikely that any modifications to the ADS-B protocol are implemented within a timely manner, as the roll-out of the protocol itself was already extremely slow: Albeit the deployment started in 2009, it is still not in ubiquitous use, with the deadline for mandatory implementation having been postponed several times after the first one in 2015, with the latest deadline being 2025. Nevertheless, by 2021, 97 % of the observed commercial airlines were ADS-B compliant already [Sun+21]. Modifying a system that has, on one hand, required a substantial amount of time for enrolment and is still not ubiquitously used, and, on the other hand, might require for the significant amount of already-enrolled devices to be replaced in favour of a new protocol, is unlikely to happen. Nonetheless, the approaches for a protocol redesign in favour of cryptographic protection are discussed in more detail in Section 5.2.1.

The second commonly discussed strategy to improve the reliability of the ADS-B protocol concern the signal: On one hand, it is possible to apply **fingerprinting techniques on the signal**. This can be done via hardware- software- or channel-based verification methods. On the other hand, modifications on the signal itself to include techniques like spread spectrum protocols or frequency hopping can increase the resistance to jamming and impede eavesdropping on the content. These approaches are discussed in more detail in Section 5.2.2.

The third strategy concerns the **verification of signal origin**. This can be done by multilaterating the signal and thereby verifying the signal origin stated within the message, or by using crowd sensors and their typical overlapping reception range to gauge the legitimacy of the content of a received ADS-B message. This is discussed in Section 5.2.3. Additionally, this chapter introduces another technique to verify signal legitimacy: A strategy for location verification of ADS-B messages using distributed public sensors, introduced in Section 5.3.

Relevant Publications The location verification scheme LoVe described in Section 5.3 has previously been published in [McD+23].

5.1 Attacks on ADS-B

Protocols lacking encryption and authentication are vulnerable to several attacks. In the case of ADS-B, five types of attacks [WSG20] can be distinguished:

- eavesdropping
- message injection
- message modification
- message deletion and
- jamming.

Additionally, attacks on the GNSS system have direct consequences for ADS-B messages and should therefore be considered an additional attack vector. The following sections will elaborate on the potential attacks targeting ADS-B.

Eavesdropping Since ADS-B messages are transmitted in plaintext, it is easily possible to perform a passive attack on the link: Attackers can eavesdrop on the traffic using off-the-shelf equipment like a Software Defined Radio (SDR) and an antenna for the 1090 MHz frequency band. Websites like the OpenSky network even rely on crowdsourcing to gather their data: Thousands of independent users set up ADS-B receivers and feed the data into the database. The data, in turn, can be accessed by researchers and paying customers.

Message Injection The lack of authentication of messages makes it possible to send fake ADS-B messages. The goal can be either to overpower signals sent by a specific aircraft to modify the trajectory registered by a ground station, or to inject a whole new non-existent aircraft, a *ghost plane* [WSG20]. Using a message injection attack it is potentially possible to disrupt normal flight procedures, as the received signals can appear on surveillance monitors. In aircraft, *ADS-B-In* data is used in the Airborne Collision Avoidance System (ACAS) to map surrounding aircraft and avoid potential collisions. The system provides advisories to pilots, who can then manually modify the trajectory or choose not to react to potentially false alarms [ICA20]. Spoofed ADS-B messages appearing on ground tower surveillance monitors can also cause air traffic controllers to issue instructions to alter the course. However, ground

towers typically use additional sources of information (e.g. the Primary Surveillance Radar (PSR)), and could therefore validate the existence of ghost planes prior to issuing instructions. Injecting ADS-B messages is therefore a means of confusing automated digital systems, but the ultimate decisions are still made by humans.

Particularly seeing that aggressions are welling up globally, a subform of a ghost plane injection is thinkable: A *ghost fleet injection*, a DDoS-like attack using ADS-B signals. If a very large number of ghost planes were injected using ADS-B messages, distinguishing between real and fake aircraft would be difficult for both ground stations and ADS-B-In-capable aircraft. Ground stations would likely have to temporarily abandon air surveillance using ADS-B and instead rely on PSR only. Aircraft would lose all capability to chart the surrounding airspace using ADS-B messages and would have to instead rely on their line of sight.

Message Modification Using message overlay techniques, bit flipping or a combination of message deletion and injection, it is a possible – but very complex – task to modify messages sent by a legitimate node. An overlay attack attempts to replace a legitimate message with a high-power signal, while bit flipping can change specific bits in a message to modify the content. The combination of both can be used to flip bits in a message using a strong signal [WSG20]. Note that, of course, it is not enough to just flip specific bits in the content, but that the parity has to be adjusted as well since the message would otherwise be discarded. Such an attack again requires impeccable synchronisation with the target node. An alternative attack can be to first provoke message deletion by disrupting enough bits in a transmitted message, and then insert a modified message in its stead [Pöp+11]. Since high precision is required to modify signals, which is further hampered by the large distances the messages can travel, message modification would need to be done either in very close proximity to the transmitting aircraft or to a specific receiving ground station.

Message Deletion and Jamming As mentioned in Section 2.4, the ADS-B protocol is designed in such a way to contain a 24-bit CRC error-detecting code for every one of the 112 bit long messages. With this parity, it is possible to correct up to 5 corrupt bits per message. A message containing more than 5 bit-errors can not be corrected and is therefore considered corrupt and discarded. An attacker can use this by emitting a timed signal to generate bit errors in specific messages. Aircraft whose ADS-B messages are deleted disappear from surveillance monitors, which increases the risk of collisions [WSG20]. Since ADS-B is a broadcast protocol and reception of a message is not acknowledged by others, the corresponding parties will remain oblivious to the corruption of the messages. Message deletion is technically complex since it requires very strict time synchronisation [WSG20]. Instead of targetting

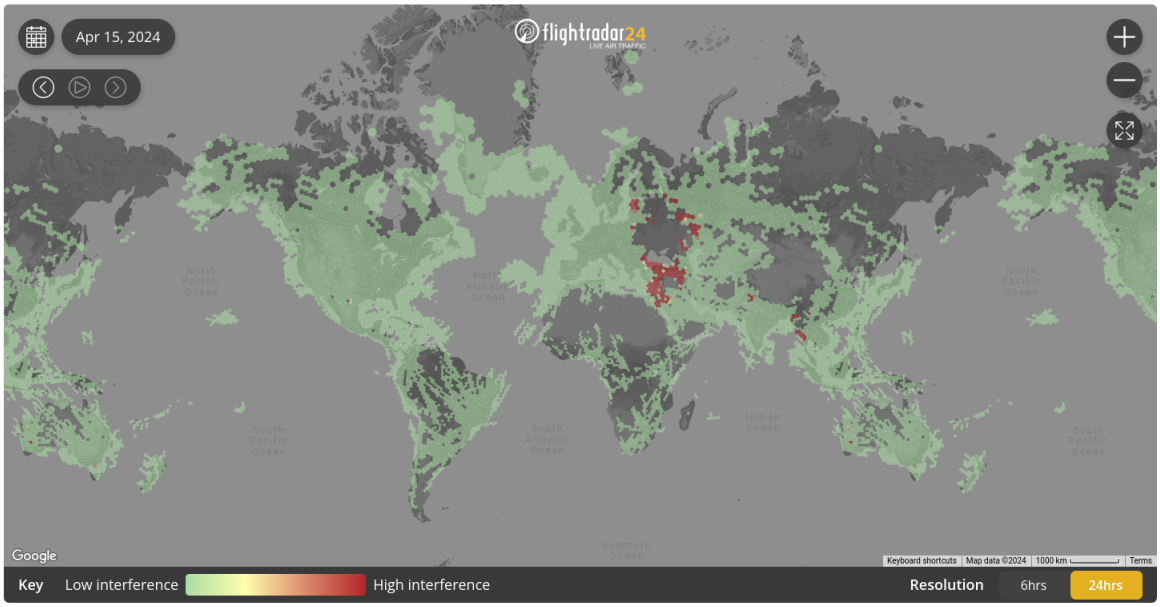


Figure 5.1: An overview of GPS-jamming attacks recorded via ADS-B signals, provided by Flightradar24 [Fli24]. Areas coloured in green denote low interference, while red colouring denotes high interference. Interference is particularly pronounced in south-eastern Russia and in the Middle East. Screenshot of [Fli24].

single messages, ADS-B message transmission and reception can be disrupted by jamming, i.e. by sending high-power data via the whole frequency band.

GPS-Attacks Instead of attacking the ADS-B system directly, a recent surge in attacks on Global Navigation Satellite System (GNSS) (cf. Section 2.4.3), e.g. Galileo, GPS, GLONASS, BeiDou, has been detected. While Russian-based attacks took rise even before its attack on Ukraine, an increase could also be noted around the time of the attacks of the Hamas, in the last quarter of 2023, over three distinct regions in the Middle East: Around Tel Aviv, Baghdad and Cairo [Zee23; Gau23]. Since the ADS-B system relies on legitimate GPS signals, spoofing or jamming the GNSS system also directly affects the ADS-B system. Vice versa, it is possible to use ADS-B sensor data to construct a map of GNSS jamming or spoofing [Fli24], as can also be seen in Fig. 5.1.

5.2 Preliminary Studies and Related Work

An influential publication regarding the insecurities of the ADS-B system was published by Costin et al. [CF12] and presented at Black Hat 2012. They constructed a lab environment in which they simulated ADS-B transmitting and receiving devices, but to ensure that no false information is accidentally transmitted, the messages were exchanged via a wired connection. This way, they modelled both passive as well as active attackers and thereby showed that it's possible to attack the system with Commercial Off-The-Shelf (COTS) equipment.

Strohmeier et al. [SLM15b], Manesh et al. [RK17] and Wu et al. [WSG20] provide surveys regarding the state of ADS-B security, attacks and countermeasures. All of them agree that the protection mechanisms previously suggested by others and then analysed in their surveys only serve to provide a quick improvement of the security and serve only as a partial answer to the addressed vulnerabilities. Neither of the proposed retrofitted security measures can guarantee properly hardened system with all-encompassing security in place. Strohmeier et al. and Wu et al. maintain that a complete redefinition of the protocol is required. Wu et al. additionally suggest to incorporate deep learning methods for vulnerability analysis and to incorporate blockchain technologies for security verification. However, Manesh et al. consider a complete replacement impractical due to the costs and time a redeployment entails. They suggest to implement methods to address vulnerabilities while retaining the system's simplicity and flexibility. They additionally suggest to implement encryption and authentication via control channels, both in-band and out-of-band.

ADS-B was designed without security features to make it as lightweight and easily usable as possible, while at the same time using existing equipment with updated functionality for transmission, instead of requiring an additional sender to be installed on the aircraft. Ensuring authenticity of the signal retroactively can be done in three different ways:

(1) The first is to **redesign the ADS-B protocol** to include encryption or message authentication. As the deployment of ADS-B has still not been completed, and modifications of the protocol are unlikely to be implemented any time soon, this is the least likely approach by which the current ADS-B protocol will be secured. Since it is possibly relevant to other broadcast systems that are to be retrofitted with security mechanisms, it will be discussed in detail in Section 5.2.1.

(2) The second approach is to **fingerprint the signal** itself. Here, a hardware-, software- or channel-based fingerprint is used to verify signal authenticity. While such fingerprinting techniques are a passive way of verifying signal integrity, possible active signal protection would entail techniques like frequency-hopping

spread spectrum (FHSS) or direct sequence spread spectrum (DSSS). These approaches will be discussed in more detail in Section 5.2.2.

(3) The third verification strategy is the **verification of signal origin**. These are presented in Section 5.2.3.

5.2.1 Cryptographic Protection

Wesson et al. [WHE14] focus on the feasibility of the use of cryptographic protection mechanisms in the ADS-B protocol. They emphasise the complexity and challenges associated with implementing cryptography for ADS-B, considering the international, bandwidth-constrained, and interference-sensitive nature of the aviation communication system. They examine both symmetric and asymmetric encryption schemes, as well as digital signatures as a means to protect the integrity of ADS-B messages.

With respect to **symmetric encryption**, Wesson et al. underline the problem of key distribution: Since the same key is required for both encryption and decryption, it has to be accessible to all ADS-B transceivers. The key distribution strategies either utilize tamper-proof hardware to store keys, or change the keys for every flight. The first approach is vulnerable to key disclosure, with the security of the system reduced to the security of the tamper-proof equipment. The second approach is inhibited by the fact that all parties that might receive the signal in question have to be able to access the key. Additionally, a possible leak of the active key database would completely compromise the system. [WHE14]

Since the key distribution and key management are the biggest problems that symmetric encryption in ADS-B messages would have to overcome, **asymmetric cryptography** is suggested instead. Here, confidentiality can be ensured by the use of a key pair consisting of a public key and a private key. While the public key of the receiver is used to encrypt the data, the receiver can use its private key for decryption. All parties involved in the transmission of the messages only have to maintain their own private key and the public keys of nearby transceivers, and no additional key exchange has to be performed [WHE14]. One drawback of this approach is the amount of messages required to be transmitted: While the current protocol requires for one message to be transmitted for all nearby receivers n to receive the information, the use of an asymmetric scheme requires for n messages to be transmitted to reach all n neighbours. Additionally, the use of asymmetric encryption is inhibited by the length restriction of 112 bits, as asymmetric encryption schemes enlarge the cleartext significantly.

The ADS-B protocol never included means to maintain confidentiality, and eavesdropping on ADS-B messages is neither discouraged nor prohibited. The protection goal of confidentiality is therefore likely not considered important in this specific protocol. The protection goals of availability and integrity, on the other hand, hold greater significance: Availability is maintained through constant transmissions occurring every 0.5 seconds. And although the protocol contains no means of integrity protection, message injection, message modification, and ghost plane injections are only possible due to the lack of authentication.

An approach to authenticate the sender and thereby ensure the legitimacy of the source, and additionally ensure that they have not been tampered with during transmission, can be implemented using **digital signatures**: When an aircraft transmits an ADS-B message, it can sign the message with its private key, creating a digital signature. Receivers can then verify the signature using the sender's public key to confirm that the message was indeed sent by the claimed aircraft and that it has not been altered. This mechanism helps prevent impersonation or spoofing of aircraft identities and the injection of false information into the ADS-B network. Just like message encryption, the length restriction of 112 bits makes it difficult to apply a signature scheme to the current protocol. The Elliptic Curve Digital Signature Algorithm (ECDSA) generates considerably shorter digital signatures than all other digital signature algorithms; an ECDSA signature has a length of 448 bits [WHE14]. It provides an equivalent strength to a 112 bit long Message Authentication Code (MAC) received using a symmetric cipher, while simultaneously having the benefit of avoiding the complicated key exchange required for MACs. While the length of an ECDSA signature is still too large to meet the length restrictions of 112 bits, it could, on the one hand, be overcome by modifying the ADS-B protocol to include longer messages. Since this is unlikely to be implemented, the signed messages could instead be split over several transmissions.

Costin et al. [CF12] propose a different scheme that does not necessarily entail a modification of the packet size or structure, but instead the way the packet content is sent and received: They suggest to compute a signature over each N messages, and transmit parts of the signature over the cycle of the N messages. After each cycle, recipients can compute the validity of the signature and thereby verify the message integrity.

There has been no official movement to include cryptographic schemes into the implementation of the ADS-B protocol as of today. The following sections therefore present other vantages to verify signal and location authenticity that do not rely on cryptographic enhancements.

5.2.2 Radiometric Fingerprinting and Signal Protection

Radiometric fingerprinting is a technique that can be used to verify the origin of a packet by comparing its signal fingerprint to that of previously received messages. Signal Fingerprinting of ADS-B messages can be done via three different vantage points:

(a) software-based,

(b) hardware-based or

(c) channel-based.

Software-based fingerprinting is done by abusing the differences in implementations and resulting behaviour. Since many airlines use identical hardware, it is difficult to differentiate between devices using software-based criteria. **Hardware-based fingerprinting** on the other hand uses radiometric techniques, making use of modulation differences or clock skew to calculate device signatures [RK17]. Since both hardware-based as well as software-based fingerprinting methods are more applicable in close proximity of the transmitting antenna and mainly rely on non-mobile devices, a long-distance and highly dynamic protocol such as ADS-B exacerbates such fingerprinting techniques [WSG20].

Channel-based fingerprinting on the other hand uses features inherent to the communication channel, such as the received signal strength, the channel impulse response or the carrier phase [RK17]. Leonardi et al. [LGF17] additionally combine channel-based fingerprinting and artificial intelligence: Using the phase pattern generated by the transponder oscillator, they train a neural network to classify ADS-B signals into one of seven different aircraft classes. While this does not constitute a reliable way of fingerprinting single aircraft, it is likely applicable to verify measurements found via other fingerprinting techniques, e.g. in a *data fusion* scenario, where ADS-B data is combined with other sources of information, such as the PSR, as introduced in Section 5.2.3.

An example of **modifying wireless communication protocols to withstand jamming and eavesdropping attacks** is to utilise frequency-hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS). In these technologies, the traffic is spread over several channels, utilising a pre-shared spreading code or a specific frequency hopping mode [WSG20]. A secret pre-shared spreading sequence is difficult to distribute in an ADS-B setting, and a fixed pre-shared spreading sequence still allows attackers to perform jamming attacks by applying the sequence to their own attack. Therefore, schemes like the Uncoordinated DSSS (UDSSS) introduced by Pöpper et al. [PSC09] allow for jamming resistance: A message is signed with the

sender's private key, and an error encoding is added to allow for a small number of bit errors. The message is then spread according to a set of spreading sequences, which is publicly known. Neither the attacker nor the receiver know which spreading sequence was chosen, and have to record repeated retransmissions and reassemble the message according to a sliding window approach. While the application of UDSSS to ADS-B message transmission would entail a protocol modification, it would improve the protocol to entail both message authentication and jamming resistance. While UDSSS was *not* designed for use in the ADS-B protocol, Wu et al. [WSG20] suggest it as a means of combating jamming and eavesdropping in ADS-B, but also highlight that the waste of bandwidth resources, it's low performance in comparison to the current system and the additional time required might impede it's use in the ADS-B protocol.

5.2.3 Origin Verification

Several publications have recently attempted to provide verification of ADS-B messages on distributed crowd sensors, e.g. using **Multilateration (MLAT)** on the signals [DAP20]. MLAT is also typically used in telecommunication networks to provide location assessment. Here, a minimum of four receivers are required to pick up the same signal. Signal features like the time of transmission or time of arrival are used to accurately track the location of the sender. Conventional MLAT is therefore rather applicable in limited airspaces, e.g. in the vicinity of airports, where it is also commonly used. Several publications attempt to localise signal origins using MLAT-like approaches but typically apply modifications to reduce the required sensor set: Two approaches published by Strohmeier et al. [SLM15a] perform either location verification or location estimation using the measurement of the Time Difference of Arrival (TDoA). The authors collect a data set and analyse its verifiability with respect to MLAT and then compare it to their own TDoA-based approach. They adapt their approach to work with a minimum of two sensors receiving one message and provide an improvement to location accuracy estimation compared to conventional MLAT by 41 %.

Another approach at origin verification was introduced by Jansen et al. [Jan+21]: They propose a **Machine Learning (ML)-based** approach, in which they utilize vectorsets to evaluate ADS-B sensor response patterns. This is done using a random-forest model. Using a data set of ADS-B messages collected via 729 sensors distributed over Europe in the OpenSky Network, they construct a vector containing the sensor IDs for every single one of the messages received on a specific day. This vector is representative of all sensors that recorded a particular message. They then apply the ML technique Decision Trees (DT) to the vectors. This way, they are also

able to identify various attacks on the ADS-B system, including message injection, modification, and GPS spoofing. While the approach is applicable to a sensor set of size 729, it likely doesn't scale, e.g., in a worldwide scenario where thousands or hundreds of thousands of sensors are in use. In fact, to perform verification of messages captured in European airspace, the authors had to split the training data into several grids to facilitate separate processing. Additionally, since the approach relies on the evaluation of vector sets, a low sensor distribution in an area could negatively influence the verification process.

Various other approaches typically concern **data fusion** for verification of ADS-B messages, combining the positioning via ADS-B with localisation via primary (PSR) or secondary radar or other current traffic control systems [WSG20; SLM15a]. While the combination of various localisation techniques are typically expensive to deploy, they also defeat the original purpose of using ADS-B for surveillance in remote areas, difficult terrain or over oceans.

In the following, the main contribution of this chapter is presented: An approach at comparing newly received signals to a previously constructed mask of legitimate signals.

5.3 Location Verification using Distributed Public Sensors

The various approaches at location verification that have been published to date use either a multitude of crowdsourced sensors to verify the position autonomously, combine various positioning techniques or use machine learning to evaluate publicly available information. These approaches have in common that they are rather costly, either in terms of the number of sensors required to surveil a large area or in terms of computational cost. In this section, a particularly lightweight alternative is presented: Location Verification (LoVe) verifies the transmitted positions in ADS-B messages captured by a specific sensor by comparing them to previously monitored locations by the same sensor. LoVe was previously published in [McD+23], which was written within the scope of this dissertation.

5.3.1 System, Attacker and Threat Model

The **system model** is composed of ADS-B messages which are periodically sent by legitimate aircraft and recorded using geographically distributed ground sensors from the OpenSky network and the Flightradar24 network. The sensors report the

received messages to a central server, which, as assumed in this system model, accumulates them and implements LoVe to verify the positional claims.

The **attacker** considered can monitor and **actively transmit** ADS-B messages using COTS equipment like an SDR and additionally multiple stationary ADS-B transmitters. They have both **bounded computational resources**, as well as **bounded financial resources**, e.g. they are capable of transmitting messages from a stationary position, and have no means of moving as fast as a legitimate aircraft would. The bounded financial resources also include that they have no means of e.g. acquiring a drone capable of imitating the movements and speed of an airplane, with which they could transmit legitimate-seeming ADS-B messages from the correct GPS coordinates.

The **threats** LoVe can defend against are message injection and ghost plane injection, location spoofing and attacks on sensors. The first two attacks are possible since ADS-B messages are neither encrypted nor authenticated: An attacker can **inject legitimate-looking messages**, either to modify the data associated with an existing aircraft or to inject a non-existent aircraft, a ghost plane. The injected messages would be recorded by surveillance monitors and have the potential to disrupt flight procedures: Since the ACAS uses ADS-B data for in-flight collision avoidance and accordingly provides advisories to pilots to modify the trajectory if the ADS-B messages of another aircraft are captured in the vicinity, the aircraft crew might react to ghost plane transmissions. Additionally, air traffic controllers might issue instructions to alter the aircraft course in reaction to fake ADS-B messages. LoVe can also be used to detect such threats.

Another plausible attack that LoVe can defend against is location spoofing. There are two distinct ways that this can be achieved: Either via **GNSS spoofing** or by **directly modifying messages**. The latter attack is a subform of message modification as described above: Instead of injecting a whole message, parts of a legitimate ADS-B message are *overshadowed* using a stronger signal, or modified by flipping single bits [SLM13b]. Needless to say, injecting modifications into existing messages requires not only for the bits of interest to be flipped, but also for manipulation of the parity check. When a received message can not be corrected via error correction, it is dropped. This behaviour can be exploited by an attacker to cause message deletion. While LoVe can **not detect message deletion attacks**, **coordinates outside the legitimate coordinate range can be detected and flagged**.

Another threat LoVe can defend against are **attacks on sensors**. Again, there are two ways this can be achieved: Either by flooding specific legitimate sensors with bogus messages, or by forwarding illegitimate messages via an own, injected, sensor.

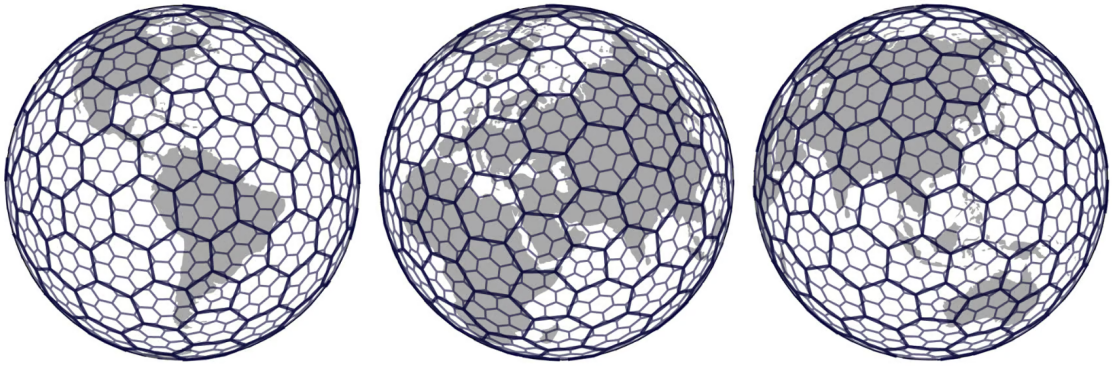


Figure 5.2: A visualisation of H3 cells spanning the earth surface from three different perspectives. With a low resolution, only 122 very large cells are required to cover the whole surface of the earth, while a higher resolution, as e.g. seen in Fig. 5.3, requires a far larger number of smaller cells to do the same. Image source: [Bro18].

5.3.2 LoVe Approach

LoVe employs a mask-based approach: All previously recorded coordinates are transformed to an index, by which a plausibility mask is constructed. This mask contains a classification of signals received by a sensor: All previously received coordinates are mapped using an indexing system, which is described in more detail below, constructing a mask of legitimately received signals for each sensor. To test a new signal, its coordinates are transformed using the indexing system and then compared to the range of signal reception that is common for this particular sensor. If the received coordinates are outside the range of the plausibility mask, they are flagged.

The **indexing system** chosen is H3[1], a hexagonal geospacial indexing system. While other projections of the world map, e.g. the mercator projection, are characterised by variable cell size, H3 provides evenly distributed cells in 16 different resolutions. H3 attempts to map the surface of the earth using hexagons and pentagons, as shown in Fig. 5.2, which, added together, make up the total amount of cells. Table 5.1 shows a list of all resolutions and the respective number of cells. The lowest resolution, 0, is made up of only 122 cells, while 15, the highest resolution, covers the earth surface using 569 707 381 193 162 cells. Table 5.1 also depicts the average area of a hexagon in km^2: It ranges from around one square metre in resolution 15 to about four million square kilometres in resolution 0. Seeing that ADS-B signals can typically be received up to a range of 370 km around the transmitter [Sca02], the resolutions 2-7 are most

1. https://h3geo.org

Resolution	Total Cells	Hexagons	Pentagons	Avg. Hexagon Area (km^2)	Pentagon Area (km^2)
0	122	110	12	4,357,449.416078381	2,562,182.162955496
1	842	830	12	609,788.441794133	328,434.586246469
2	5,882	5,870	12	86,801.780398997	44,930.898497879
3	41,162	41,150	12	12,393.434655088	6,315.472267516
4	288,122	288,110	12	1,770.347654491	896.582383141
5	2,016,842	2,016,830	12	252.903858182	127.785583023
6	14,117,882	14,117,870	12	36.129062164	18.238749548
7	98,825,162	98,825,150	12	5.161293360	2.604669397
8	691,776,122	691,776,110	12	0.737327598	0.372048038
9	4,842,432,842	4,842,432,830	12	0.105332513	0.053147195
10	33,897,029,882	33,897,029,870	12	0.015047502	0.007592318
11	237,279,209,162	237,279,209,150	12	0.002149643	0.001084609
12	1,660,954,464,122	1,660,954,464,110	12	0.000307092	0.000154944
13	11,626,681,248,842	11,626,681,248,830	12	0.000043870	0.000022135
14	81,386,768,741,882	81,386,768,741,870	12	0.000006267	0.000003162
15	569,707,381,193,162	569,707,381,193,150	12	0.000000895	0.000000452

Table 5.1: The number of cells, hexagons and pentagons, including the pentagon and hexagon areas, at each resolution [H323].

fitting for the LoVe scheme: They cover an average area per hexagon between around 5 km^2 and 86 000 km^2.

Sample data to construct a database containing real-world ADS-B data could be obtained from both the OpenSky Network and FlightRadar24. Both store their data in a different format: For each message recorded, OpenSky stores which sensors obtained the specific message. FlightRadar, in turn, separates messages into flights and then stores CSVs with each message recorded by the specific plane during the flight. Both data sets were recorded on July 23rd, 2021. For both data sets, the range was manually limited to contain **only data recorded within the European continent**, with the latitude bounded between 30 and 75, and the longitude between -25 and 45. The OpenSky data set was recorded by only **971 sensors** which collected 160 526 553 distinct ADS-B messages. This amounts to every sensor capturing around 2 141 598.77 messages. The OpenSky data set exhibits a significant overlap in messages recorded: a particular message was on average **recorded by 12.95 sensors simultaneously**. The data format used in the data set supports such overlap in message reception, and the overlap in turn can serve to enhance trust in the data received.

The FlightRadar data set contained 14 092 420 messages recorded by **11 594 sensors**, which amounts to an average of 1208.12 messages captured per sensor. The data format does not support the storage of overlapping messages recorded by different sensors. The overlap was therefore calculated by taking all fields of an ADS-B message into account and **calculating a hash over it**. The script can be seen in Listing 5.1: In row 8, a hash is constructed over several items contained in the data set: The current altitude, heading, latitude, longitude and speed. In line 9, several

```
def read_from_file_fill_db_with_hash(myfile, db, counter):
    with open(myfile, newline='\n') as csvfile:
        allelements = csv.DictReader(csvfile, delimiter=',')
        for row in allelements:
            current_sensor = row['radar_id']
            lati = row['latitude']
            longi = row['longitude']
            myhash = hash(str(row['altitude']) +
                str(row['heading']) + str(lati) + str(longi) +
                str(row['speed']))
            sql = 'INSERT INTO COORDINATES (id, sensor_id, lat,
                long, hash) values(' + str(counter) + ', ' + str(
                 current_sensor) + ', ' + str(lati) + ', ' +
                    str(longi) + ', ' + str(myhash) + ')'
            db.execute(sql)
            counter = counter + 1
    return counter
```

Listing 5.1: The python script used to construct a database of flightradar data, including a hash of all attributes recorded in the ADS-B signal. The hash can then be used to compare the received signals and estimate how many signals were recorded by more than one sensor.

elements of the ADS-B message are then stored in a database, including the hash previously constructed.

Using SQL queries on the database then reveals the number of overlapping messages recorded by more than one sensor: `SELECT DISTINCT hash FROM COORDINATES;` retrieves the number of distinct hashes, and `SELECT id FROM COORDINATES ORDER BY id DESC LIMIT 1;` shows the total number of entries. The results show 58 491 973 distinct rows and 58 382 552 distinct hashes: **0.18 % of the messages exhibited a hash that had also been recorded by another sensor** in the data set, and was therefore a message received by more than one sensor. Particularly seeing that the FlightRadar data set was recorded by 11 594 sensors, while the OpenSky network only used 971 sensors, the small number of overlapping messages is a surprise, seeing that a message was on average recorded by 12.95 sensors simultaneously in the OpenSky data set.

5.3.2.1 Use of the H3 Geospacial Index

Each of the ADS-B messages were processed so as to transform their coordinates to H3 cell IDs, in the following called *h3id*. A database table is then constructed for the LoVe-scheme with the combination of h3id and sensor ID as primary key, with each row filled with the combination of h3id and sensor ID, and the amount

```
1  import h3
2
3  def construct_h3_table(db, tablename_in, tablename_out,
       cellsize):
4      with db:
5          db.execute("CREATE TABLE " + tablename_out + "(h3id
               TEXT, sensor_id INTEGER, amount INTEGER, primary key
               (h3id, sensor_id));")
6          counter = 0
7          db.execute("SELECT id, sensor_set, lat, long FROM " +
               tablename_in)
8          data = db.fetchall()
9          h3dict = {}
10         for row in data:
11             counter += 1
12             sensor_set = row[1]
13             lat = row[2]
14             lng = row[3]
15             h3id = h3.latlng_to_cell(lat, lng, cellsize)
16             for sensor_id in sensor_set:
17                 tmpid = h3id + "," + str(sensor_id)
18                 try:
19                     h3dict[tmpid] = h3dict[tmpid]+1
20                 except(KeyError):
21                     h3dict[tmpid] = 1
22             if (counter % 100000) == 0:
23                 print("managed", counter, "entries")
24         for e in h3dict:
25             h3id, sensor_id = e.split(",")
26             amount = h3dict[e]
27             db.execute("INSERT INTO " + tablename_out + " (h3id,
                   sensor_id, amount) values ('" + h3id + "', '" +
                   sensor_id + "', " + str(amount) + ")")
28         print("transferred all entries to", tablename_out)
```

Listing 5.2: Using the previously constructed database, a new table is created containing the h3id, the sensor ID and the amount of times a message was recorded in the specific cell by the sensor.

of times the sensor has received a signal in this cell in total, as can be seen in Listing 5.2. The overall amounts of messages with respect to the resolution can be seen in Table 5.2: the lower the resolution, the more area one single cell covers, and the higher the amount of messages captured per cell is, as can also be observed in Fig. 5.2. The higher the resolution is, the less messages were captured per cell. This is also illustrated in Fig. 5.3. It depicts a partial map of Europe overlapped with H3-cells in resolution 4. The colouring denotes the highest amount of messages captured in a cell by one sensor.

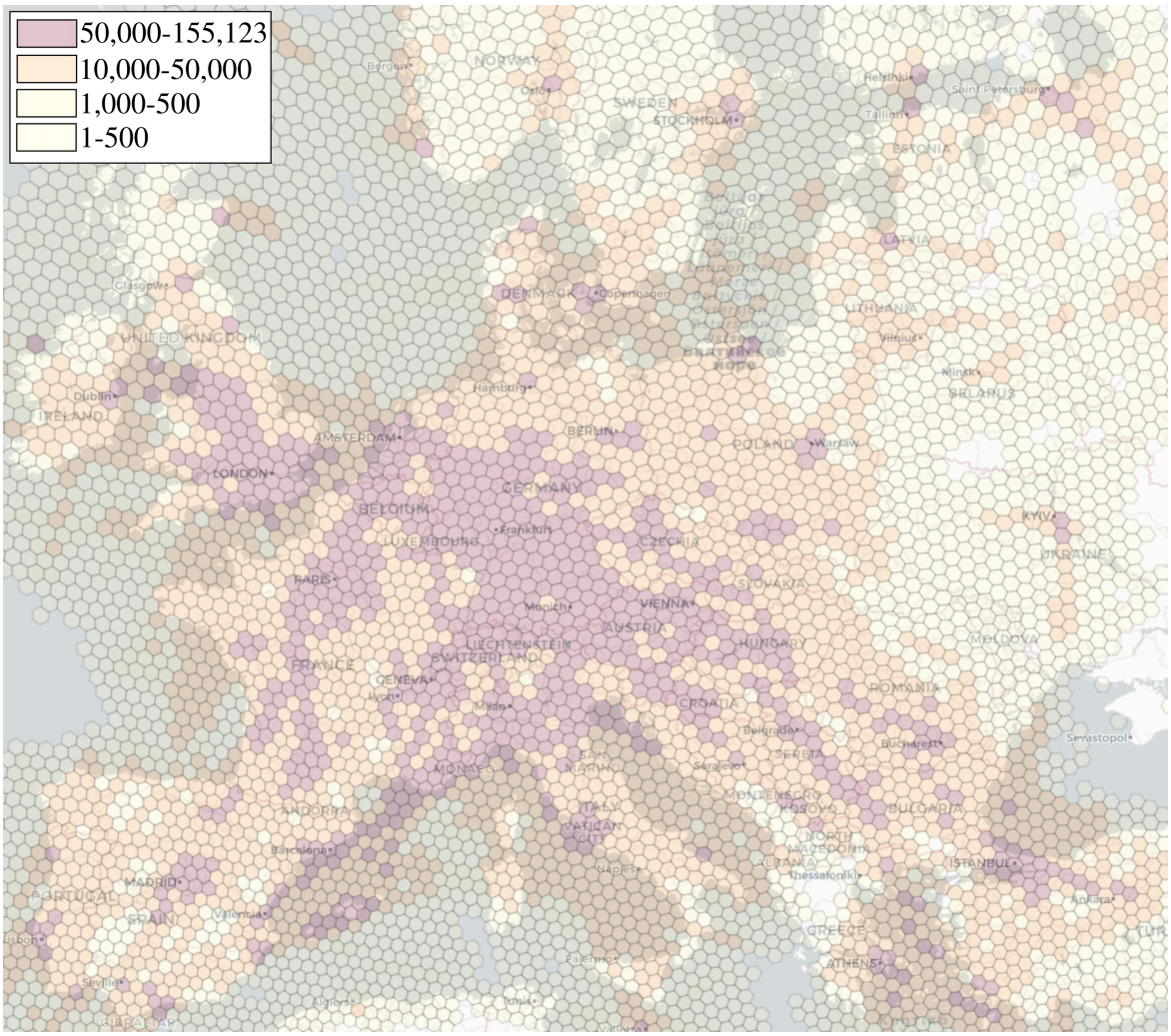


Figure 5.3: An illustration of the hexagons in resolution 4 covering Europe. The colours illustrate the highest number of messages received by a sensor in the cell, ranging from 1 to 155 123. Image source: [McD+23].

The processing, indexing via H3 and storage of single ADS-B messages can be observed in Fig. 5.4: The table on the left shows an example of several messages captured by multiple sensors of the OpenSky network. The first entry with ID 4 in this figure is the sensor with sensor ID -1408236192. The latitude and longitude recorded in this particular message correspond to the h3id 841ec95fffffff. The table on the right of Fig. 5.4 shows several entries of this sensor in the amount table, the first of which shows that this sensor recorded 26 221 messages in the cell of the h3id mentioned above.

Another example of sample data in resolution 4 based on OpenSky network data can be found in Fig. 5.5. In this case, the data is sorted by h3id, and it can be seen

id	sensor_set	lat	long
4	{-1408236192}	40.69959737486758	28.78761291503906
5	{970515328,-1408237567,-1408234…	51.97663879394531	7.724380493164062
6	{852742020,-1408235387,-1408233…	48.84745788574219	10.95667912409856
7	{-1408232695,2020281642,9547850…	50.87796020507812	7.183351258973818
8	{90310}	39.87189858646716	25.29156494140625
9	{-1408235078,91319}	42.38429260253906	28.8265644420277
10	{1408236197}	34.98209807832362	29.6173095703125

	h3id	sensor_id	amount
7	841ec95fffffffff	-1408236192	26221
8	841ec97fffffffff	-1408236192	144448
9	841ec99fffffffff	-1408236192	51895
10	841ec9bfffffffff	-1408236192	87154
11	841ec9dfffffffff	-1408236192	41049
12	841ecb1fffffffff	-1408236192	13765
13	841ecb3fffffffff	-1408236192	3339

Figure 5.4: On the left hand side, each row shows one of the original OpenSky messages containing a set of sensors that recorded the message and the latitude and longitude transmitted in the message. The right hand side shows data transferred to the amount table in resolution four: latitude and longitude were converted to an h3-cell ID, called *h3id* in this scheme. The corresponding sensor that recorded the message and the amount of times the specific sensor recorded a message in the h3 cell are stored together.

h3id ▼ 1	sensor_id	amount
843f76bfffffffff	-1408233591	16024
843f769fffffffff	1408236197	38
843f769fffffffff	90310	11267
843f769fffffffff	-1408233591	15831
843f767fffffffff	90310	209
843f767fffffffff	-1408233591	7147
843f765fffffffff	90310	428
843f765fffffffff	-1408233591	6472
843f763fffffffff	1408236197	119
843f763fffffffff	90310	1296
843f763fffffffff	-1408233591	13932
843f761fffffffff	1408236197	20

Figure 5.5: Sample data from the *amount table* in resolution 4, sorted by h3id. Image source: [McD+23].

Resolution	Number of Hexagons	Average Hexagon Area (km²)	Sensor-location-pairs per data set		Average Number of msg. per sensor-location-pair		Test time (s) for 200 000 entries	
			OpenSky	FlightRadar	OpenSky	FlightRadar	OpenSky	FlightRadar
2	5870	86 801.78	9963	26 133	208 721.51	535.43	0.549	0.586
3	41 150	12 393.43	32 339	65 220	64 302.93	214.54	0.602	0.678
4	288 110	1770.35	143 143	223 018	14 527.38	62.74	0.717	0.806
5	2 016 830	252.90	776 226	758 857	2678.98	18.44	1.350	1.322
6	14 117 870	36.13	4 442 948	1 948 721	468.04	7.18	5.138	2.483
7	98 825 150	5.16	25 086 048	3 741 523	82.89	3.74	29.049	4.438

Table 5.2: Number and size of hexagons for resolutions 2 to 7 in the H3 geospacial index and the respective amounts of sensor-location pairs, average number of messages per sensor-location-pair and the test time the LoVe scheme required to test 200 000 entries. Adapted from [McD+23].

that several different sensors recorded messages in the same cell, but received very diverging numbers of messages: in the case of the second h3id, which appears three times in the data set, the sensor 1408236197 in the second row captured only 38 messages in this cell, while the fourth row shows the sensor -1408233591 captured 15.831 messages in the same cell. This can likely be accredited to the sensors being located in different areas, and therefore having only very limited overlapping reception.

5.3.3 Verification

The approach LoVe takes to verify the origin of an incoming message consists of these steps:

1. **Transform** incoming latitudinal and longitudinal coordinates to an h3id.
2. **Verify** whether the sensor-location-pair exists in the amount-table.
3. If not, **check** whether the sensor exists at all in the data set.

If the second check fails and the received coordinates are spoofed and not within the usual range of coordinates, the message is considered illegitimate.

To test the approach, a labelled test set for both the OpenSky and the FlightRadar data set were required. The **OpenSky network** consists of a crowdsourced network of sensors, and access to the historical database can be acquired by anyone doing research on the data. Acquiring a test set was therefore easy, since data from another day, in this case July 24th 2021, could be downloaded and used. From this, 100 000 random entries were retrieved and labelled true. To generate false test data, the following approach was used: For every sensor, all maximum and minimum latitudes and longitudes ever received were collected. Then, a random float, uniformly distributed between 0.1 and 10, was added or subtracted from or to the respective maximum or minimum latitude or longitude, depending on a random boolean. This simple fuzzing approach ensures that only coordinates outside the expected range can be found in the data labelled false.

FlightRadar on the other hand is a commercial website that offers live airtraffic monitoring. While the trial data of one day was offered up for free, a sample of a second day would have cost 1500€. So instead, 100 000 random entries were extracted, added to a test set and labelled true, and deleted from the original data set. The false data was created using the same approach as for the OpenSky data.

Using these test data sets, verification tests were performed for H3 resolutions 2 to 7. The results demonstrate that the LoVe approach performs reliable classification, which will be elaborated on in the following.

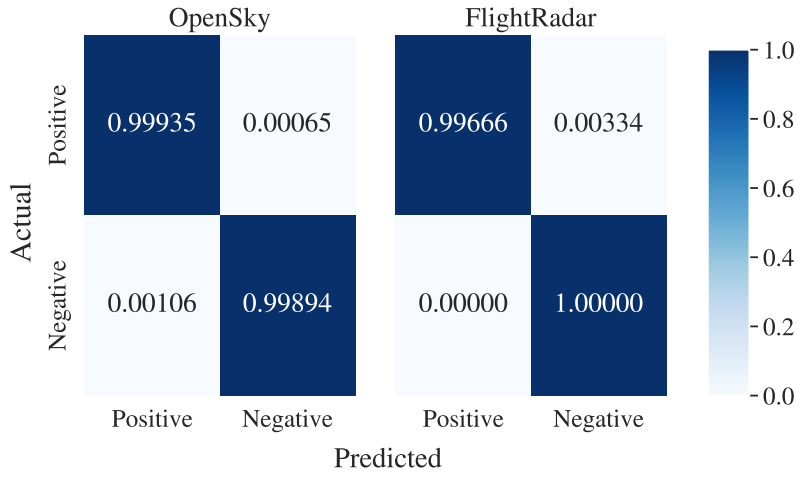


Figure 5.6: Heat map comparing the test runs with labelled test data sets of both the OpenSky data (left) and FlightRadar data (right), using the LoVe approach and H3 resolution 4. Image source: [McD+23].

5.3.4 Verification Evaluation

As described previously, the LoVe approach was tested using labelled data. The false positive rates for both FlightRadar and OpenSky tests lie between 0 and 0.001 06, while the false negative rates are between 0.000 65 and 0.003 34, as is also depicted in Fig. 5.6. The tests yield the best results at resolution 4, since both higher and lower resolutions result in higher false positive and false negative rates.

The time required for testing 200 000 entries depends on the resolution and spans between 0.586 and 4.438 seconds in case of the FlightRadar data set, and between 0.549 and 29.049 seconds in case of the OpenSky data set. Exact results can be gathered from Table 5.2. From the table, it also becomes apparent that the larger the table size, the higher the execution time; the surprisingly high execution time that the OpenSky data set requires in resolution 7 can be explained by the large number of sensor-location-pairs in this resolution, which amounts to 25 million. The relative execution time and the respective accuracy are depicted in Figure 5.7. This figure illustrates that the relative execution time rises steeply the higher the resolution, while the accuracy diverges towards 1 for resolutions 3 and 4, and is lower for the resolutions below and above. These results confirm that resolution 4 yields the best accuracy while requiring comparatively little time. When comparing the OpenSky and FlightRadar data sets, it also becomes apparent that the solution scales very well: A ten times higher number of sensors only marginally increases the computation time, as the focus of the evaluation does not rely on message content, but instead on sensor-location-pairs.

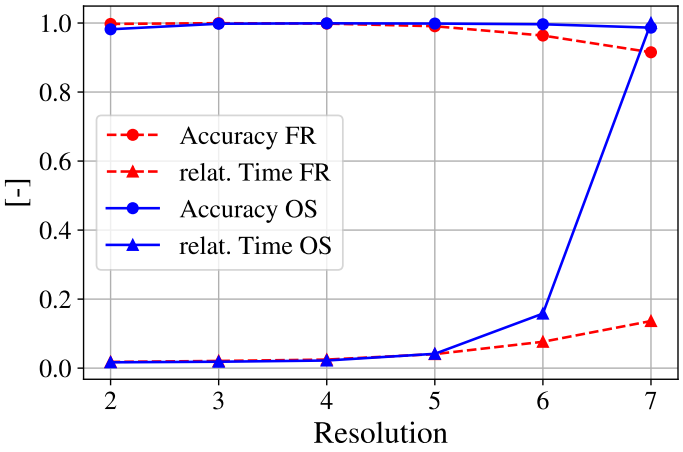


Figure 5.7: The accuracy and relative time of the tests conducted with both the FlightRadar (abbreviated FR and drawn in red) and OpenSky (abbreviated OS and drawn in blue) data sets with respect to the resolutions 2 to 7 [McD+23].

5.3.5 Comparison to ML-Baseline

In order to establish a speed comparison between the LoVe implementation and a machine learning approach we chose the **k-Nearest-Neighbour (kNN) classification**. This classifier is well known, often and easily used and has been chosen as a basis for location verification algorithms for ADS-B messages before [SLM15a]. As in the implementation of the LoVe algorithm, we use batch learning mode, i.e. an entire set of of training examples used offline. This training set contains one million vectors of previously labelled entries. The labels used for the output y are $\{0, 1\}$, where 0 denotes an invalid location recorded by a sensor, whereas 1 is a coordinate within the legitimate coordinate range of a sensor. The input vector X consists of sensor ID, latitude and longitude. In order to have a good selection of neighbours, eight test runs were made with the following values for $k = 5$, 15, 50, 75, 100, 125. The model was created by using a data set containing 200 000 vectors that were known to be valid sensor readings. The **false negative rate** varied from 4.29 %, with 8581 sensors and $k = 5$ neighbours, down to 3.2 %, with 6400 sensors and $k = 100$ neighbours. Due to the nature of the k-Nearest-Neighbours, slow runs were to be expected. Times required for each execution varied from approximately 120 seconds up to 137 seconds which is more than **50 to 100 times slower than our implementation** on such a small data set, as can be seen in Figure 5.8.

The figure shows the times required per run with each different amount of neighbours. As no change occurs in the LoVe implementation, a line is visible at around two seconds. The optimal value for k, reaching the highest accuracy while simulta-

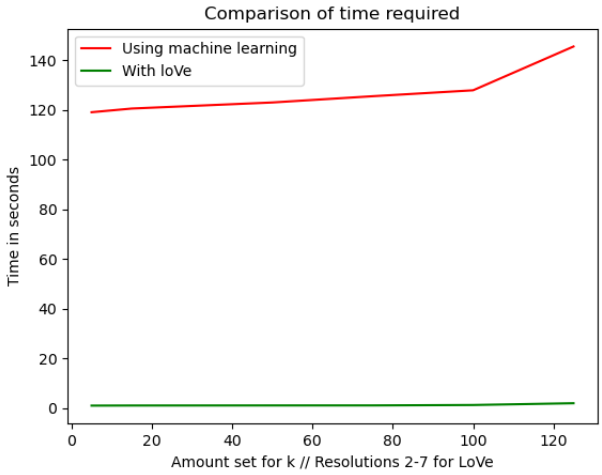


Figure 5.8: Comparison of the time needed for KNN in comparison to LoVe.

neously requiring an acceptably low execution time would have been either $k = 100$ or $k = 125$. For clarity all executions are displayed in the figure.

To compare LoVe to another ML approach, the well known and often used ML-classifier **Support Vector Machine (SVM)** was additionally chosen. An entire set comprising training examples was used offline in batch learning mode. While the same training data set was used as for LoVe, the discerning features sensor ID, latitude and longitude were scaled from 0 to 1. The SVM was additionally parametrised with the values obtained via a hyperparameter search, with gamma=4641 and c=10. As the training time of an SVM grows quadratically with respect to the number of training samples, training it required approximately 48 hours. After training the classifier, it was tested against a subset of the FlightRadar data set, comprising 200 000 data records. 832 of the records were misclassified as **false negatives**. In comparison to LoVe, this is an increase of the false negative rate of 0.004 91. Both the false positive rate, as well as the true negative rate remain 0.0, respective 1.0.

Running the tests of 200 000 samples required 91.3 minutes, which is an **increase by more than a factor of 6000** in comparison to LoVe.

5.3.6 Attack Detection and Comparison to Other Solutions

LoVe can detect three kinds of attacks: an ADS-B message injection attack with spoofed coordinates, a GNSS-spoofing attack and an attack on the sensor network. **Injecting ADS-B messages** into a system with distributed sensors is, on the one hand, trivial, since these message can be transmitted from any location, and the

lack of authentication means that any message is expected to be trustworthy. A ghost plane injected into the system, transmitting coordinates that lie within the transmission range of the attacker's position, will not be detected by the system, since the coordinates would lie within the legitimate range the surrounding sensors normally receive. As soon as the attacker attempts to "move" the ghost plane, the coordinates transmitted by the attacker and those within the legitimate range of the sensor will not overlap anymore, causing the ADS-B messages to be flagged. And as an attacker is unable to move at the speed of an aircraft while staying on ground, and dynamically switching between a large numbers of transmitters on the ground to ensure that the sender position matches that of the transmitted coordinates is out of scope for computationally and financially bounded attackers, LoVe is able to identify ghost plane transmissions.

A **GPS spoofing attack** on the other hand is not a direct attack on the ADS-B system, but rather an indirect attack, since spoofed GNSS coordinates cause the ADS-B system to receive and transmit the spoofed coordinates as well: As explained in Section 2.4.3, it is possible to inject spoofed GNSS coordinates into civilian GNSS systems using an SDR, which would in turn cause the mode S transponder to transmit the spoofed coordinates via ADS-B messages. LoVe can detect spoofed coordinates since a sensor receiving an ADS-B message ranging outside it's normal receiving window would flag the messages as improbable due to previously unobserved coordinates.

The last attack that LoVe can detect is an **attack on the sensor network** itself. If a sensor-location-pair could not be detected in the database, this could either mean that a sensor is receiving data containing coordinates outside its range, or that a previously unknown sensor is inserting messages into the system. In the latter case, the sensor could be a legitimate and newly added sensor, and while it is possible to manually add it to the database, it is considered future work in both this thesis and the publication to develop a secure way of adding new sensors. It could, on the other hand, also be an illegitimate message inserted by a malicious party and intended to inject spoofed ADS-B messages into the system without actually transmitting ADS-B messages. While an injection of messages into the system using an unknown sensor ID would be flagged immediately due to the third step of the verification process, an attack using an existing sensor ID would only be detected if the inserted coordinates were outside the legitimate sensor range.

With the limitation of the coordinates of the injected messages having to be outside the legitimate range of the sensors *or* the sensors having to be unknown to the system, LoVe is able to detect message injection and modification attacks, GPS attacks and attacks on sensors. Since it does not modify the ADS-B protocol, it can not prevent eavesdropping or jamming. Table 5.3 shows a comparison of LoVe to the other attacks

	Cryptographic Protection		Signal Fingerprinting & Protection			Origin Verification		
	[WHE14]	[CF12]	[RK17]	[LGF17]	[PSC09]	[SLM15a]	[Jan+21]	LoVe
Eavesdropping	✓	✓	-	-	✓	-	-	-
Message Injection	✓	✓	✓	✓	✓	✓	✓	✓*
Message Modification	✓	✓	✓	✓	✓	✓	✓	✓*
Jamming	-	-	-	-	✓	-	-	-
GPS Spoofing	-	-	-	-	-	✓	✓	✓*
Attacks on Sensors	-	-	-	-	-	-	-	✓

*: As soon as the transmitted coordinates are outside the normally received coordinate range

Table 5.3: Comparison of the attacks that previously suggested approaches can defend against, including LoVe and various publications included in Section 5.2.

introduced in Section 5.2: Using **cryptographic protections** as suggested by Wesson et al. [WHE14] and Costin et al. [CF12], it is possible to prevent eavesdropping and to detect injected or modified messages, but jamming, GPS spoofing or attacks on the sensors are impossible and out of scope for cryptographic solutions, as these just strive to protect the ADS-B signal itself.

The **signal fingerprinting solutions proposed** by Riahi Manesh et al. [RK17] and Leonardi et al. [LGF17] attempt to detect modifications by calculating device signatures on the signal itself, the latter combining it with classification using a neural network. Since this is, again, applied directly to the signal emitted by the aircraft, these solutions can only detect message injection and modification attacks. Pöpper et al. [PSC09] on the other hand provide the only solution that allows jamming resistance: They suggest to use an Uncoordinated Direct Sequence Spread Spectrum (UDSSS) approach, in which a signed message is spread according to publicly known spreading sequences. The signature ensures authentication, and thereby thwarts attempts at message injection or modification, while the use of uncoordinated signal spreading impedes eavesdropping and jamming. GPS spoofing can not be detected using this approach since the legitimacy of the GNSS signal is assumed, and attacks on sensors are out of scope of this solution as well.

In the field of publications attempting to assess **signal origin legitimacy**, in which LoVe can also be found, Strohmeier et al. [SLM15a] perform location evaluation and location verification using MLAT. This way, it is possible to verify the position of the aircraft in relation to the information transmitted in the ADS-B message. They can detect message injection and modification attacks and GPS spoofing, since their approach multilaterates the location of the aircraft. Both jamming and attacks on sensors are out of scope of this solution. Jansen et al. [Jan+21] use an approach

that is most like LoVe, since it also relies on the use of crowdsourced sensors. They construct vectorsets to analyse ADS-B transmission patterns. A vector consist of all the sensors that received a particular message, which, just like LoVe, forms a basis of legitimately received messages. They then apply ML techniques to evaluate for a newly received message whether the sensor reception pattern matches that of the previously recorded vectors. While the approach can identify both message modification and injection attacks, as well as GPS spoofing, the implementation of sensor reputation, which could detect attacks on sensors, is declared future work. As the modification of the ADS-B protocol is out of scope of origin verification approaches, neither technique can protect ADS-B messages from eavesdroppers.

5.4 Conclusion

The stage of data transfer requires for transmissions to be reliable and available, but also for the message content to be protected from eavesdropping and disclosure, as well as tampering with. Broadcast protocols on the other hand have their own requirements: While asymmetric cryptography works well in protocols with a large amount of senders and receivers, as it facilitates easier key exchange than symmetric schemes, the amount of messages that have to be transmitted is increased significantly in comparison to plaintext transmission. In the ADS-B protocol, instead of constructing difficult schemes to facilitate key exchange or message authentication, security features were just omitted completely.

ADS-B is an important communication protocol that can help increase air surveillance in unpopulated areas. It's far reach makes it an excellent choice to use in addition to conventional air traffic monitoring. Simultaneously, as it contains no security or privacy features, the whole system is attackable via various trajectories. To mitigate this properly, a complete redesign of the system to include cryptographic protection mechanisms would be required. Since a redesign is costly and unlikely, other protection mechanisms using the signal itself have been proposed in various academic publications: These range from signal fingerprinting via radiometric features, clock skew or software-based transmission patterns, to origin verification using MLAT, ML-based approaches or data fusion. An additional approach is Location Verification (LoVe) via distributed public sensors, as introduced in this chapter: LoVe is a mask-based approach that maps previously received signals using the H3 hexagonal indexing system and evaluates newly received signals against them to determine their plausibility. It consists of one database table per resolution. The evaluation shows that resolution 4 yields the best consensus between comparison speed and low false negative rate.

LoVe is a lightweight scheme as it omits operating on message content. Because of this, even a large volume of messages can be represented without significant computational overhead: only the total count per sensor-location-pair is recorded, and a high count increases sensor trust. The scheme is also flexible, allowing for new sensors to be easily added to the database. Adding new sensors does not require costly retraining of models.

Various approaches using multilateration or machine learning for location verification require a large sensor density; MLAT typically requires for at least four sensors to receive the same signal. Some publications attempt to perform MLAT with a reduced amount of sensors, while still achieving good results [DAP20; SLM15a]. Other publications [Jan+21] perform machine learning on a vector set constructed using preferably large set of sensors to receive every single message. These solutions all have in common that they require a large amount of sensors to receive the signals. LoVe works well with both a smaller amount of sensors as well as a high sensor density: In the case of FlightRadar with 11 594 sensors, only about 0.18 % of the messages were received by more than one sensor. LoVe is just as applicable in case of high sensor density: The OpenSky data set was recorded by only 971 sensors, with each message received on average by 12.95 sensors. In fact, a large number of sensors does not impact the computational cost of LoVe significantly. The scheme is therefore easily usable on the sensor network as it is right now, and even well applicable in regions with low sensor density. Additionally, since LoVe does not require to know the position of the sensor, but only its reception area, it can maintain location privacy of the sensor.

All these features make LoVe a flexible, lightweight tool capable of identifying messages from illegitimate sensors and recognising location spoofing, message modification, and message injection attacks aimed at inserting illegitimate aircraft locations into the system. This makes it an easy-to-integrate basis for other established schemes.

6

Conclusion and Future Work

The ability of mobile devices to provide wireless communication offers both benefits and risks: On one hand, ubiquitous connectivity supports information exchange for casual use and enhances security-critical processes. On the other hand, a wireless interface introduces a broader range of potential attacks compared to wired connections, exposing a larger attack surface. The previous chapters demonstrated that each phase of communication presents unique challenges, pitfalls and potential vulnerabilities.

In the phase of **device discovery** in Wi-Fi networks, mobile devices emit probe requests to discover nearby known networks. A particularly problematic part of this is that probe requests can be used for device tracking, thus creating a privacy risk for their users. The chapter first provided an overview over the inherent privacy features of mobile operating systems. Then, device discovery was observed from both the attacker's as well as the defender's perspective: A field study revealed a wealth of private information accidentally contained within the SSID field of a large number of probe requests, which can be used to infer information on their users, and for tracking purposes. An evaluation of previously proposed tracking techniques additionally showcased the need to improve device discovery. The mitigations that were proposed in reaction to both the field study and previous attacks are manifold: One suggestion previously proposed by other publications is to cease the use of active discovery and instead improve passive discovery to be equivalent in terms of speed. Even though it would be the most effective technique to prevent device tracking via probe requests, active discovery (i) remains the favoured technique for device discovery chosen by manufacturers, and (ii) is required when attempting

to connect to a hidden network. To accommodate the latter, a privacy-friendly and tracking resistant means of hidden network discovery was proposed: A hash-based scheme for covert SSID transmission. With respect to the former, three additional approaches could help improve device privacy and reduce tracking: Both the use of generic probe requests and that of a generic MAC address would increase the sizes of the anonymity sets individual devices are a part of, and thereby significantly reduce their aptitude for tracking. The third proposal suggests an enhanced user interface featuring advanced Privacy-by-Design techniques by default, along with increased user control to provide users with greater oversight of their device's behaviour.

Challenges and pitfalls inherent to the second communication phase, in which **connection establishment** occurs, are illustrated using the example of the use of VPNs in public Wi-Fi networks. Here, deprecated protection mechanisms such as WEP and WPA contain severe inherent vulnerabilities, and open networks feature no means of encryption. In both cases, attackers can monitor the traffic and potentially gain insight into sensitive information. Traffic protected via WPA2 can be decrypted in case the attacker has knowledge over the pre-shared key and can capture the 4-way handshake exchanged during connection establishment, or via the KRACK attack in case they have no knowledge over the pre-shared key. OWE is a protocol for unauthenticated but encrypted communication, which is the new de-facto standard for public Wi-Fi hotspots in WPA3. Here, the use of a Diffie-Hellman key exchange makes passive attacks infeasible, but an active attack via a fake access point bearing the same SSID is still possible. And an access point set up using WPA3 and a publicly shared password is as of 2024 infeasible to attack via passive attacks, but the protection mechanisms can be circumvented via a downgrade attack or fake access point attacks. This shows that using public Wi-Fi hotspots carries the risk of packet sniffing and decryption. While older protocols are vulnerable to passive attacks, even current protocols can be attacked, though primarily via active attacks.

To reduce the risk of information leaks in public Wi-Fi networks, a popular approach is to employ a VPN. It encrypts the traffic prior to transmission and tunnels all data via the VPN servers. In theory, both the content as well as the metadata of traffic generated on a client device employing a VPN is therefore concealed from attackers surveilling the network the device is connected to. In practice, information leakage can compromise the data confidentiality, and implementation deficits can lead to captive deadlocks within public networks employing a captive portal. This dissertation therefore proposes a mitigation strategy for the implementation of VPN bootstrapping: A selective VPN bypass to prevent captive deadlocks. The implementation of this strategy would ensure both usability of VPNs in public Wi-Fi networks, as well as leakage-free VPN bootstrapping, thereby ensuring all requirements for safe use and functionality of VPNs in public Wi-Fi networks.

In the stage of **data transfer**, a particular focus of this dissertation lay in the transmission of ADS-B signals. They are used to continuously broadcast aircraft information, allowing both surveillance towers as well as other aircraft to record and map their relative position. This protocol was devised for transmission via a Mode S transponder, which is already present in many aircraft. Only slight modification were required to enable it to additionally transmit ADS-B messages. The protocol was therefore required to fit the ADS-B messages into 112-bit structures, and as the original Mode S messages contain no means of encrypting or signing the content, neither does ADS-B. Research conducted in the field of ADS-B security proposed several protocol changes to include security measures into the 112-bit packet structure. However, protocol changes are unlikely, leading to alternative approaches for signal authenticity verification. By conducting signal verification, either through radiometric fingerprinting or signal-origin verification, security assurances for ADS-B messages can be achieved without altering the protocol. This thesis introduces an additional technique to verify signal integrity: LoVe, a scheme for location verification using distributed public sensors. It can recognise location spoofing, message modification and message injection attacks and is able to identify messages transmitted by illegitimate sensors. Additionally, it is usable in existing sensor networks, and even applicable in areas with low sensor density. In summary, LoVe demonstrates that retrofitting security measures into an unmodifiable protocol is feasible and effective, enhancing the system's overall security.

6.1 Revisiting the Research Questions

Using three protocols as examples, this dissertation illustrated various pitfalls and challenges present in the separate communication phases. They serve to answer the **research questions** posed in Section 1.2, addressing the following topics:

- Privacy friendly device discovery,
- Challenges encountered and overcome during connection establishment,
- Retrofitting security measures to protect the phase of data transfer and
- Knowledge transfer for increased security among different protocol versions.

In the following, the research questions first raised in the beginning of the dissertation are revisited, and their answers discussed in detail.

6.1.1 Research Question 1: Privacy-Friendly Device Discovery

How can device discovery be performed in a privacy-friendly manner? What modifications have to be employed in current protocols to improve the state of the art?

In their current state, the widespread use of probe requests facilitates tracking of the mobile devices that emit them. Additionally, many devices send probe requests that contain sensitive information, which can range from personal details, like names, previously visited locations such as vacation homes or hospitals, to passwords. As such, they are unfit for privacy-friendly device discovery. Chapter 3 of this thesis proposed four means of improving the state of the art:

The most privacy-friendly approach to device discovery is **passive discovery**. Here, no probe requests are transmitted; instead, mobile devices monitor beacons from nearby routers. This approach has previously been proposed [Fra+06; WK18; Goo+19] and is reiterated and summarised here within the context of this thesis to emphasise its significance as a privacy-preserving technology. On the other hand, passive discovery can *not* be used to discover hidden networks. This dissertation therefore proposes a privacy-friendly alternative to transmitting the plaintext SSIDs of hidden networks during active discovery: A **hash-based SSID transmission scheme**, in which the SSID is hashed in conjunction with a salt consisting of the MAC address and the sequence number of the packet. This way, an attacker observing the hashed SSID will have no means of linking it to another hashed SSID, unless they gain knowledge of the plaintext.

The reason manufacturers resort to using active discovery instead of passive discovery is not only the increase in speed, but also the additional information gain: Probe responses received during active discovery are also used to perform coarse device localisation, since routers are typically stationary, and the knowledge of at least three surrounding access points in conjunction with the signal strength can be used to perform multilateration of a device. The question is therefore how active discovery can be improved such that its use maintains user and device privacy as best as possible, and significantly impedes tracking. An approach this dissertation explores concerns the reduction of information contained within probe requests to the point it becomes infeasible to distinguish probe request by their content. This can be achieved by the use of **generic probe requests**, which are content reduced such that the information element contains no more information than strictly required to receive probe responses. This approach was explored with respect to its impact on security and privacy and with respect to maintaining functionality. The results indicate that it is both feasible, with no adverse impact on functionality, and an effective way to enhance privacy and security, maximising anonymity sets and making devices less susceptible to known attacks.

The last approach to enhance privacy in active discovery is **removing potentially identifiable information from the MAC address**: Due to the lack of standardisation of MAC address randomisation, many manufacturers' implementations do not sufficiently prevent device tracking or information inference. An alternative to MAC address randomisation was explored, tested, and found to provide comparable connection establishment times but a greater privacy benefit than inadequately implemented, established randomisation schemes.

In summary, there are **three methods for conducting privacy-friendly device discovery**:

- Utilising passive discovery.
- Implementing active discovery with reduced and generic content during undirected probing.
- When using hidden networks, a privacy-friendly way of performing device discovery is to use the hash-based SSID transmission scheme; hashed SSID transmissions are not required for undirected probe requests.

The modifications required to enable the privacy enhancements on the AP and the probing device vary: For the hash-based scheme, both the protocols implemented on the mobile device and the routers have to be modified to support the comparison of hashed SSIDs. For active discovery with reduced and generic content, the modifications only have to be implemented on the mobile devices, and their amendment does not impact the functionality of active discovery. Regarding the use of a generic MAC address, protocol modifications have to be made on mobile devices to utilise one generic address instead of MAC address randomisation. The proposal was tested and shown to have no negative impact on either the probing behaviour or the responding behaviour. In conclusion, all suggested mitigations were analysed and found to be applicable for improving the current State-of-the-Art, providing privacy-friendly device discovery.

6.1.2 Research Question 2: Challenges of Connection Establishment

What pitfalls can endanger the phase of connection establishment? How can they be overcome while preserving the functionality?

To reduce mobile data use, users tend to utilise public Wi-Fi networks with their Wi-Fi capable mobile devices. Depending on the technology used to protect the network, it is possible for attackers to either directly monitor unencrypted traffic or decrypt the traffic with knowledge of the pre-shared key. While WPA3 support is mandatory

for all devices built after July 2020 that request Wi-Fi certification from the Wi-Fi Alliance [Ebb20], its predecessor protection schemes are still widely used, and the flaws they exhibit during connection establishment have a continuous impact. WPA2, for example, can be **passively attacked**, particularly if monitored during the 4-way handshake exchanged during connection establishment. An attacker capturing this handshake can **decrypt the subsequent encrypted communication**. Passive attacks on WPA3 and OWE are infeasible as of today, but both protocols can be attacked using **active attacks by setting up fake APs**. Hence, the use of public Wi-Fi networks remains dangerous, regardless of the technology employed, and the connection establishment phase is particularly vulnerable to **encryption key disclosure**.

In order to nevertheless utilise public Wi-Fi networks, users can protect both their metadata as well as the content of their communication by using a VPN. Theoretically, all traffic is thereby encrypted and transmitted via a VPN tunnel. In practice, however, traffic leaks occurring particularly surrounding the phase of connection establishment reduce the trust placed into VPNs, and a significant number of VPNs could be shown to be unusable in public Wi-Fi networks due to their lacking support for captive portal detection. The dangers arising during the use of a public Wi-Fi network are therefore manifold: Both **compromised encryption keys** as well as **data leakage during VPN bootstrapping** and **captive deadlocks** endanger the safe use of public Wi-Fi networks in conjunction with a VPN. Since the dangers inherent to the use of public Wi-Fi networks can not be mitigated, VPN bootstrapping has to be improved, as suggested in Chapter 4: A selective VPN bypass during bootstrapping allows for captive portal detection and remediation, and the complete blockage of all other traffic until successful VPN establishment ensures that no leakage occurs. The selective VPN bypass is therefore an **adequate method for securing connection establishment while preserving privacy and functionality**, allowing for secure subsequent data transfer even in unsecured public Wi-Fi networks.

6.1.3 Research Question 3: Retrofitted Security

How can security measures be retrofitted into protocols, and secondly, if protocol changes are impossible, what other means of securing the communication and its authenticity can be made? What means can be used to protect unencrypted and unauthenticated data transfer from modifications? How can modifications be detected in a protocol lacking integrity protection?

The large number of ongoing IT security incidents demonstrates that security and privacy features are often lacking or absent in protocol design. However, many protocols can be modified to support security and privacy features and thereby improve the overall system security. In the case of active discovery, for example, the

use of generic probe requests would not, in fact, require a protocol change, but only a reduction of content to unify and generalise the structure and content of probe requests. Another example of retrofitted security measures could be observed in 2014 and 2015, where both Android and Apple implemented MAC address randomisation to protect users from tracking via their hardware MAC address. Both measures, as well as the use of a generic MAC address in probe requests, can be implemented without protocol modifications, since they fit into the previously used protocol, and the **changes do not impact the general system functionality while having a positive impact on security**.

In general, retrofitting security measures is particularly successful and likely to occur if the modifications can be integrated **without changing the protocol**. Therefore, the adaption of the hash-based scheme for covert SSID transmission is unlikely to be implemented: Even though the scheme does not require a modification of probe request packets and the hashed SSID fits into the current SSID slot, both mobile devices as well as routers would have to be modified to adapt hashing and the comparison of hashes. While the scheme is an adequate and good means of improving the current state, its implementation is unlikely. The best way to increase privacy in the context of hidden networks is to avoid using them, as this eliminates the need to transmit SSIDs.

The insecurities surrounding the use of public Wi-Fi networks can not easily be amended with retrofitted security mechanisms: While the vulnerabilities included in former protocol versions were mitigated with each successive protection scheme, the protocols are in such widespread use and adhere to such strict standards, that all modifications, even minor ones, are difficult to employ. Therefore, vulnerabilities like the possibility to infer the PTK from the 4-way handshake in WPA2 will still be in place for as long as WPA2 is in use. When using a public Wi-Fi network, users must rely on their own security mechanisms, such as a VPN. Here, the **absence of strict standardisation and lack of certification** result in an environment that can **adapt to vulnerabilities** much more easily, but is also more prone to individual implementation errors. This is demonstrated in the lack of support for captive portal detection in the native VPN APIs, and much more so since traffic leakage outside the tunnel is very widespread, as could be shown in Section 4.6.5. To mitigate both the leakage as well as the inability of many VPN clients to support captive portal detection, a selective VPN bypass was suggested. Since this significantly improves both the security properties and the usability of VPN clients in public Wi-Fi networks, both are likely to be implemented in commercial as well as native VPN clients.

The ADS-B protocol is an example for a broadcast protocol with no inherent security features, but with the same requirements for protocol stability as Wi-Fi networks. In fact, no protocol updates have been made since the original release, and protocol

changes are unlikely to happen: The installation of ADS-B transmitters is mandatory for certain aircraft (cf. Section 2.4), and installing an alternative transmitter is not desirable from the perspective of manufacturers, since it has no additional commercial value. Measures to increase system security of the ADS-B protocol **should not entail either modifications of the installations nor the protocol itself**. This eliminates the introduction of cryptographic protection mechanisms, since they would entail protocol modifications. The protection mechanisms still applicable in this scenario therefore concern signal fingerprinting and verification approaches. This dissertation introduces an approach for location verification using distributed public sensors. It utilises a mask of legitimate signals, and evaluates newly received messages as to their alignment with the mask. This way, **despite the absence of integrity protection in the ADS-B protocol**, it is possible to detect illegitimate sensor transmissions as well as location spoofing, message modification and message injection attacks. The approach is particularly feasible for use, since it can be utilised on existing public sensor infrastructure, does not require a high sensor density and has a very good detection rate of illegitimate signals. It is therefore an **ideal means of retrofitting security mechanisms** into the ADS-B protocol, and asserting signal legitimacy in the face of a protocol void of both confidentiality and authenticity guarantees.

In summary, **it is possible to retrofit security mechanisms** into the schemes explored in this dissertation. The modifications which are most likely to be implemented have a large impact on security or privacy, while simultaneously having no negative impact on the system's functionality. And even in the case of largely unmodifiable protocols like ADS-B, it is possible to retroactively introduce certain security measures; however, others, such as confidentiality, **cannot be retrofitted without modifying the protocol**. The more standardised a protocol is, the more challenging it becomes to introduce mitigations for vulnerabilities and add additional security measures. Simultaneously, less standardised and uncertified protocols, such as the tested VPNs, are more error-prone but are also more likely to have vulnerabilities addressed in the short term.

6.1.4 Research Question 4: Knowledge Transfer

Are the "lessons learned" from older protocols transferred to newer versions of the protocol, and which elements are particularly critical?

This question of how the knowledge gained from existing attacks and through various protocol iterations has contributed to the improvement of protection schemes can only be addressed in relation to the longevity of the protocol in question, as well as its requirements for standardisation and certification.

In the context of probe requests, research exposing the privacy implications in former versions quickly provoked changes, with the previously used UAA being replaced by a randomised MAC address, and the SSID being omitted if possible. These changes neither concern the packet structure, nor do they impact functionality, and can therefore be freely made by manufacturers. Changes to the packet structure however, would have been unacceptable in the stable ecosystem of the Wi-Fi standard. This can also be seen in the development of Wi-Fi protection schemes: Mistakes made in WEP and WPA were mitigated in subsequent versions, but the longevity of the standard and the requirement to provide downgrading functionality to assert the functionality of older devices ensures that many previous vulnerabilities can still be exploited today. An example for this is the possibility to extract the PTK by monitoring the 4-way-handshake: This attack is infeasible in WPA3, but in the connection with routers providing WPA3 in *transition mode*, an attacker can perform a downgrade attack, and thereby force users to connect via WPA2 instead. In answer to the research question: Yes, the "lessons learned" are transferred from older protocols, but to **maintain long-term functionality**, the continuous provisioning of the previous, vulnerable protocols typically **remains an attack vector**.

With respect to the ADS-B protocol, this question can be answered very quickly: No. Despite the lack of security mechanisms in the ADS-B protocol, **changes are unlikely to occur**, since modifications of this partially mandatory technology might have to entail the installation of new equipment. While it was initially conceived as an alternative to PSR, its lack of security guarantees makes it unfit for this task, unless used in conjunction with other techniques to establish signal or positional verification. Therefore, **a protocol superseding ADS-B** will likely utilise the positive aspects, like the frequency spectrum allowing long-range communication and location propagation even in remote areas, while simultaneously featuring a different packet structure, which would include enough space to preserve integrity and ensure authenticity, and potentially even provide confidentiality.

In summary, the lack of confidentiality and integrity protection for ADS-B messages is not seen as critical to system functionality, and potentially even as a **feature** of the system. The main attention this lack of security receives is of academic nature. But since it simultaneously allows for various attacks, it should be taken seriously and *not* discarded lightly, particularly in the design of subsequent protocols for air surveillance. However, the stability of the protocol and the long-term use of transmitting devices complicate protocol modifications. With respect to probe requests on the other hand, the devices in question are exchanged much more frequently, with some users replacing their mobile devices every two years or even more frequently. Because of this, modifications on the probe request content have a large impact within a comparably short time span. Simultaneously, the privacy implications of the transmission of probe requests were critical and therefore required direct at-

tention. Taking just these two examples into account, it could be conjectured that direct privacy implications for individual users cause rapid reactions, while security implications that can be controlled by e.g. by using additional technology, in the case of ADS-B the PSR system, do not provoke mitigations.

6.2 Outlook on Future Work

This section provides an outlook on possible future work that could extend the research presented in this dissertation, and could improve system security in all covered protocols.

6.2.1 Real-World Evaluations and Regulatory Possibilities

Future research into the area of probe requests could implement the hash-based scheme for covert SSID transmission and evaluate its feasibility in a real-world scenario. This should also be done for generic probe requests. Exploring the feasibility of regulatory measures to mandate content reduction schemes from a legal perspective could enhance device privacy in the whole field of active discovery.

6.2.2 Improvements over Time

Concerning the chapter on VPN usage, it would be interesting to explore the modifications undertaken in response to the disclosures of data leakage and the inability to perform captive portal detection, both in native clients as well as commercial clients.

6.2.3 Protocol Re-Design

With respect to ADS-B messages, instead of improving the current system, future work should design a whole new protocol for location transmission for air surveillance. This system should include means of asserting message integrity and authenticity. Since these are of highest priority, risk assessment would have to evaluate the additional need to provide confidentiality. In this context, the question of key distribution and key access is particularly interesting, seeing that a multitude of parties would require access to the propagated messages, and that the protocol would have

to include means of replacing compromised encryption keys and propagating them, while using preferably small data frames.

List of Publications

The following list presents the publications which were published during the writing of this thesis, and the parts of the dissertation they are discussed in.

1. Christian Burkert, Johanna Ansohn McDougall, Hannes Federrath and Mathias Fischer. Analysing Leakage during VPN Establishment in Public Wi-Fi Networks. In: ICC 2021 - IEEE International Conference on Communications. 2021, pp. 1–6. doi: 10.1109/ICC42927.2021.9500375.

   This paper is discussed in Chapter 4.

2. Johanna Ansohn McDougall, Christian Burkert, Daniel Demmler, Monina Schwarz, Vincent Hubbe and Hannes Federrath. *Probing for Passwords – Privacy Implications of SSIDs in Probe Requests.* In: *Applied Cryptography and Network Security.* Ed. by Giuseppe Ateniese and Daniele Venturi. Cham: Springer International Publishing, 2022, pp. 376–395. isbn: 978-3-031-09234-3.

   This paper is discussed in Chapter 3.

3. Johanna Ansohn McDougall, Alessandro Brighente, Willi Großmann, Ben Ansohn McDougall, Joshua Stock and Hannes Federrath. *LoVe is in the Air – Location Verification of ADS-B Signals using Distributed Public Sensors.* In: *IEEE International Conference on Communications (ICC).* ICC. 2023.

   This paper is discussed in Chapter 5.

4. Johanna Ansohn McDougall, Alessandro Brighente, Anne Kunstmann, Niklas Zapatka and Hannes Federrath. *Reduce to the MACs - Privacy Friendly Generic Probe Requests.* In: *ICT Systems Security and Privacy Protection, 39th IFIP TC 11 International Conference, SEC 2024 (Edinburgh, UK, June 12, 2024).* Ed. by Steven FFurnell et al. Springer, 2024.

   This paper is discussed in Chapter 3

5. Johanna Ansohn McDougall, Alessandro Brighente, Anne Kunstmann, Niklas Zapatka, Hanna Schambach and Hannes Federrath. *Probing with a Generic MAC Address: An Alternative to MAC Address Randomisation.* In: *2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM).* 2024.

   This paper is discussed in Chapter 3

An additional publication to which I contributed during the writing of this thesis, but which is not included in this thesis is this:

1. Christian Burkert, Johanna Ansohn McDougall, Hannes Federrath: *Data Minimisation Potential for Timestamps in Git: An Empirical Analysis of User Configurations.* In: *ICT Systems Security and Privacy Protection. SEC 2022. IFIP Advances in Information and Communication Technology, vol 648.* Ed. by W. Meng, S. Fischer-Hübner, and C.D. Jensen, Springer, Cham, 2022.

List of Supervised Theses

The following list includes the bachelor's theses and studies I supervised while working on this dissertation. The topics were provided by me, in the case of co-supervision, they were provided by both of us. Since most theses were completed in German, their titles have been translated into English for this enumeration.

1. Multisniffer - Conceptual design, implementation and evaluation of a monitoring tool for IoT protocols. Master's thesis, October 2019. *The student did not finish this thesis. Co-supervision with Monina Schwarz.*
2. Analysis and evaluation of data protection mechanisms and security of smart toys. Bachelor's thesis, January 2020.
3. Secure VPN-bootstrapping in unencrypted Wi-Fi hotspots. Bachelor's thesis, June 2020. *Co-supervision with Christian Burkert.*
4. Patching of vulnerabilities in IoT devices in critical infrastructure. Bachelor's thesis, February 2021. *Co-supervision with Marius Stübs.*
5. Personal data in Signal: Infrastructure and data analysis of the messaging application (Co-supervision). Master's Thesis, January 2022. *The student did not finish this thesis. Co-supervision with Tobias Müller.*
6. Security measures for SSIDs to enable the privacy friendly use of hidden networks. Master's thesis, April 2022. *The student did not finish this thesis.*
7. Security Mechanisms and Attacks in WPA2 and WPA3, Bachelor's thesis, October 2022.

8. Conception, implementation and evaluation of a leakage-analyser for probe requests on mobile devices. Bachelor's thesis, December 2022. *Co-supervision with Christian Burkert.*

9. Generic-Address-Scheme for the privacy friendly use of probe requests. Study, March 2023. *The paper [McD+24a] is based on the results of this study.*

Bibliography

[Aar14] Aaron Mamiit. *Apple implements random MAC address on iOS 8. Goodbye, marketers*. https://www.techtimes.com/articles/8233/20140612/apple-implements-random-mac-address-on-ios-8-goodbye-marketers.htm. 2014. (Visited on 06/14/2014).

[Ali+19] Suzan Ali et al. *On Privacy Risks of Public WiFi Captive Portals*. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Ed. by Cristina Pérez-Solà et al. Cham: Springer International Publishing, 2019, pp. 80–98. ISBN: 978-3-030-31500-9.

[All20] Wi-Fi Alliance. *Opportunistic Wireless Encryption Specification - Version 1.1*. Tech. rep. 2020. URL: https://www.wi-fi.org/system/files/Opportunistic_Wireless_Encryption_Specification_v1.1.pdf.

[And] Android Developers. *VPN*. URL: https://developer.android.com/guide/topics/connectivity/vpn (visited on 06/26/2024).

[And23] Android Developers. *Android 6.0 Changes*. https://developer.android.com/about/versions/marshmallow/android-6.0-changes. 2023. (Visited on 10/25/2023).

[Appa] Apple. *Entitlement*. URL: https://developer.apple.com/documentation/bundleresources/entitlements (visited on 06/26/2024).

[Appb] Apple. *Network Extensions Entitlement*. URL: https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_developer_networking_networkextension (visited on 06/26/2024).

[Appc] Apple. *Packet Tunnel Provider*. URL: https://developer.apple.com/documentation/networkextension/packet_tunnel_provider (visited on 06/25/2024).

[Appd] Apple. *Personal VPN*. URL: https://developer.apple.com/documentation/networkextension/personal_vpn (visited on 06/25/2024).

[Appe] Apple. *VPN On Demand Rules*. URL: https://developer.apple.com/documentation/networkextension/personal_vpn/vpn_on_demand_rules (visited on 06/28/2024).

[Appf] Apple Platform Deployment. *VPN overview for Apple device deployment*. URL: https://support.apple.com/en-gb/guide/deployment/depae3d361d0/1/web/1.0 (visited on 06/28/2024).

[Ass22] NBAA - National Business Aviation Association. *ADS-B Privacy FAQ*. Mar. 22, 2022. URL: https://nbaa.org/aircraft-operations/security/privacy/ads-b-privacy-faq/.

[Aut23] Internet Assigned Numbers Authority. *IEEE 802 Numbers*. 2023. URL: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml#ieee-802-numbers-1.

[Bab10] J. Randolph Babbitt. *Automatic Dependent Surveillance-Broadcast (ADS–B) Out Performance Requirements To Support Air Traffic Control (ATC) Service*. Washington, DC, USA: Federal Aviation Administration, May 28, 2010. URL: https://www.govinfo.gov/content/pkg/FR-2010-05-28/pdf/2010-12645.pdf (visited on 04/11/2024).

[Bar+13] Marco Valerio Barbera et al. *Signals from the crowd: uncovering social relationships through smartphone probes*. In: *Proceedings of the 2013 conference on Internet measurement conference* (2013). URL: https://api.semanticscholar.org/CorpusID:17216849.

[Bar+22] Marco V. Barbera et al. *CRAWDAD sapienza/probe-requests*. 2022. DOI: 10.15783/C76C7Z. URL: https://dx.doi.org/10.15783/C76C7Z.

[Blo+19] Maximilian Blochberger et al. *State of the Sandbox: Investigating macOS Application Security*. In: *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, WPES@CCS 2019, London, UK, November 11, 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, and Josep Domingo-Ferrer. ACM, 2019, pp. 150–161. DOI: 10.1145/3338498.3358654.

[Boo+01] Skip Booth et al. *Securing L2TP using IPsec*. RFC 3193. Nov. 2001. DOI: 10.17487/RFC3193. URL: https://www.rfc-editor.org/info/rfc3193.

[Bro18] Isaac Brodsky. *H3: Uber's Hexagonal Hierarchical Spatial Index*. https://www.uber.com/en-DE/blog/h3/. Accessed: 2024-05-21. 2018.

[Bur+21] Christian Burkert et al. *Analysing Leakage during VPN Establishment in Public Wi-Fi Networks*. In: *ICC 2021 - IEEE International Conference on Communications*. 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500375.

[BZO15] Carlos J. Bernardos, Juan Carlos Zuniga, and Piers O'Hanlon. *Wi-Fi Internet Connectivity and Privacy: Hiding Your Tracks on the Wireless Internet*. In: *IEEE Conference on Standards for Communications and Networking (CSCN)*. 2015 IEEE Conference on Standards for Communications and Networking (CSCN). IEEE, Oct. 28, 2015, pp. 193–198. DOI: 10.1109/CSCN.2015.7390443. URL: http://ieeexplore.ieee.org/document/7390443/ (visited on 11/18/2021).

[CF12] Andrei Costin and Aurélien Francillon. *Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices*. In: July 2012.

[Che+13] Ningning Cheng et al. *Characterizing privacy leakage of public WiFi networks for users on travel*. In: *2013 Proceedings IEEE INFOCOM*. 2013, pp. 2769–2777. DOI: 10.1109/INFCOM.2013.6567086.

[CKB14] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. *Linking Wireless Devices Using Information Contained in Wi-Fi Probe Requests*. In: *Pervasive and Mobile Computing* 11 (Apr. 2014), pp. 56–69. ISSN: 15741192. DOI: 10.1016/j.pmcj.2013.04.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S1574119213000618 (visited on 11/18/2021).

[Dan] Dan Harkins. *Wi-Fi CERTIFIED Enhanced Open™: Transparent Wi-Fi® protections without complexity*. URL: https://www.wi-fi.org/beacon/dan-harkins/wi-fi-certified-enhanced-open-transparent-wi-fi-protections-without-complexity (visited on 06/04/2021).

[DAP20] Ala' Darabseh, Hoda AlKhzaimi, and Christina Pöpper. *MAVPro: ADS-B message verification for aviation security with minimal numbers of on-ground sensors*. In: *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2020, pp. 53–64.

[Dar] Darkaudax. *Airdecap-ng*. URL: https://www.aircrack-ng.org/doku.php?id=airdecap-ng (visited on 07/12/2024).

[Dav20] Corbin Davenport. *Android 11 lets you not automatically connect to specific Wi-Fi networks*. 2020. URL: https://www.androidpolice.com/2020/07/08/android-11-lets-you-not-automatically-connect-to-specific-wi-fi-networks/ (visited on 07/08/2020).

[DM05] D.A. Dai Zovi and S.A. Macaulay. *Attacking automatic wireless network selection*. In: *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. 2005, pp. 365–372. DOI: 10.1109/IAW.2005.1495975.

[Doc24] AOSP Documentation. *MAC Randomization Behavior*. Jan. 31, 2024. URL: https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior (visited on 03/14/2024).

[DPČ19] Ante Dagelić, Toni Perković, and Mario Čagalj. *Location Privacy and Changes in WiFi Probe Request Based Connection Protocols Usage Through Years*. In: *International Conference on Smart and Sustainable Technologies (SpliTech)*. IEEE. 2019, pp. 1–5.

[Ebb20] Philipp Ebbecke. *Protected Management Frames enhance Wi-Fi Network Security*. 2020. URL: https://www.wi-fi.org/beacon/philipp-ebbecke/protected-management-frames-enhance-wi-fi-network-security (visited on 01/10/2022).

[Ebi15] Takuji Ebinuma. *Giters: GPS-SDR-SIM*. 2015. URL: https://giters.com/pistoletpierre/gps-sdr-sim (visited on 05/21/2024).

[Enn] Greg Ennis. *The Early History of the Wi-Fi Logo*. Accessed: 2024-10-11. URL: https://www.gregennis.net/wifi-fun-facts-2/the-early-history-of-the-wi-fi-logo.

[Eur21] European Space Agency (ESA). *GPS General Introduction*. https://gssc.esa.int/navipedia/index.php/GPS_General_Introduction. Accessed: 2024-10-11. 2021.

[Eur23a] European Commission - Committee C07900. *Annex - C(2023)3538 on Designing Mobile Phones and Tablets to be Sustainable (Ecodesign)*. Accessed: 2024-10-29. 2023. URL: %7Bhttps://ec.europa.eu/info/law/better-regulation/have-your-say/initiatives/12797-Designing-mobile-phones-and-tablets-to-be-sustainable-ecodesign_en%7D.

[Eur23b] European Union. *Commission Regulation (EU) 2023/826*. Accessed: 2024-10-29. 2023. URL: %7Bhttps://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32023R0826%7D.

[EUR24] EUROCONTROL. *Automatic Dependent Surveillance – Broadcast*. 2024. URL: https://www.eurocontrol.int/service/automatic-dependent-surveillance-broadcast (visited on 04/11/2024).

[Fen+21] Ellis Fenske et al. *Three Years Later: A Study of MAC Address Randomization In Mobile Devices And When It Succeeds*. In: *PETS'2021* (2021), pp. 164–181.

[Fli24] Flightradar. *GPS jamming map*. 2024. URL: https://www.flightradar24.com/data/gps-jamming (visited on 04/15/2024).

[Fra+06] Jason Franklin et al. *Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting*. In: *USENIX Security'06*. USENIX Association, 2006.

[Fre15] Julien Freudiger. *How Talkative is Your Mobile Device? An Experimental Study of Wi-Fi Probe Requests*. In: *WiSec'15*. ACM, 2015. DOI: 10.1145/2766498.2766517. URL: https://doi.org/10.1145/2766498.2766517.

[Gau23] Matthew Gault. *Commercial Flights Are Experiencing 'Unthinkable' GPS Attacks and Nobody Knows What to Do*. Nov. 20, 2023. URL: https://www.vice.com/en/article/m7bk3v/commercial-flights-are-experiencing-unthinkable-gps-attacks-and-nobody-knows-what-to-do (visited on 05/08/2024).

[GG03] Marco Gruteser and Dirk Grunwald. *Enhancing Location Privacy in Wireless LAN through Disposable Interface Identifiers: A Quantitative Analysis*. In: *Proceedings of the 1st ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. WMASH '03. San Diego, CA, USA: Association for Computing Machinery, 2003, pp. 46–55. ISBN: 1581137680. DOI: 10.1145/941326.941334.

[GGP22] Carlos Andres Gomez, Laura Juliana Guerrero, and Luis Fernando Pedraza. *Evolution of the Use of Random MAC Addresses in Public Wi-Fi Networks.* In: *Journal of Engineering Science & Technology Review* 15.3 (2022).

[Gib23] Samuel Gibbs. *Google Pixel 8 Pro launched with thermometer and seven years of updates*. The Guardian. Oct. 4, 2023. URL: https://www.theguardian.com/technology/2023/oct/04/google-pixel-8-pro-launched-thermometer-cameras-seven-years-of-updates (visited on 04/09/2024).

[GIS13] GIS Resources. *Fundamentals of GPS Signal And Data*. https://gisresources.com/fundamentals-of-gps-signal-and-data_2/. Accessed: 2024-10-11. 2013.

[Goo+19] Frederik Goovaerts et al. *Improving Privacy Through Fast Passive Wi-Fi Scanning*. In: *Secure IT Systems*. Springer, 2019, pp. 37–52. ISBN: 978-3-030-35055-0.

[Gre12] Andy Greenberg. *Next-gen air traffic control vulnerable to hackers spoofing planes out of thin air*. In: *Forbes Magazine. Retrieved September* 10 (2012), p. 2014.

[Gro23] Internet Security Research Group. *A Better Internet: Tenth Anniversary*. Accessed: 2024-10-11. 2023. URL: https://www.abetterinternet.org/tenth-anniversary/.

[Gu+20] Xiaolin Gu et al. *Probe Request Based Device Identification Attack and Defense*. In: *Sensors* 20.16 (2020). ISSN: 1424-8220. DOI: 10.3390/s20164620. URL: https://www.mdpi.com/1424-8220/20/16/4620.

[H323] H3. *Tables of Cell Statistics Across Resolutions*. 2023. URL: https://h3geo.org/docs/core-library/restable/.

[HK17] Dan Harkins and Warren "Ace" Kumari. *Opportunistic Wireless Encryption)*. RFC 8110. 2017. URL: https://datatracker.ietf.org/doc/rfc8110/.

[Hog17] Giles Hogben. *Android Developers Blog: Changes to Device Identifiers in Android O*. Apr. 10, 2017. URL: https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html (visited on 03/06/2024).

[Hon+05] Osamu Honda et al. *Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency*. In: *Performance, Quality of Service, and Control of Next-Generation Communication and Sensor Networks III*. Ed. by Mohammed Atiquzzaman and Sergey I. Balandin. Vol. 6011. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. Oct. 2005, pp. 138–146. DOI: 10.1117/12.630496.

[HTC23] Tianlang He, Jiajie Tan, and Shueng-Han Gary Chan. *Self-Supervised Association of Wi-Fi Probe Requests Under MAC Address Randomization*. In: *IEEE Transactions on Mobile Computing* 22 (2023), pp. 7044–7056. URL: https://api.semanticscholar.org/CorpusID:252232386.

[ICA20] ICAO. *Proposals for the amendment of Annex 10, Volume IV regarding airborne collision avoidance system X (ACAS X)*. Online. Accessed on May 16, 2024. 2020. URL: https://www.icao.int/APAC/Meetings/2020%20SURICG5/Flismy01_Proposal%20for%20Amendment%20ACAS%20X.pdf.

[IEE14] IEEE. *IEEE Std 802-2014 - Local and Metropolitan Area Networks: Overview and Architecture*. https://ieeexplore.ieee.org/document/6847097. 2014. DOI: 10.1109/IEEESTD.2014.6847097.

[IEE20] IEEE. *IEEE Std 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. 2020. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363693 (visited on 04/29/2021).

[Ikr+16] Muhammad Ikram et al. *An Analysis of the Privacy and Security Risks of Android VPN Permission-Enabled Apps*. In: *Proceedings of the 2016 ACM on Internet Measurement Conference - IMC '16*. The 2016 ACM. Santa Monica, California, USA: ACM Press, 2016, pp. 349–364. ISBN: 978-1-4503-4526-2. DOI: 10.1145/2987443.2987471. (Visited on 09/02/2020).

[Jan+21] Kai Jansen et al. *Trust the Crowd: Wireless Witnessing to Detect Attacks on ADS-B-Based Air-Traffic Surveillance*. In: Feb. 2021. DOI: 10.14722/ndss.2021.24552.

[Jona] Jongerius, Jerry. *Wi-fi 1/2/3 – Legacy 802.11*. Accessed: 2024-10-29. URL: %7Bhttps://www.wiisfi.com/#legacy%7D.

[Jonb] Jongerius, Jerry. *Wi-Fi 4 (802.11n) 2.4 GHz and 5 GHz*. Accessed: 2024-10-29. URL: %7Bhttps://www.wiisfi.com/#wifi4%7D.

[Kar17] Rickard Karlsson. *EzMole: A new prototype for securing public Wi-Fi connections*. MA thesis. Luleå University of Technology, Computer Science, 2017, p. 58.

[Kar23] Namrata Karnam. *Windows 10, version 22H2 end of support date updated*. https://learn.microsoft.com/en-us/lifecycle/announcements/windows-10-22h2-end-of-support-update. Accessed: 2024-10-11. Apr. 2023.

[Kau+14] Charlie Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. Oct. 2014. DOI: 10.17487/RFC7296. URL: https://www.rfc-editor.org/info/rfc7296.

[Kau05] Charlie Kaufman. *Internet Key Exchange (IKEv2) Protocol*. RFC 4306. Dec. 2005. DOI: 10.17487/RFC4306. URL: https://www.rfc-editor.org/info/rfc4306.

[Kha+18] Taha Khan et al. *An Empirical Analysis of the Commercial VPN Ecosystem*. In: Oct. 2018, pp. 443–456. ISBN: 978-1-4503-5619-0. DOI: 10.1145/3278532.3278570.

[KK20] Warren "Ace" Kumari and Erik Kline. *Captive-Portal Identification in DHCP and Router Advertisements (RAs)*. RFC 8910. 2020. URL: https://datatracker.ietf.org/doc/rfc8910/.

[Leg16] Deniz Legezo. *Research on unsecured Wi-Fi networks across the world*. Securelist by Kaspersky. 2016. URL: https://securelist.com/research-on-unsecured-wi-fi-networks-across-the-world/76733/ (visited on 05/30/2024).

[LGF17] Mauro Leonardi, Luca Gregorio, and Davide Fausto. *Air Traffic Security: Aircraft Classification Using ADS-B Message's Phase-Pattern*. In: *Aerospace* 4 (Oct. 2017), p. 51. DOI: 10.3390/aerospace4040051.

[Lin23] Weyde Lin. *Tunneling and VPN*. In: *Trends in Data Protection and Encryption Technologies*. Ed. by Valentin Mulder et al. Cham: Springer Nature Switzerland, 2023, pp. 149–154. ISBN: 978-3-031-33386-6. DOI: 10.1007/978-3-031-33386-6_26. URL: https://doi.org/10.1007/978-3-031-33386-6_26.

[Lot+21] Ahmed Lotfy et al. *Privacy Issues of Public Wi-Fi Networks*. In: May 2021, pp. 656–665. ISBN: 978-3-030-76345-9. DOI: 10.1007/978-3-030-76346-6_58.

[LZM18] Huaxin Li, Haojin Zhu, and Di Ma. *Demographic Information Inference through Meta-Data Analysis of Wi-Fi Traffic*. In: *IEEE Trans. Mob. Comput.* 17.5 (2018), pp. 1033–1047. DOI: 10.1109/TMC.2017.2753244.

[Man23] Network Manager. *Network Manager NMSettingVpn - Connection properties for Virtual Private Networks*. 2023. URL: https://networkmanager.dev/docs/libnm/latest/NMSettingVpn.html#nm-setting-vpn-get-persistent (visited on 06/27/2024).

[Mar+17] Jeremy Martin et al. *A Study of MAC Address Randomization in Mobile Devices and When It Fails*. In: *PETS'2017* 4 (2017), pp. 268–286.

[Mata] Paolo Matarazzo. *Microsoft Learn - VPN Auto-Triggered Profile Options*. URL: https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/vpn/vpn-auto-trigger-profile (visited on 06/27/2024).

[Matb] Paolo Matarazzo. *Microsoft Learn - VPN Security Features - LockDown VPN*. URL: https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/vpn/vpn-security-features#lockdown-vpn (visited on 06/27/2024).

[McD+22] Johanna Ansohn McDougall et al. *Probing for Passwords – Privacy Implications of SSIDs in Probe Requests*. In: *Applied Cryptography and Network Security*. Ed. by Giuseppe Ateniese and Daniele Venturi. Cham: Springer International Publishing, 2022, pp. 376–395. ISBN: 978-3-031-09234-3.

[McD+23] Johanna Ansohn McDougall et al. *LoVe is in the Air – Location Verification of ADS-B Signals using Distributed Public Sensors*. In: *IEEE International Conference on Communications (ICC)*. ICC. 2023.

[McD+24a] Johanna Ansohn McDougall et al. *Probing with a Generic MAC Address: An Alternative to MAC Address Randomisation*. In: *2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2024, pp. 1–6. DOI: 10.23919/SoftCOM62040.2024.10721764.

[McD+24b] Johanna Ansohn McDougall et al. *Reduce to the MACs - Privacy Friendly Generic Probe Requests*. In: *ICT Systems Security and Privacy Protection, 39th IFIP TC 11 International Conference, SEC 2024* (Edinburgh, UK, June 12, 2024). Ed. by Steven Furnell et al. Springer, 2024.

[MCT17] Célestin Matte, Mathieu Cunche, and Vincent Toubiana. *Does disabling Wi-Fi prevent my smartphone from sending Wi-Fi frames?* Research Report RR-9089. Inria - Research Centre Grenoble – Rhône-Alpes ; INSA Lyon, Aug. 2017. URL: https://hal.inria.fr/hal-01575519.

[Myr22] Andrew Myrick. *How To Manage Wi-Fi Networks on iPhone in iOS 16*. Dec. 29, 2022. URL: https://appletoolbox.com/how-to-manage-wi-fi-networks-on-iphone-in-ios-16/ (visited on 03/14/2022).

[New22] Samsung News. *Samsung Sets the New Standard of Four Years of OS Upgrades to Ensure the Most Up-to-Date and More Secure Galaxy Experience*. 2022. URL: https://news.samsung.com/us/samsung-galaxy-os-upgrade-one-ui-android-unpacked-2022/ (visited on 04/05/2024).

[Ope] OpenVPN. *Why OpenVPN uses TLS*. https://openvpn.net/faq/why-openvpn-uses-tls/.

[Org14] International Civil Aviation Organization. *Aeronautical Telecommunications. Volume IV - Surveillance and Collision Avoidance Systems*. International Standards and Recommended Practices. Montréal, Quebec, Canada: ICAO, 2014. URL: https://www.spilve.lv/library/law/Annex%2010%20Volume%20IV.pdf.

[PA22] Lucia Pintor and Luigi Atzori. *Analysis of Wi-Fi Probe Requests Towards Information Element Fingerprinting*. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 3857–3862. DOI: 10.1109/GLOBECOM48099.2022.10001618.

[Pan+07] Jeffrey Pang et al. *802.11 User Fingerprinting*. In: *MobiCom'07*. The 13th Annual ACM International Conference. ACM, 2007, p. 99. ISBN: 978-1-59593-681-3. DOI: 10.1145/1287853.1287866. URL: http://portal.acm.org/citation.cfm?doid=1287853.1287866 (visited on 11/18/2021).

[Pöp+11] Christina Pöpper et al. *Investigation of Signal and Message Manipulations on the Wireless Channel*. In: *Computer Security – ESORICS 2011*. Ed. by Vijay Atluri and Claudia Diaz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 40–59.

[Pro20] Proton Team. *VPN Bypass Vulnerability in Apple iOS*. Mar. 25, 2020. URL: https://protonvpn.com/blog/apple-ios-vulnerability-disclosure/ (visited on 10/27/2020).

[PS07] Jeffrey Pang and Srinivasan Seshan. *Tryst: The Case for confidential Service Discovery*. In: *HotNets'07*. 2007.

[PSC09] Christina Pöpper, Mario Strasser, and Srdjan Capkun. *Jamming-resistant Broadcast Communication without Shared Keys*. In: *18th USENIX Security Symposium*. Jan. 2009, pp. 231–248.

[PT20] Tommy Pauly and Darshak Thakore. *Captive Portal API*. RFC 8908. Sept. 2020. DOI: 10.17487/RFC8908. URL: https://www.rfc-editor.org/info/rfc8908.

[RK17] Mohsen Riahi Manesh and Naima Kaabouch. *Analysis of vulnerabilities, attacks, countermeasures and overall risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) system*. In: *International Journal of Critical Infrastructure Protection* 19 (2017), pp. 16–31. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2017.10.002.

[Rob+17] Pieter Robyns et al. *Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices*. In: *Secur. Commun. Networks* 2017 (2017), 6235484:1–6235484:21. URL: https://api.semanticscholar.org/CorpusID:46761199.

[San+23] Muhammad Sangeen et al. *Blind-trust: Raising awareness of the dangers of using unsecured public Wi-Fi networks*. In: *Computer Communications* 209 (2023), pp. 359–367. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2023.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0140366423002396.

[Sca02] John Scardina. *Overview of the FAA ADS-B Link Decision*. July 2, 2002. URL: https://www.icao.int/safety/acp/ACPWGF/ACP-WG-M-5/WGM510.pdf (visited on 07/02/2002).

[Sch+14] Matthias Schäfer et al. *Bringing up OpenSky: A large-scale ADS-B sensor network for research*. In: *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IEEE. 2014, pp. 83–94.

[Ser23] Windows Server. *Tutorial: Deploy Always On VPN - Set up infrastructure for Always On VPN*. 2023. URL: https://learn.microsoft.com/en-us/windows-server/remote/remote-access/tutorial-aovpn-deploy-setup (visited on 06/27/2024).

[SLM13a] Matthias Schäfer, Vincent Lenders, and Ivan Martinovic. *Experimental analysis of attacks on next generation air traffic communication*. In: *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*. ACNS'13. Banff, AB, Canada: Springer-Verlag, 2013, pp. 253–271. ISBN: 9783642389795. DOI: 10.1007/978-3-642-38980-1_16.

[SLM13b] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. *Security of ADS-B: State of the Art and Beyond*. In: *IEEE Communications Surveys & Tutorials* 17 (July 2013).

[SLM15a] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. *Lightweight Location Verification in Air Traffic Surveillance Networks*. In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. Singapore, Republic of Singapore: Association for Computing Machinery, 2015, pp. 49–60. ISBN: 9781450334488.

[SLM15b] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. *On the Security of the Automatic Dependent Surveillance-Broadcast Protocol*. In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), pp. 1066–1087. DOI: 10.1109/COMST.2014.2365951.

[SMW99] Bruce Schneier, Mudge, and David A. Wagner. *Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)*. In: *International Exhibition and Congress on Network Security*. 1999. URL: https://api.semanticscholar.org/CorpusID:247846.

[Som+19] Nissy Sombatruang et al. *Factors influencing users to use unsecured wifi networks: evidence in the wild*. In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '19. Miami, Florida: Association for Computing Machinery, 2019, pp. 203–213. ISBN: 9781450367264. DOI: 10.1145/3317549.3323412. URL: https://doi.org/10.1145/3317549.3323412.

[SRV21] Domien Schepers, Aanjhan Ranganathan, and Mathy Vanhoef. *Let Numbers Tell the Tale: Measuring Security Trends in Wi-Fi Networks and Best Practices*. In: *WiSec '21*. ACM, 2021, pp. 100–105. ISBN: 9781450383493. DOI: 10.1145/3448300.3468286. URL: https://doi.org/10.1145/3448300.3468286.

[sta21a] statcounter. *Mobile & Tablet Android Version Market Share Worldwide - Dec 2021*. 2021. URL: https://gs.statcounter.com/os-version-market-share/android/mobile-tablet/worldwide#monthly-202112-202112-bar (visited on 01/11/2022).

[sta21b] statcounter. *Mobile Operating System Market Share Worldwide*. 2021. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (visited on 01/13/2022).

[sta24a] statcounter. *Android Version Market Share Worldwide - March 2024*. 2024. URL: https://gs.statcounter.com/os-version-market-share/android# (visited on 04/04/2024).

[sta24b] statcounter. *Desktop Windows Version Market Share Worldwide - May 2024*. 2024. URL: https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide (visited on 06/27/2024).

[sta24c] statcounter. *Mobile & Tablet iOS Version Market Share Worldwide - March 2024*. 2024. URL: https://gs.statcounter.com/os-version-market-share/ios/mobile-tablet/worldwide (visited on 04/04/2024).

[sta24d] statcounter. *Mobile Vendor Market Share Worldwide*. 2024. URL: https://gs.statcounter.com/vendor-market-share/mobile (visited on 04/05/2024).

[Ste23] Meghan Stewart. *Extended Security Updates (ESU) program for Windows 10*. https://learn.microsoft.com/en-us/windows/whats-new/extended-security-updates. Accessed: 2024-10-11. May 2023.

[Sun+21] Junzi Sun et al. *OpenSky Report 2021: Insights on ADS-B Mandate and Fleet Deployment in Times of Crisis*. In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. 2021, pp. 1–10. DOI: 10.1109/DASC52595.2021.9594361.

[Sup23] Apple Support. *How to forget a Wi-Fi network on iPhone, iPad, or Mac*. https://support.apple.com/en-us/102480. 2023. (Visited on 10/13/2023).

[TB09] Erik Tews and Martin Beck. *Practical attacks against WEP and WPA*. In: *IACR Cryptol. ePrint Arch.* 2008 (2009), p. 472. URL: https://api.semanticscholar.org/CorpusID:775144.

[TC21] Jiajie Tan and S-H Gary Chan. *Efficient Association of Wi-Fi Probe Requests under MAC Address Randomization*. In: *INFOCOM'21*. IEEE. 2021, pp. 1–10.

[Tip+11] Nils Ole Tippenhauer et al. *On the Requirements for Successful GPS Spoofing Attacks*. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*. 2011, pp. 75–86. DOI: 10.1145/2046707.2046719. URL: https://dl.acm.org/doi/10.1145/2046707.2046719.

[Ura+20] Marco Uras et al. *WiFi Probes sniffing: an Artificial Intelligence based approach for MAC addresses de-randomization*. In: *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. Oct. 2020. DOI: 10.1109/CAMAD50429.2020.9209257.

[Van+16] Mathy Vanhoef et al. *Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms*. In: *Asia CCS '16*. ACM, 2016, pp. 413–424. ISBN: 9781450342339. DOI: 10.1145/2897845.2897883. URL: https://doi.org/10.1145/2897845.2897883.

[VP17] Mathy Vanhoef and Frank Piessens. *Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2*. In: CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1313–1328. ISBN: 9781450349468. DOI: 10.1145/3133956.3134027. URL: https://doi.org/10.1145/3133956.3134027.

[VR20] Mathy Vanhoef and Eyal Ronen. *Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd*. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 517–533. DOI: 10.1109/SP40000.2020.00031.

[WHE14] Kyle D. Wesson, Todd E. Humphreys, and Brian L. Evans. *Can Cryptography Secure Next Generation Air Traffic Surveillance?* In: 2014. URL: https://api.semanticscholar.org/CorpusID:21207906.

[Wik19] Wikipedia contributors. *GPS-Spoofing — Wikipedia, The Free Encyclopedia*. https://de.wikipedia.org/wiki/GPS-Spoofing. 2019. (Visited on 07/03/2019).

[Wik23] Wikipedia contributors. *Pixel 5a — Wikipedia, The Free Encyclopedia*. 2023. URL: https://en.wikipedia.org/wiki/Pixel_5a (visited on 04/05/2024).

[Wik24a] Wikipedia contributors. *Captive Portal — Wikipedia, The Free Encyclopedia*. June 5, 2024. URL: https://en.wikipedia.org/wiki/Captive_portal (visited on 06/18/2024).

[Wik24b] Wikipedia contributors. *iOS 16 — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/IOS_16. 2024. (Visited on 03/30/2024).

[Wik24c] Wikipedia contributors. *iOS Version History — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/wiki/IOS_version_history (visited on 04/05/2024).

[Wik24d] Wikipedia contributors. *Pixel 4a — Wikipedia, The Free Encyclopedia*. 2024. URL: https://en.wikipedia.org/wiki/Pixel_4a (visited on 04/05/2024).

[Wip16] Andrew Wippler. *WiFi Captive Portal*. Accessed: 2024-10-11. 2016. URL: https://andrewwippler.com/2016/03/11/wifi-captive-portal/.

[WK16] Otto Waltari and Jussi Kangasharju. *The Wireless Shark: Identifying WiFi Devices Based on Probe Fingerprints*. In: *Proceedings of the First Workshop on Mobile Data*. MobiData '16. Singapore, Singapore: Association for Computing Machinery, 2016, pp. 1–6. ISBN: 9781450343275. DOI: 10.1145/2935755.2935757. URL: https://doi.org/10.1145/2935755.2935757.

[WK18] Otto Waltari and Jussi Kangasharju. *Quantifying the Information Leak in IEEE 802.11 Network Discovery*. In: *Wired/Wireless Internet Communications*. Ed. by Kaushik Roy Chowdhury et al. Cham: Springer International Publishing, 2018, pp. 207–218. ISBN: 978-3-030-02931-9.

[WMB20] Jack Wilson, David McLuskie, and Ethan Bayne. *Investigation into the security and privacy of iOS VPN applications*. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. ARES '20. Virtual Event, Ireland: Association for Computing Machinery, 2020. ISBN: 9781450388337. DOI: 10.1145/3407023.3407029. URL: https://doi.org/10.1145/3407023.3407029.

[WSG20] Zhijun Wu, Tong Shang, and Anxin Guo. *Security Issues in Automatic Dependent Surveillance - Broadcast (ADS-B): A Survey*. In: *IEEE Access* 8 (2020), pp. 122147–122167. DOI: 10.1109/ACCESS.2020.3007182.

[ZBA24] Juan-Carlos Zúñiga, Carlos J. Bernardos, and Amelia Andersdotter. *Randomized and Changing MAC Address State of Affairs*. Internet-Draft draft-ietf-madinas-mac-address-randomization-15. Work in Progress. Internet Engineering Task Force, July 2024. 19 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-madinas-mac-address-randomization/15/.

[Zee23] Mark Zee. *GPS Spoofing Update: Map, Scenarios and Guidance*. Nov. 8, 2023. URL: https://ops.group/blog/gps-spoofing-update-08nov2023/ (visited on 05/08/2024).

[Zha+19] Fan Zhao et al. *A localization and tracking scheme for target gangs based on big data of Wi-Fi locations*. In: *Cluster Computing* 22.1 (2019), pp. 1679–1690.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Zudem versichere ich, dass dieses gebundene Exemplar der Dissertation und das in elektronischer Form eingereichte Dissertationsexemplar (über den Docata-Upload) und das beim Studienbüro Informatik zur Archivierung eingereichte gedruckte gebundene Exemplar der Dissertationsschrift identisch sind.

Hamburg, im Oktober 2024

Johanna Ansohn McDougall