TECHNISCHE
UNIVERSITÄT
DARMSTADT

AVAILABILITY BY DESIGN: PRACTICAL DENIAL-OF-SERVICE-RESILIENT DISTRIBUTED WIRELESS NETWORKS

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

DISSERTATION

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
von

MILAN STUTE

Erstreferent: Prof. Dr.-Ing. Matthias Hollick
Korreferent: Prof. Guevara Noubir, Ph. D.

Darmstadt 2020
Hochschulkennziffer D17

Milan Stute, *Availability by Design: Practical Denial-of-Service-Resilient Distributed Wireless Networks*, Dissertation, Technische Universität Darmstadt, 2020.

Fachgebiet Sichere Mobile Netze
Fachbereich Informatik
Technische Universität Darmstadt
Jahr der Veröffentlichung: 2020
Tag der mündlichen Prüfung: 14. Februar 2020
URN: urn:nbn:de:tuda-tuprints-114573

Do or do not. There is no try.
—Yoda

ABSTRACT

Distributed wireless networks (DWNs) where devices communicate directly without relying on Internet infrastructure are on the rise, driving new applications and paradigms such as multimedia, authentication, payment, Internet of things (IoT), vehicular communication, and emergency response. However, the increased societal reliance on technology and the resulting "always-on" expectations of the users increase the risk of denial-of-service (DoS) attacks as they can leverage disruption in new ways beyond extortions (ransomware) that are common in today's Internet ecosystem. These new risks extend to our physical world, directly impacting our daily lives, e. g., by being locked out of a smart home or by disrupting vehicular collision avoidance systems. As a research community, we need to protect those new applications that—as we find—can be mapped to a total of three distinct networking scopes: neighbor, island, and archipelago. In this thesis, we advance the field in each of these scopes. First, we analyze two proprietary neighbor communication protocols, Apple Wireless Direct Link (AWDL) and Apple AirDrop, that are deployed on more than 1.4 billion devices worldwide. During the process, we uncover several security and privacy vulnerabilities ranging from design flaws to implementation bugs leading to a machine-in-the-middle (MitM) attack on AirDrop, a DoS attack on AWDL preventing communication, and DoS attacks enabling crashing of neighboring devices. In addition, we found privacy leaks that enable user identification and long-term tracking. All attacks can be mounted using low-cost off-the-shelf hardware. In total, we disclose eight distinct vulnerabilities that we mitigate in collaboration with Apple. Second, we design and implement a new island communication protocol tailored to IoT scenarios, which provides provable protections against previously neglected risks such as wormhole- and replay-supported greyhole attacks. We support our analytical findings with testbed experiments. Third, we propose an archipelago-scope communication framework for emergencies that achieves resiliency against flooding and Sybil attacks. We evaluate our design using an original expert knowledge-based simulation that models human mobility during the aftermath of the 2013 Typhoon Haiyan in the Philippines. Finally, and to nourish future research, we provide a guide for analyzing Apple's wireless ecosystem and publish several software artifacts, including an AWDL Wireshark dissector, open AWDL and AirDrop implementations, a prototype of our IoT communication protocol, and our natural disaster mobility model.

ZUSAMMENFASSUNG

Verteilte drahtlose Netzwerke, in denen Geräte direkt und ohne Verbindung zum Internet kommunizieren, ermöglichen neue Anwendungen und Paradigmen, wie z.B. Multimedia, Internet der Dinge (IoT), Verkehrsvernetzung und Notfallkommunikation. Die zunehmende Abhängigkeit der Gesellschaft von Technologie und die daraus resultierende Erwartung an deren Verfügbarkeit erhöhen jedoch das Risiko von Denial-of-Service-Angriffen (DoS). Diese können Störungen auf neue Weise nutzen, die über Erpressungsmethoden hinausgehen. So können sich DoS-Angriffe direkt auf die physische Welt und das tägliche Leben auswirken, z.B. durch Aussperren aus einem Smart Home oder Stören der Kollisionsvermeidungssysteme in Fahrzeugen. Daher werden Schutzmechanismen für diese neuen Anwendungen benötigt, die auf insgesamt drei Netzwerkbereiche abgebildet werden können: Nachbarschaft, Insel und Archipel. Diese Arbeit enthält Beiträge zu jedem dieser Bereiche. Zunächst werden zwei proprietäre Nachbarschaftsprotokolle, Apple Wireless Direct Link (AWDL) und Apple AirDrop, analysiert, die weltweit auf mehr als 1,4 Milliarden Geräten eingesetzt werden. Dabei werden mehrere Sicherheits- und Privatheitsprobleme aufdeckt, die von Design- bis hin zu Implementierungsfehlern reichen und so die Manipulation von über AirDrop ausgetauschten Daten, die Verhinderung von AWDL-Kommunikation und das Abstürzen benachbarter Geräte ermöglichen. Darüber hinaus erlauben verschiedene Privatheitsprobleme die Identifizierung von Benutzern und Verfolgung von Geräten. Alle Angriffe können mit kostengünstiger Hardware durchgeführt werden. Die insgesamt acht offengelegten Sicherheitslücken wurden in Zusammenarbeit mit Apple geschlossen. Zweitens wird ein neues, auf IoT-Szenarien zugeschnittenes Protokoll in Inseln entworfen und implementiert, das beweisbaren Schutz vor bisher vernachlässigten Risiken wie Wurmloch- und „Replay“-gestützten DoS-Angriffen bietet. Die analytischen Ergebnisse werden durch Testbed-Experimente untermauert. Drittens wird ein Framework für Notfallkommunikation in Archipelen vorgestellt, das resilient gegen Fluten- und Sybil-Angriffe ist. Das Design wird anhand eines neu entwickelten und auf Expertenwissen basierenden Mobilitätsmodell evaluiert, das die Ereignisse nach dem Taifun Haiyan 2013 auf den Philippinen abbildet. Abschließend – und um zukünftige Forschung zu fördern – werden ein Leitfaden zur Analyse des drahtlosen Ökosystems von Apple und mehrere Software-Artefakte veröffentlicht, darunter ein AWDL-Protokolldissektor für Wireshark, quelloffene AWDL- und AirDrop-Implementierungen, ein Prototyp des IoT-Protokolls und das Mobilitätsmodell für Naturkatastrophen.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support of several people. To this end, I first and foremost express my gratitude to my advisor Prof. Matthias Hollick for encouraging me and offering me the opportunity to pursue a Ph. D. I particularly appreciate his support for my family plans and for leaving me a considerable amount of freedom that—among other things—taught me resilience against setbacks. Second, I thank Prof. Guevara Noubir for hosting my research visit in Boston and for accepting my request to act as the second advisor for my thesis.

I thank my colleagues for providing a pleasant and productive working atmosphere, in particular, Flor Àlvarez and Jiska Classen for giving feedback on my thesis, Lars Almon for his hardware support and maintenance of the mesh testbed, and Doris Müller for taking a lot of paperwork off my hands and handling the administrative peculiarities of the university. I thank my students for working with dedication and for taking on often risky projects. Especially, I thank David Kreitschmann for bootstrapping the research on Apple's wireless ecosystem. Further, I thank Susan Roden and Anne Salajka for thoroughly proofreading my thesis.

I thank my friends for taking off my mind of this thesis, be it cooking or climbing sessions. I thank my parents for their never-ending love and support. I thank my son Samuel for challenging me, teaching me patience, and—most importantly—continuously making me smile. Finally, I thank my wife Dina for always supporting me, for traveling the world with me from near and far, and generally for her unconventional ideas and out-of-the-box thinking that make life so much more exciting and enjoyable.

My research has been partially funded by the LOEWE initiative within the NICER[1] project and the German Federal Ministry of Education and Research (BMBF) and the State of Hesse within the CRISP and ATHENE[2] research centers. Also, I thank the Font Awesome[3] team for providing free high-quality vector graphics that allowed me to create a unified look for Figures 1, 4, 22, 27 and 46 in this thesis.

1 https://www.nicer.tu-darmstadt.de
2 https://www.athene-center.de
3 https://fontawesome.com

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PROGRAMS

ACRONYMS

ACK	acknowledgment
AF	action frame
AP	access point
ARP	Address Resolution Protocol
AW	availability window
AWDL	Apple Wireless Direct Link
BLE	Bluetooth Low Energy
BSSID	basic service set identifier
CRL	certificate revocation list

DCF	distributed coordination function
DD	direct delivery
DNS	Domain Name System
DNS-SD	DNS service discovery
DoS	denial-of-service
DRO	disaster response organization
DRT	disaster response team
DTN	disruption-tolerant networking
DWN	distributed wireless network
EAW	extended availability window
EU	European Union
EW	extension window
FCS	frame check sequence
FIFO	first-in first-out
GO	group owner
IBSS	independent basic service set
IDR	international disaster response
IoT	Internet of things
MAC	medium access control
MANET	mobile ad hoc network
mDNS	multicast DNS
MIF	master indication frame
MitM	machine-in-the-middle
MR	message rank
MSG	message
NAN	Neighbor Awareness Networking
NDP	Neighbor Discovery Protocol
NFC	near-field communication
OS	operating system
OSOCC	on-site operations coordination center
OUI	organizational unique identifier

PDR	packet delivery rate
PKI	public-key infrastructure
PKT	packet
POI	point of interest
PS	priority set
PSF	periodic synchronization frame
RDC	reception/departure center
RSSI	received signal strength indication
RTT	round-trip time
RWP	random waypoint
SEB	source-based elastic buckets
SRO	search and rescue operation
SSHWS	Saffir–Simpson hurricane wind scale
TLV	type-length-value
TTL	time-to-live
TU	time unit
UN	United Nations
USRT	urban search and rescue team
UUID	universally unique identifier

LIST OF PUBLICATIONS

During the course of writing this thesis, I co-authored several papers and articles that I list below.

JOURNAL AND MAGAZINE ARTICLES

[1] Milan Stute, David Kreitschmann, and Matthias Hollick. "Reverse Engineering and Evaluating the Apple Wireless Direct Link Protocol." In: *ACM GetMobile* 23.1 (Mar. 2019). **Part of this thesis**. DOI: 10.1145/3351422.3351432 (cit. on p. xxiii).

CONFERENCE AND WORKSHOP PAPERS

[2] Lars Baumgärtner, Stefan Kohlbrecher, Juliane Euler, Tobias Ritter, Milan Stute, Christian Meurisch, Max Muhlhäuser, Matthias Hollick, Oskar von Stryk, and Bernd Freisleben. "Emergency Communication in Challenged Environments via Unmanned Ground and Aerial Vehicles." In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Oct. 2017. DOI: 10.1109/GHTC.2017.8239244.

[3] Florian Kohnhäuser, Milan Stute, Lars Baumgärtner, Lars Almon, Stefan Katzenbeisser, Matthias Hollick, and Bernd Freisleben. "SEDCOS: A Secure Device-to-Device Communication System for Disaster Scenarios." In: *IEEE Conference on Local Computer Networks (LCN)*. **Part of this thesis**. Extended in [16]. Oct. 2017. DOI: 10.1109/LCN.2017.47 (cit. on pp. xxi, xxiv).

[4] Jeremy Lakeman, Matthew Lloyd, Romana Challans, Angus Wallace, Paul Gardner-Stephen, Milan Schmittner, and Matthias Hollick. "A Practical and Secure Social Media Facility for Internet-Deprived Populations." In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Oct. 2017. DOI: 10.1109/GHTC.2017.8239236.

[5] Christian Meurisch, Julien Gedeon, Artur Gogel, The An Binh Nguyen, Fabian Kaup, Florian Kohnhäuser, Lars Baumgärtner, Milan Schmittner, and Max Muehlhäuser. "Temporal Coverage Analysis of Router-Based Cloudlets Using Human Mobility Patterns." In: *IEEE Global Communications Conference (GLOBECOM)*. Dec. 2017. DOI: 10.1109/GLOCOM.2017.8255035.

[6] Milan Schmittner, Arash Asadi, and Matthias Hollick. "SEMUD: Secure Multi-hop Device-to-Device Communication for 5G Public Safety Networks." In: *IFIP Networking Conference*

and Workshops. **Part of this thesis**. Extended in [15]. June 2017. DOI: 10.23919/IFIPNetworking.2017.8264846 (cit. on pp. xxi, xxiv).

[7] Milan Schmittner and Matthias Hollick. "Xcastor: Secure and Scalable Group Communication in Ad hoc Networks." In: *IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2016. DOI: 10.1109/WoWMoM.2016.7523512.

[8] Daniel Steinmetzer, Milan Stute, and Matthias Hollick. "TPy: A Lightweight Framework for Agile Distributed Network Experiments." In: *International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH)*. ACM, Nov. 2018. DOI: 10.1145/3267204.3267214.

[9] Milan Stute, David Kreitschmann, and Matthias Hollick. "One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol." In: *ACM Conference on Mobile Computing and Networking (MobiCom)*. Best Community Paper Award. **Part of this thesis**. Oct. 2018. DOI: 10.1145/3241539.3241566 (cit. on p. xxiii).

[10] Milan Stute, Max Maass, Tom Schons, and Matthias Hollick. "Reverse Engineering Human Mobility in Large-scale Natural Disasters." In: *ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. **Part of this thesis**. Nov. 2017. DOI: 10.1145/3127540.3127542 (cit. on p. xxiii).

[11] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. "A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link." In: *USENIX Security Symposium*. **Part of this thesis**. Aug. 2019. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/stute (retrieved Dec. 9, 2019) (cit. on pp. xxiii, xxiv).

POSTERS AND DEMONSTRATORS

[12] Christian Meurisch, The An Binh Nguyen, Julien Gedeon, Florian Konhäuser, Milan Schmittner, Stefan Niemczyk, Stefan Wullkotte, and Max Mühlhauser. "Upgrading Wireless Home Routers as Emergency Cloudlet and Secure DTN Communication Bridge." In: *IEEE International Conference on Computer Communication and Networks (ICCCN)*. July 2017. DOI: 10.1109/ICCCN.2017.8038485.

[13] Milan Schmittner. "DoS-resistant Buffer Management for Delay/Disruption-Tolerant Networks." In: *Cybersecurity and Privacy Summer School (CySeP)*. June 2017. URL: https://people.kth.se/~papadim/cysep/poster.html (retrieved Dec. 9, 2019).

[14] Milan Stute, David Kreitschmann, and Matthias Hollick. "Demo: Linux Goes Apple Picking: Cross-Platform Ad hoc Communication with Apple Wireless Direct Link." In: *ACM Conference on Mobile Computing and Networking (MobiCom)*. Best Demo Award. **Part of this thesis**. Oct. 2018. DOI: 10.1145/3241539.3267716 (cit. on p. xxiii).

UNDER PEER REVIEW

[15] Milan Stute, Pranay Agarwal, Abhinav Kumar, Arash Asadi, and Matthias Hollick. "LIDOR: A Lightweight DoS-Resilient Communication Protocol for Safety-Critical IoT Systems." In: *IEEE Internet of Things Journal (IoT-J)* (submitted). **Part of this thesis**. Extended from [6] (cit. on pp. xx, xxiii, xxiv).

[16] Milan Stute, Florian Kohnhäuser, Lars Baumgärtner, Lars Almon, Stefan Katzenbeisser, Matthias Hollick, and Bernd Freisleben. "RESCUE: A Resilient and Secure Device-to-Device Communication Framework for Emergencies." In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* (submitted). **Part of this thesis**. Extended from [3] (cit. on pp. xix, xxiii, xxiv).

COLLABORATIONS AND MY CONTRIBUTION

Exploring and solving complex research problems is a process most effectively tackled as a team. Ideas evolve by discussing them with others. Crossing topic boundaries is accelerated by involving field experts. The results presented in this thesis have benefited from the same synergies.[4] I am happy to have worked with many bright colleagues and dedicated students—thank you Lars Almon, Pranay Argawal, Arash Asadi, Lars Baumgärtner, Bernd Freisleben, Alexander Heinrich, Matthias Hollick,[5] Stefan Katzenbeißer, Florian Kohnhäuser, David Kreitschmann, Abhinav Kumar, Max Maass, Alex Mariotto, Sashank Narain, Guevara Noubir, and Tom Schons.

When research is collaborative, it becomes hard to break down contributions and assign credit to an individual author. For all papers that form this thesis, each author was involved in discussing ideas, debating results, and finalizing the presentation in varying degrees. In the following, I detail the contributions of my co-authors and myself per chapter. In addition, I follow the regulations of the Department of Computer Science at Technische Universität Darmstadt and give an account of the parts that include verbatim or revised fragments of previous publications that form this thesis as indicated in the preceding list of publications.[6]

Chapters 1 and 2 collate the contributions, background, and related work sections of the core papers that form this thesis [15, 16, 9, 11].

Chapter 3 is based on joint work with Tom Schons and Max Maass [10]. Tom created the presented disaster mobility model during his master thesis under the co-supervision of Max and myself. In particular, Tom conducted the interviews and implemented the mobility model and scenarios (Appendix C.5). I ran the evaluation.

Chapter 4 distills my personal experience from reverse engineering and supervising student reverse engineering projects. While the chapter has not been published in this current form, Sections 4.2 to 4.4 are revisions of previous publications [9, 11].

Chapter 5 is based on three joint publications with David Kreitschmann [14, 9, 1]. David conducted a large part of the reverse engineering work on AWDL during his master thesis under my supervision. I augmented his analysis with more details and conducted the experimental evaluation. David wrote the first version of the Wireshark dissector (Appendix C.1) which I refactored, extended with

4 For that reason, I will use the pronoun "we" in the presentation of the technical parts of this thesis.

5 My advisor, Matthias Hollick, is a co-author on all of my papers for discussing, guiding, and providing feedback for each project.

6 References in this chapter refer to my list of publications given on pages xix to xxi.

additional fields, and prepared for publication. I am the sole author of the AWDL implementation (Appendix C.2).

Chapter 6 describes and discusses Apple's AirDrop protocol and is based on [11, Section 3]. Insights about the protocol were gathered by Alexander Heinrich during a lab project under my supervision. I completed the analysis, including details such as the requirements for authenticated connections (Section 6.3). Alexander also wrote the first version of OpenDrop (Appendix C.3), which I refactored, extended, and prepared for publication.

Chapter 7 contains a security analysis of AWDL and AirDrop and is based on joint work with my students David, Alexander, and Alex Mariotto as well as Guevara Noubir and Sashank Narain of Northeastern University, Boston [11]. David mentioned some attack ideas such as desynchronization and tracking via broadcast medium access control (MAC) addresses in his master thesis. Alex provided an initial feasibility study of the desynchronization attack (Section 7.1) in his master thesis. Guevara guided the project and provided ideas such as conducting a supporting user survey that would strengthen the final results. Sashank was responsible for designing and carrying out the survey and summarizing related work on user tracking in Appendix A. I conducted all analyses, implemented all proofs of concept (PoCs), and conducted all experimental evaluations.

Chapter 8 is based on joint work with Arash Asadi as well as Abhinav Kumar and Pranay Argawal of IIT Hyderabad, which is currently under review [15]. Arash and I published an initial version of the protocol [6]. I was responsible for designing, implementing (Appendix C.4), and evaluating the protocol. Arash provided the application scenario. We involved Abhinav and Pranay for an extension of the paper. They contributed the overhead and convergence analysis (Sections 8.3 and 8.4). In addition, I conducted testbed experiments (Section 8.6) that replace the simulation-based evaluation of the original paper.

Chapter 9 is based on joint work with Florian Kohnhäuser, Lars Baumgärtner, Lars Almon, Stefan Katzenbeißer, and Bernd Freisleben, which is awaiting review feedback [16]. We jointly discussed the features of the framework, designed the base protocol (Section 9.2), and published an initial version [3]. Florian provided the design for the mobile certification authorities (Section 9.3). I designed the buffer management with its Sybil-secure extension (Sections 9.4 and 9.5) as well as implemented the protocol and conducted the evaluation (Section 9.6).

Part I

PRELUDE

1 INTRODUCTION

Distributed wireless network (DWN) services are becoming more prevalent and are being increasingly deployed using infrastructure-less architectures. Big tech companies such as Apple and Google are moving from cloud-based solutions to distributed services using wireless technologies for commercial applications, including crowd-sourced location tracking [73], file sharing [10, 72], and proximity-based authentication procedures [9]. The reasons for this shift can be partially attributed to the Snowden revelations [74], which have increased societal privacy consciousness and, thus, rendered "off-the-grid" solutions more attractive. Other factors also contribute to this paradigm shift: in light of 5G and the Internet of things (IoT) [33], networks have to deal with tens of billions of devices.[1] To enable scaling and to reduce the load in their core networks, operators want to move traffic towards the networks' edges. Also, some applications such as vehicular communication require extremely low latencies and, thus, simply cannot use today's infrastructure-based communication such as cellular networks. Finally, DWNs can substitute for infrastructure-based communication in the aftermath of natural disasters such as floods, hurricanes, or earthquakes, which are expected to occur more frequently around the globe due to climate change [135]. During disasters, panic reactions and physical damage often lead to inoperable local communication infrastructures [99], impairing disaster response operations. As a solution, ubiquitous mobile devices such as smartphones can form backup networks in which devices forward messages for one another [123, 128].

Distributed wireless networks (DWNs) enable novel commercial and humanitarian applications and help to scale to billions of IoT devices.

1.1 MOTIVATION

The increased general reliance of society on technology and the resulting "always-on" expectations of the users nourish denial-of-service (DoS) attacks as they use disruption for leverage [49]. The paradigm shift from cloud-based to DWN applications requires reevaluating existing and detecting emerging security threats to design systems for availability. In particular, the pervasiveness of these novel applications and their integration in our physical world enable new attack vectors and render attacks on availability even more severe.

Societal reliance on technology make disruptions through DoS attacks a real threat.

Today, DoS attacks are the most prevalent form of attacks against Internet services [187]. Such attacks exploit vulnerable Internet pro-

1 Predictions vary strongly and range from 20.4 billion by the year 2020 [67], over 22.3 billion by 2024 [56], to 41.6 billion by 2025 [98].

tocols and services such as NTP to launch amplification attacks [44]. Mitigations for DoS attacks are typically based on detection and filtering and require respective infrastructure [200]. While DoS attacks on Internet services typically result in inconvenience and financial losses for the target, results can be much more devastating in DWN applications. Due to the pervasiveness of these applications, their nonfunctioning has a direct impact on human lives, which might make them more attractive targets in the future. Consider a smart home: a single malicious device could prevent somebody from opening a gate or door by merely jamming the wireless unlock procedure. Such DoS attacks may enable new kinds of ransomware that require payment for users to re-enter their house or apartment. In a vehicular setting, collision avoidance systems, where vehicles broadcast their presence and current speed, require multihop communication [194]. If messages were not forwarded, human lives were put at risk. Finally, if an attacker has the resources to cause a large-scale blackout of the cellular communication infrastructure, they might also have the capabilities to disrupt an emergency backup network.

In DWN applications, DoS attacks not only profoundly impact our everyday life but may endanger human lives.

Due to existing as well as new threats coming from DoS on various wireless applications, we believe that a systematic look at the communication protocols involved is crucial.

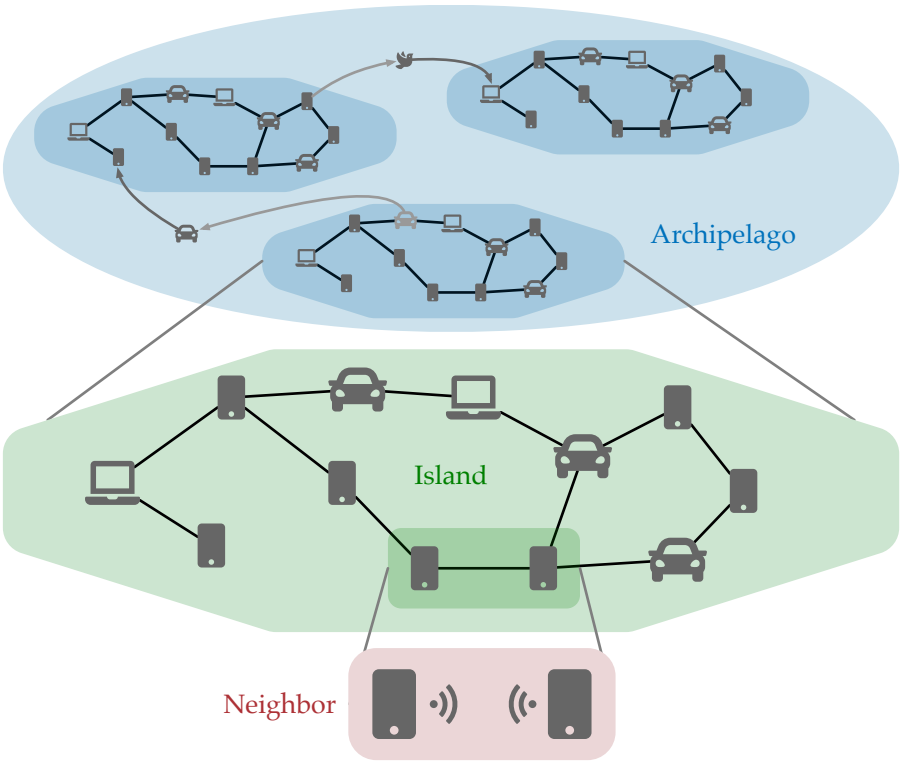
1.2 CHALLENGES AND GOALS

In this thesis, we address DoS attacks in different applications by considering and working with the generic network model shown in Figure 1. The model consists of several *nodes* that can be any kind of wirelessly connected devices such as smartphones, vehicles, or other IoT appliances. Further, the model defines three communication *scopes*. The first scope extends to the *neighbors* of a node, i. e., communication with other nodes that are within the physical communication range with one another. The second scope encompasses an *island*, i. e., a group of nodes indirectly connected via multiple hops. Routing protocols enable communication within islands. Finally, the third scope adds space-time paths that are created when nodes move from one island to another. Such an *archipelago* is considered in disruption-tolerant networking (DTN) settings and deploys store-carry-forward schemes. Note that with each scope, communication range but also delay increases and, therefore, render the scope usable or unusable for a specific application.

Our network model comprises three scopes: neighbor, island, and archipelago.

This model can be used to describe a wide range of networks and applications. For example, Apple AirDrop [10] allows sharing files between two directly connected devices and, as such, only uses the neighbor scope. Typical IoT applications use mesh networking to interconnect devices [31] and, thus, additionally, use the island scope. Smartphone-based backup networks use the neighbor scope

Smartphone-based emergency communication is the running example in this thesis.



(Chapter 3)
Chapter 9

Chapter 8

(Chapter 4)
Chapters 5–7

Figure 1: Networking model comprising three scopes used in this thesis. We list the chapters that are (partially) concerned with the particular scope.

for device-to-device communication, can make use of island communication within groups of nodes, and use the archipelago scope to extend communication range to a metropolitan area or a district. As emergency communication uses all three scopes, we adopt this use case as a running example for this thesis.

The ultimate goal of this thesis is providing *practical and comprehensive protections against DoS attacks for DWN protocols* for all networks that fall into this model. We approach this problem by (1) analyzing and improving existing protocols and (2) designing new protocols. Securing each scope poses its unique challenges. We discuss them in the following.

1.2.1 *Neighbor Communication*

Experimenting with DWNs require an accessible neighbor communication technology.

Neighbor communication provides the basic building block for multihop protocols on the island and archipelago scopes. To conduct practical experiments on modern mobile devices, e. g., for smartphone-based emergency networks, we need a high-throughput, accessible, and reliable technology. Many existing technologies enable direct wireless communication between two devices. However, most of them are impractical or cannot be used on modern mobile devices. And we

found that those that work well are proprietary and have not been openly scrutinized for security.

PRACTICAL NEIGHBOR COMMUNICATION Standardized wireless neighbor communication technologies such as Wi-Fi ad hoc, Wi-Fi Peer-to-Peer, or Bluetooth Low Energy (BLE), so far, have been unsuccessful in providing a practical high-throughput communication link between neighboring mobile devices. The problems of these technologies include low energy efficiency, low reliability, and low throughput. Around 2014, Apple introduced Apple Wireless Direct Link (AWDL), a proprietary undocumented Wi-Fi-based ad hoc protocol that today drives numerous services within Apple's ecosystem, such as AirDrop. While we found that these services worked surprisingly well in terms of reliability and performance during an initial assessment, the protocol is currently only available on Apple's platforms covering over 1.4 billion devices.

Wi-Fi ad hoc, Peer-to-Peer, and BLE have distinct shortcomings.

UNKNOWN DENIAL-OF-SERVICE PROTECTION SCOPE The main problem with analyzing proprietary protocols such as AWDL is that they are typically closed-source. In such cases, assessing the DoS protection scope is non-trivial and requires several steps. In particular, we (1) need a methodology and tools to open such a closed ecosystem, (2) understand the protocols at hand, and finally (3) conduct a security analysis.

Analyzing closed-source proprietary protocols is a key challenge addressed in this thesis.

1.2.2 *Island Communication*

In the past two decades, the mobile ad hoc network (MANET) and mesh network communities have intensively investigated island communication. While there have been numerous routing protocol proposals for these types of networks, only a few of them are currently deployed in practice (e. g., OLSR [43]). And none of them offer decent protection against DoS attacks. Many research projects have attempted to retro-fit security on existing protocols or designed new protocols from scratch. We have found that these proposals often consider a weak attacker model or lack analytical proof of their security properties, which means that they miss protections against certain well-known DoS attacks. Also, we have found that those proposals typically have only been evaluated in a simulator. The lack of a practical implementation might also be one of the reasons why these protocols have never been adopted.

MANET routing has been studied extensively in the past years, but no practical and secure protocol has been proposed.

LIMITED DENIAL-OF-SERVICE PROTECTION SCOPE Many early secure MANET routing protocols share a fundamental flaw: while they have security measures to protect the route discovery (i. e., control plane), they are susceptible to dropping attacks in the data plane.

Dropping can be especially devastating if an attacker launches a blackhole attack, where they attract traffic towards themselves (e. g., by mounting a wormhole attack) only to drop it afterward. This flaw has been acknowledged by the research community so that more recent and sophisticated proposals integrate control and data plane to prevent dropping attacks even on the data plane. Most of them rely on an (end-to-end) feedback mechanism to ensure that messages arrive at the destination. However, there are many different types of such dropping attacks, some of which have remained unaddressed.

Protocols that integrate both control and data plane security offer the best protection against DoS attacks.

PRACTICAL EVALUATION ON REAL HARDWARE Related works proposing new secure integrated routing protocols typically rely on simulation-driven experiments to assess their performance. While simulations allow evaluating complex scenarios with different types of mobility, simulators typically do not capture the impact of processing overhead on the nodes, which might be non-negligible for secure protocols that employ cryptographic operations. Therefore, to assert the practicality and applicability of such proposals, we believe that it is essential to evaluate those protocols on real hardware.

Simulation is the prevalent form of evaluation for MANET protocols.

1.2.3 *Archipelago Communication*

Communication in the archipelago scope leverages DTN techniques. Due to its cooperative, distributed, and resource-constrained nature, DTN communication is particularly susceptible to DoS attacks [195]. Also, the performance of DTN communication and, in extension, that of any security mechanism is highly dependent on node mobility.

LIMITED DENIAL-OF-SERVICE PROTECTION SCOPE Existing DTN security solutions focus on single attacks such as flooding [120, 152] or route manipulation [118, 119]. Furthermore, Sybil attacks, where an adversary operates under multiple identities, have not been addressed in previous DTN research. To summarize, while there exist solutions to some problems, so far, there is no approach comprehensively mitigating different DoS attacks.

Some DoS attacks have not been addressed in the archipelago scope.

PRACTICAL EVALUATION WITH REALISTIC MOBILITY The determining factor of the performance for inter-island communication is node mobility because moving nodes create space-time paths between islands. This can be simply explained: if none of the nodes moves while the network is partitioned into islands, then communication between islands can never occur. Also, if there are fast nodes that move frequently and fast between islands, then communication delay is low. Communication delay is a crucial factor for the usability of a network, e. g., days-old emergency messages might be useless. Therefore, the evaluation of any system that attempts to solve a particular purpose

As mobility greatly affects DTN performance, we need realistic mobility models for evaluation.

needs to use a realistic mobility model that reflects the use case. For the emergency use case, there currently exist no realistic mobility models for disasters on the scale of archipelago communications.

1.3 CONTRIBUTIONS

We analyze existing and design new protocols.

To tackle these challenges and achieve our goals, we make the following main contributions: (1) we dissect the AWDL ad hoc protocol, a state-of-the-art high-performance neighbor communication protocol, (2) we conduct a security and privacy analysis on AWDL and the related Apple AirDrop protocol, (3) we design a DoS-resilient protocol for island-scope communication, and (4) we design a DoS-resilient framework for archipelago-scope communication.

1.3.1 *Dissection of Apple's Wireless Ecosystem*

We study AWDL as it solves many problems of existing neighbor communication protocols.

AWDL is designed as a successor to the less popular Wi-Fi ad hoc mode and Wi-Fi Peer-to-Peer [42] and solves many practical issues such as energy consumption and concurrent communication with an infrastructure Wi-Fi network. As such, it is currently the only practical[2] high-throughput wireless neighbor communication protocol. We conduct a comprehensive investigation on AWDL through binary and runtime analysis and present its frame format and operation. In short, AWDL is based on the IEEE 802.11 standard and makes use of vendor-specific extensions that allow custom protocol implementations. Each AWDL node periodically emits custom action frames containing a sequence of availability windows (AWs) indicating its readiness to communicate with other AWDL nodes. An elected master node synchronizes these sequences. Within these AWs, nodes can communicate with their neighbors using a dedicated data frame format. Outside the AWs, nodes can tune their Wi-Fi radio to a different channel to communicate with an access point, or turn it off to save energy. We summarize our main contributions:

AWDL implements a channel hopping mechanism as a Wi-Fi overlay.

We release a Wireshark dissector and an open AWDL implementation.

- We provide insights into the macOS operating system (OS) and its Wi-Fi driver architecture and debugging facilities to help future reverse engineering endeavors (Chapter 4).
- We present the AWDL frame format and operation in detail (Sections 5.1 and 5.2).

2 Neighbor Awareness Networking (NAN) is the unofficial standardized successor of AWDL but is currently only available in a few Android phones. We detected NAN-related code in the Wi-Fi driver of macOS 10.15 beta (which was removed in the release version). Therefore, we assume that Apple will also adopt NAN in one of its next major OS releases, possibly using an AWDL co-existence mode for backward compatibility.

- We conduct an experimental analysis of AWDL to assess election behavior, synchronization accuracy, throughput, and channel hopping strategies (Section 5.4).
- We publish an AWDL protocol dissector for Wireshark (Appendix C.1).
- We re-implement AWDL in C for Linux-based systems and make the code available as open-source software (Appendix C.2).

1.3.2 *Security and Privacy Analysis of Apple Wireless Direct Link and AirDrop*

We conduct a comprehensive security and privacy analysis of AWDL.

We discover a protocol-level DoS attack, implementation bugs, and new MitM attack on AirDrop.

We conduct the first security analysis of AWDL and its integration with BLE, starting with the reverse engineering of protocols and code supported by analyzing patents. Our analysis reveals several security and privacy vulnerabilities ranging from design flaws to implementation bugs enabling different kinds of attacks. We present a machine-in-the-middle (MitM) attack enabling stealthy modification of files transmitted via AirDrop, a DoS attack preventing communication between devices, privacy leaks allowing user identification and long-term tracking undermining medium access control (MAC) address randomization, and targeted DoS and blackout DoS attacks (i. e., enabling simultaneous crashing of all neighboring devices). The flaws span AirDrop's BLE discovery mechanism, AWDL synchronization, UI design, and Wi-Fi driver implementation. We demonstrate that the attacks can be stealthy, low-cost, and launched by devices not connected to the target Wi-Fi network. We implement proofs of concept (PoCs) and demonstrate that the attacks can be mounted using a low-cost (20 US$) micro:bit device and an off-the-shelf Wi-Fi card. Specifically, these are our contributions:

We reverse engineer AirDrop and implement an open version of the protocol that we make publicly available.

- We reverse engineer the AirDrop protocol and its integration with AWDL and the BLE stack (Chapter 6).
- We re-implement AirDrop in Python and make the code available as open-source software (Appendix C.3).
- We discover security and privacy vulnerabilities in AWDL and AirDrop and present four novel network-based attacks on iOS and macOS. These attacks include (1) a DoS attack on AWDL effectively preventing communication (Section 7.1), (2) a DoS–supported MitM attack which intercepts and modifies files transmitted via AirDrop (Section 7.2), (3) two blackout DoS attacks on Apple's AWDL implementations which allow crashing Apple devices in proximity (Section 7.3), and (4) a long-term device tracking attack that may reveal personal information such as the name of the device owner (Appendix A).
- We propose practical mitigations for all attacks and share them with Apple. We list all vulnerabilities and fixes in Appendix B.

1.3.3 *Secure Island Communication Protocol*

We propose *LIDOR*, a lightweight DoS-resilient multihop protocol that secures island communication. While LIDOR provides authenticated and (optionally) confidential communication, more importantly, it uses an end-to-end feedback mechanism to quickly detect and locally repair broken paths, thus, comprehensively mitigating different variants of DoS attacks. LIDOR's path selection provably converges even in the presence of hard-to-detect wormhole-supported greyhole attacks. By leveraging symmetric-key cryptographic primitives, we ensure efficient operation of LIDOR even on embedded devices leading to low end-to-end delivery delays. We validate our proposal by running testbed experiments. The following are our main contributions:

LIDOR provably converges under a wormhole-supported greyhole attack.

- We present LIDOR, the first multihop communication protocol that comprehensively protects against all well-known variants of blackhole and greyhole attacks (Section 8.2).
- We analytically prove that LIDOR's communication overhead is lower than Castor [64], which is—as we found—the most secure and comprehensive multihop solution in the state-of-the-art (Section 8.3).
- We provide analytical proof that LIDOR converges even in the presence of a wormhole-supported greyhole attack (Section 8.4).
- We implement a LIDOR prototype in C++ using lightweight symmetric cryptographic primitives and make it available as open-source software (Appendix C.4).
- We conduct experiments in our testbed to validate our analytical findings. In particular, we show that LIDOR does not incur additional overhead under attack and significantly increases delivery rates under attack compared to the Castor protocol (Section 8.6).

We implement a LIDOR prototype and conduct testbed experiments.

1.3.4 *Secure Archipelago Communication Framework*

We present *RESCUE*, a resilient and secure device-to-device communication framework for emergency scenarios, which provides comprehensive attack protection. RESCUE's basic communication protocol relies on epidemic routing, authenticated and immutable messages, and an effective acknowledgment processing. This way, common attacks, such as message or routing manipulation, blackholing, or impersonation, are already prevented. Yet, as in today's Internet infrastructure [204], the key challenge is to defend against DoS attacks originating from *individuals* as well as *multiple identities* (Sybil attack) that flood the network. For this purpose, RESCUE pursues a twofold mitigation technique. First, certificates are used to cryptographically bind users to network identifiers, which hinders the adversary from assuming multiple identities. Since traditional static certificate infrastructures

RESCUE features a minimalistic communication protocol to thwart several well-known DoS attacks.

may be unavailable in the disaster area—and more generally—in disaster scenarios, we propose a novel distributed approach that enables new users to obtain certificates from mobile authorities also during a disaster. Second, RESCUE applies a novel buffer management scheme that substantially increases message delivery rates, i. e., availability, in the presence of flooding attacks, both from individuals and multiple identities. As our solution relies on node-local decisions rather than a (complex) distributed protocol, it provides a *minimal surface to attacks* and causes *no network overhead* by design. In addition, instead of identifying and excluding misbehaving users from the network, our scheme provides a fair allocation of available resources to all users. Hence, RESCUE does not suffer from false positives, where a valid user is mistakenly excluded from the emergency communication system. In particular, we make the following contributions:

We use mobile certificate authorities to bootstrap nodes post-disaster.

- We conceive a mobile distributed certificate infrastructure tailored to disaster scenarios that hinders an adversary from assuming multiple identities to perform Sybil attacks (Section 9.3).
- We design a fair buffer management scheme that mitigates the effect of flooding attacks by individuals (Section 9.4).
- We extend our buffer management scheme with priority sets that increase resiliency against Sybil attacks, guarantee at least the performance of direct message delivery, and support unregistered users without a certificate (Section 9.5).
- We design and implement a realistic disaster mobility model based on recent disasters, which we make available as open-source software (Chapter 3 and Appendix C.5).
- We evaluate RESCUE using large-scale network simulations with our realistic disaster model, demonstrating that RESCUE maintains excellent delivery rates even under attack (Section 9.6).

Using a novel buffer management scheme in combination with priority sets, RESCUE is resilient to flooding and Sybil attacks.

1.4 OUTLINE

The rest of this thesis is structured as follows. Chapter 2 provides definitions, gives background information on wireless technologies and DoS attacks, and discusses related work. We introduce the emergency scenario, including an original mobility model in Chapter 3. In Part ii, we explain Apple's wireless ecosystem from a system's perspective and provide insights into our reverse engineering process in Chapter 4. Chapters 5 and 6 explain AWDL and AirDrop in detail. In Chapter 7, we conduct a security analysis of these protocols. We design novel protocols in Part iii. Chapter 8 presents LIDOR, a lightweight, secure multihop protocol for IoT, and Chapter 9 introduces RESCUE, a secure DTN-based protocol for emergency communication. Finally, we conclude this thesis in Chapter 10 of Part iv.

We structure this thesis by scopes of the networking model.

2

BACKGROUND AND RELATED WORK

This chapter introduces terminology, explains fundamental denial-of-service (DoS) attacks, compares existing wireless neighbor communication technologies, and, finally, discusses related work on secure wireless multihop networks that together form distributed wireless networks (DWNs).

2.1 DEFINITIONS

This section differentiates the closely related terms *security*, *availability*, *resiliency*, and *dependability* in the context of wireless networks to set the scope for this thesis.

2.1.1 *Security*

We consider *security* as a network's property [27] or "system condition that results from the establishment and maintenance of measures to protect the system" [164]. In particular, security is a measure to impede or thwart malicious attacks by employing countermeasures. Security comprises the classic CIA triad with the three goals: confidentiality, integrity, and availability. While we consider confidentiality (preventing disclosure of data to unauthorized entities [101]) and integrity (preventing data modification by unauthorized entities [101]) throughout this thesis, our primary focus lies on availability.

In this thesis, we consider confidentiality and integrity, but focus on availability.

2.1.2 *Availability*

Availability means that network services are usable by an entity upon request [164]. In the context of this thesis, we explicitly consider *survivable* networks, i. e., those that are available on demand even under attack [62]. Availability can be measured by a metric that indicates the probability that a system is "in a state to perform a required function at a given instant of time or at any instant of time within a given time interval" [176]. In the context of networks and communications, availability is commonly measured as the packet delivery rate (PDR) or, conversely, as the packet loss.

We consider networks that are usable even when under attack.

2.1.3 *Resiliency*

A resilient system successfully accommodates unforeseen events.

In the broad sense, *resiliency* describes the ability of a system to "anticipate and adapt to the potential for surprise and failure" [85]. More technically and applied to the networking context, "anticipating and adapting" means possibly providing degraded service when under attack, but recovering autonomously, quickly, and fully from it. In the same context, "surprises" can be translated into new and unexpected attacks [114]. We leverage resiliency as a design concept when proposing new protocols in Chapters 8 and 9 to reduce the attack surface and to absorb and mitigate DoS attacks.

2.1.4 *Dependability*

The definition of dependability is extremely broad and exceeds the scope of this thesis.

Dependability is a complex system's property that encompasses many aspects, such as reliability, availability, and security. The traditional definition is "the ability to deliver service that can justifiably be trusted" [18]. However, more recent works have put resiliency in the dependability context as well [114]. While dependability captures many desirable aspects, its definition is too broad for the scope of this thesis, where we are concerned with security (in particular availability) and resiliency.

2.2 DENIAL-OF-SERVICE ATTACKS

We consider two types of DoS attacks: starving resources and impairing protocol operation.

In this work, we consider DoS attacks that target the availability of a communication network [103, 195]. As individual nodes are also part of the network, attacks that consume a node's resources are within scope. Generally, DoS attacks either try to starve resources directly (jamming and flooding) or impair network operation on the protocol level (dropping, spoofing, replaying, and blackholing). Some attacks listed below (spoofing, pseudospoofing, replaying, and wormholing) can also be used outside the DoS context. However, they can all be used to achieve or amplify the impact of DoS. In the following, we explain each attack, its goal, and existing mitigations.

2.2.1 *Dropping*

The arguably simplest form of network attacks is to drop packets.

Dropping is a simple DoS attack but can have a substantial impact on the performance and availability of a network. In essence, if a malicious router decides not to forward packets, and there are no alternative paths between some source and destination, that router can completely break communication. Dropping is most effective when combined with an attack that attracts traffic towards the attacker, thereby creating a *blackhole* [103]. Typical mitigations to dropping involve end-to-end feedback mechanisms that allow nodes to detect

whether a packet transmission was successful or not. If a transmission fails, the sender can try a different path. As a proactive measure, multi-path forwarding can be employed to send each packet via multiple paths. In that case, an attacker would need to compromise all used paths at the same time. However, adding redundancy comes at the cost of increased bandwidth usage.

2.2.2 *Flooding*

During flooding attacks [38], an adversary sends an excessive number of packets into the network. Their goal is to exhaust network resources. In particular, a flooding attack can attempt (1) to saturate communication links and paths, (2) to fill up node-local packet queues and buffers, or (3) to combine both methods. As a result, legitimate traffic might be dropped or might experience a low quality of service. Defending against flooding attacks is not trivial. If packets are not authenticated, source blacklisting is not an option because packets cannot be attributed to a particular node. But even if packets are authenticated, setting good policies such as rate limits [153] is a difficult task. The limits must be neither too strict (impairs legitimate communication) nor too soft (gives attacker too much leverage) as network dynamics and requirements might change over time [41].

By flooding packets, an attacker can starve buffer capacity or network bandwidth.

2.2.3 *Jamming*

A jammer tries to impede wireless communications by sending out strong radio signals that interfere with valid transmissions and prevent the receiver from successfully decoding a frame [25]. Jamming can target an entire frequency band to block all communication or can target individual frames selectively. The latter is called a reactive jammer and decides whether or not to jam based on the first few bits of a frame [71, 121, 162]. Jamming attacks can only affect a network locally, i. e., at the neighbor scope, as the attacker is constrained by transmission power. However, this might be sufficient to cause network-wide disruptions, e. g., by partitioning a network or forcing packets over an adversary-chosen path [143].

Jamming can locally prevent any wireless communication.

2.2.4 *Replaying*

An attacker may attempt to repeat a past transaction again by sending previously-recorded packets [1]. Whether such a replay attack can successfully disrupt network operation depends on the protocol. For example, replaying route responses might re-instantiate a stale route. Cryptographic end-to-end protocols employ nonces to protect against replay attacks. Nonces are unique values that are only used once and are used as an input to the calculation of cryptographic primitives

An attacker can inject old state if messages are not properly protected with nonces.

such as ciphers. Good choices for nonces are sequence numbers or values drawn at random from a large enough space to avoid reuse. Within the network, nodes can try to keep track of seen nonces and drop duplicate packets.

2.2.5 *Spoofing*

Sending messages outside the protocol definition is called spoofing.

We define spoofing as sending messages that do not conform to a particular protocol definition, e. g., for an Address Resolution Protocol (ARP) spoofing attack [156]. Spoofing can mean that an attacker sets disallowed values in specific header fields, e. g., setting a high time-to-live (TTL) value to increase the lifetime of its messages. Also, an attacker might impersonate another node by using a spoofed source address. Typically, spoofing can be prevented by using cryptographic primitives to protect message integrity, such as digital signatures, hash-based integrity checks, or more advanced authenticated encryption schemes [102]. While this works end-to-end, intermediate nodes are unable to verify the integrity and authenticity of a packet as they do not possess the required end-to-end key material. Another problem is mutable header fields that cannot have end-to-end protection and give an attacker potential leverage, e. g., by modifying hop counts.

2.2.6 *Pseudospoofing (Sybil Attack)*

A Sybil attacker operates under many identities to evade detection.

An attacker can use a large number of spoofed identities to win consensus schemes employed in different security protocols, use them to avoid detection, or circumvent blacklisting mechanisms [47]. Such pseudospoofing is also called a *Sybil* attack [54], which we will use throughout this thesis. Sybil attacks are especially problematic in settings where users are unable to authenticate each other, e. g., in peer-to-peer networks. Generally, systems that employ a public-key infrastructure (PKI) are immune to such attacks because it will be hard for the attacker to acquire multiple valid identities as long as they are unable to compromise a certificate authority (CA).

2.2.7 *Wormholing*

A fast out-of-band channel is called a wormhole.

The attacker uses a fast out-of-band channel between distant parts of the network (the "wormhole") to make valid nodes believe that they are neighbors [89]. Its effect varies depending on the target protocol. For example, wormholing can break neighbor discovery protocols. In routing protocols that rely on performance-based distance metrics such as hop count or round-trip time, wormholes can deliver route discovery messages faster and, therefore, are likely to win in route selection. This way, the attacker attracts traffic towards themselves, which allows them to perform traffic analysis or mount a dropping at-

tack. We consider *rushing* [91] and *tunneling* attacks as weaker variants of wormholing, so we do not explicitly consider them in this thesis. Mitigations for wormholes include *packet leashes* [90].

2.2.8 *Blackholing*

Blackhole attacks attract and subsequently drop packets.

In analogy to the space object, a blackhole attack tries to attract and drop network traffic [84, 92]. Blackholing targets routing services and combines routing manipulation attacks on the control plane (e. g., via spoofing or replaying) to attract traffic with packet dropping to disrupt communication on the data plane [83]. Consequently, there exist several flavors of blackhole attacks that depend on the supporting control plane attack. A more stealthy variant of a blackhole is called greyhole and uses selective packet dropping to evade detection. Mitigations for blackhole and greyhole attacks include the same measures that can mitigate routing manipulation and dropping attacks.

2.3 WIRELESS NEIGHBOR COMMUNICATION TECHNOLOGIES

Several Wi-Fi-based neighbor communication technologies have evolved over time.

Wireless neighbor communication technologies are the basic building block that enables DWNs, in particular, multihop networks, as they create the link between adjacent nodes. Over the past decades, technologies based on the IEEE 802.11 ("Wi-Fi") standard [95] evolved because the "limitations of IBSS mode [. . .] led the Wi-Fi Alliance[1] to define Wi-Fi Direct. Further, due to concerns regarding Wi-Fi Direct, Apple Wireless Direct Link (AWDL) was developed by Apple and eventually adopted by the Wi-Fi Alliance as the basis for Neighbor Awareness Networking (NAN)" [42]. In the following, we discuss these technologies as well as Bluetooth Low Energy and provide an overview in Table 1.

2.3.1 *Wi-Fi Ad Hoc*

For practical reasons, Wi-Fi ad hoc never became widely adopted.

The IEEE 802.11 independent basic service set (IBSS) mode, commonly known as "Wi-Fi ad hoc," creates a wireless network without special controller roles. An IBSS is created by sending beacon frames with an SSID and basic service set identifier (BSSID) on a particular channel. Other nodes joining the network will send out beacons themselves using the same information. The mode is robust to nodes leaving the network as all nodes broadcast beacons. The nodes do not require any further synchronization. However, IBSS has never become widely

1 The Wi-Fi Alliance is a vendor association that holds the Wi-Fi trademark for IEEE 802.11-based technologies and certifies products using the specification. Although the Wi-Fi Alliance does not formally create the standard, their certification has relevance in the market. The alliance also creates own standards based on IEEE 802.11, such as Wi-Fi Direct and Wi-Fi Aware.

TECHNOLOGY	COMMENT
Wi-Fi ad hoc [95]	never became widely deployed, mostly due to a lack of efficient power-saving mechanisms that are crucial for mobile devices.
Wi-Fi P2P [60]	assigns the role of an access point to one node, which creates a single point of failure and drains that node's battery quickly.
Apple Wireless Direct Link [174]	is a proprietary protocol and achieves "concurrent" neighbor connections and a connection with an AP by employing channel hopping as an overlay to IEEE 802.11.
Neighbor Awareness Networking [61]	has a proper specification with a design that is based on AWDL but is currently only available on selected Android devices.
Bluetooth LE [32]	is energy efficient but features low data rates compared to Wi-Fi-based technologies.

Table 1: Overview of wireless neighbor communication technologies.

adopted, mostly due to a lack of power-saving mechanisms, which are crucial for mobile devices [39]. IBSS is not supported on Android. Microsoft announced it might not be available in future versions of Windows [133]. On Apple's operating systems (OSs), encryption is not supported, and iOS only allows joining existing IBSS networks but not to create new ones.

2.3.2 *Wi-Fi Peer-to-Peer*

When the owner leaves, the remaining nodes need to re-establish a Wi-Fi P2P group.

Wi-Fi Peer-to-Peer (P2P) [60], also known under its certification name Wi-Fi Direct, connects multiple devices directly without an infrastructure network. During operation, one node assumes the role of a group owner (GO) and essentially acts as an access point (AP). It is not possible to migrate the role of the GO to another device: if the GO leaves the network, a new network must be created. Wi-Fi P2P connections are established by listening on one channel and sending probe requests on all channels, which delays the connection process in practice. Experiments show that establishing a connection takes from four to more than ten seconds [39]. Discovering devices thus drains the battery of mobile devices very fast.

2.3.3 *Apple Wireless Direct Link*

Apple Wireless Direct Link (AWDL) is a proprietary IEEE 802.11 extension which is deployed in Apple products [173, 174]. At its core, AWDL uses a channel hopping mechanism to enable "simultaneous" communication with an AP and other AWDL nodes on different chan-

nels. This channel hopping is implemented as a sequence of so-called availability windows (AWs). For each AW, a node indicates if it is available for direct communication and, if so, on which channel it will be. To allow nodes to meet and exchange data on the same channel, they need to align their sequences in the time domain. AWDL nodes elect a common master and use its time reference to achieve synchronization. In Chapter 5, we reverse engineer the protocol, derive a specification, and evaluate its performance.

AWDL is a proprietary protocol by Apple that implements a channel hopping overlay.

2.3.4 *Neighbor Awareness Networking*

Neighbor Awareness Networking (NAN) [61], also known as Wi-Fi Aware, extends IEEE 802.11 with proximity service discovery. NAN is designed to be energy efficient, allowing continuous operation on battery-powered devices [40]. NAN depends on beacon frames sent from an elected master. These synchronize the timing of all devices in an area. During a short discovery window the master sets, devices can turn their radio on, exchange service and connection information (e. g., parameters for Wi-Fi P2P), and turn their radio off again. In fact, we found that NAN employs similar concepts as AWDL, but strongly differs in specification and implementation. While standardization makes NAN a great candidate for cross-platform neighbor communication technology, there are still only a few devices that support it. Android 8 introduced an NAN API [5] which requires an NAN-enabled Wi-Fi firmware from the device vendors [5] (e. g., Google Pixel 2), which limits use on the Android platform. Further, we found that mobile Apple devices do not support NAN as of iOS 13.

NAN is the standardized successor to AWDL but is not yet widely available.

2.3.5 *Bluetooth Low Energy*

Bluetooth [32] is a separate standard with different PHY and medium access control (MAC) layers. Bluetooth operates in the same 2.4 GHz band as Wi-Fi and is often integrated into the Wi-Fi chip to share the same antennas. Bluetooth Low Energy (BLE) is incompatible with classic Bluetooth and is optimized for low energy consumption and, therefore, offers limited bandwidth. The usable maximum BLE 4.2 data rate is 394 kbit/s [53]. It is commonly implemented in small battery-powered devices such as smartwatches and fitness trackers, but it is not designed for large data transfers. Also, BLE provides the basis for Bluetooth Mesh [2, 31], a flooding-based multihop protocol designed for Internet of things (IoT) devices.

Bluetooth is suitable for low-bandwidth applications.

PROTOCOL	BLACKHOLE W/ SPOOFING	BLACKHOLE W/ SYBIL	BLACKHOLE W/ REPLAYING	BLACKHOLE W/ WORMHOLE	GREYHOLE W/ SPOOFING	GREYHOLE W/ SYBIL	GREYHOLE W/ REPLAYING	GREYHOLE W/ WORMHOLE	JAMMING	FLOODING
BTFR [193]	✓		✓		✓		✓			
2ACK [122]	✓	—		—	—	—	—	—		
Sprout [57]	✓	✓*		—	✓	✓*		—	✓*	
ODSBR [19]	✓	—		—	✓	—		—	✓*	
Castor [64]	✓	✓	—	✓	✓	✓	—	—	✓*	✓
SEMUD [160]	✓	✓	✓	✓	✓	✓	✓	—	✓*	—
LIDOR [171]	✓	✓	✓	✓	✓	✓	✓	✓	✓*	—

Table 2: Resiliency of integrated multihop protocols to different DoS attacks. We differentiate between resilient (✓), limited resilient (✓*), not resilient (—), and unknown (blank).

2.4 SECURITY IN WIRELESS MULTIHOP NETWORKS

Wireless multihop networks present the basis for island and archipelago communication in DWNs. Therefore, we discuss related work on mitigating DoS attacks in wireless multihop networks. In particular, we focus on mobile ad hoc networks (MANETs) for island communication and disruption-tolerant networking (DTN) for archipelago communication.

2.4.1 *Security in Mobile Ad Hoc Networks*

Integrated routing approaches offer the best security as they protect both the control and data plane.

Initial works on MANET routing protocols [88, 92, 144, 158, 199] only secure the control plane of the network layer and, therefore, cannot comprehensively protect against (selective) dropping attacks on the data plane. Similarly, protocols only protecting the data plane [21, 145], can only detect packet loss but not react on it. Therefore, the best approach against blackhole and greyhole attacks is an integrated approach that protects both the control and data plane. We provide a summary of such integrated approaches in Table 2. As we are interested in a comprehensive DoS resiliency, we briefly point out the strengths and drawbacks of each approach in the following. We base our comparison on the adversary model presented in Section 8.1.1.

2ACK [122] selectively acknowledges data packets and is thus vulnerable to all types of greyhole attacks. In addition, the protocol is vulnerable to colluding attackers. ODSBR [19] uses authentic end-to-end acknowledgments for data packets and resorts to path probing to identify broken links. The latter makes the protocol vulnerable to Sybil and wormhole attacks, where a large number of fictitious links

are created, and all have to be explicitly identified. Sprout [57] uses path probing to evaluate the quality of entire paths instead of links. Since the protocol relies on source routing, the source needs to be able to identify all other nodes. In addition, Sprout was shown to perform worse than Castor under the wormhole attack. BTFR [193] is similar to Sprout in design featuring source routing and end-to-end acknowledgments. Castor [64] has an elegant design to use end-to-end acknowledgments and achieves higher resiliency against sophisticated attacks such as blackholes and wormholes by incorporating an implicit and independent route discovery. SEMUD [160] is our work based on Castor, which adds protection against replay attacks and reduces Castor's overhead. Finally, LIDOR [171] extends SEMUD and introduces provably convergent path selection, effectively making the protocol resilient to wormhole-supported greyhole attacks. We present LIDOR in Chapter 8.

Our LIDOR protocol shares fundamental design aspects with Castor to provide strong protection against DoS attacks.

None of the above protocols propose new anti-jamming techniques, yet, some mention them as a requirement [19]. However, even without explicit protection against jamming, most secure routing protocols will implicitly avoid locally jammed areas as they are considered unreliable [57, 64, 160, 171]. Of all approaches, Castor [64] is the only one that proposes a rate-limiting mechanism to thwart flooding attacks. Since SEMUD [160] and LIDOR [171] are both based on Castor, adopting a similar mechanism would be straightforward, but is not discussed in this thesis.

2.4.2 *Security in Disruption-Tolerant Networks*

Since our communication framework can withstand a variety of attacks (see Table 12), it supports and complements several existing secure opportunistic communication systems [35, 87, 189]. We review related work on DTN attacks as well as possible countermeasures and compare them with our solution presented in Chapter 9. In addition, Table 3 shows attacks addressed in related works.

FLOODING ATTACK MITIGATION Flooding DoS attacks on unauthenticated DTNs were discussed in the literature, but contrary to previous findings [38], we show in Chapter 9 that authentication is essential for reliable operation. In authenticated networks, one proposal [120] enforces rate limits that are hard-coded in certificates, using a distributed protocol. Nodes exceeding their rate limit are blacklisted and excluded from the network. Our initial work, SEDCOS [107], implements flooding protection implicitly via a fair buffer management strategy. In Chapter 9, we extend this scheme as part of RESCUE.

Contrary to the findings in previous work, we observe that flooding attacks have a severe impact on unauthenticated DTNs.

SYBIL ATTACK MITIGATION Previous works try to identify Sybil identities and then take appropriate actions to exclude them from

WORK	FLOODING	REPLAYING	SPOOFING	SYBIL	DROPPING	WORMHOLING	BLACKHOLING
Encounter-Based Routing [141]	—	—	—	—	✓	—	✓
Encounter Tickets [117]	—	—	—	—	✓	—	✓
Claim-Carry-and-Check [120]	✓	✓*	✓*	—	—	—	—
Trust-Based Spreading [177]	✓	✓*	✓*	✓	—	—	—
SEDCOS [107]	✓	✓*	✓	—	✓	—	✓
RESCUE [171]	✓	✓*	✓	✓	✓	—	✓

Table 3: DTN attacks addressed in previous works. An attack is addressed (✓), implicitly addressed (✓*), or not addressed (—).

Previous works attempt to detect Sybil identities.

the network. One approach exploits the users' social networks [197, 198]. However, this requires communication between peers, which is feasible for online peer-to-peer systems but not for DTN scenarios. In [155], Sybil identities are detected at direct neighbor nodes. This approach is suitable for proximity services, but not for DTNs, where Sybil nodes might be multiple hops away. In [177], nodes bootstrap trust relationships randomly and then collaboratively filter bogus messages. In contrast to the above works, our proposal in Chapter 9 does not identify attackers but accepts their presence and uses a design that absorbs their impact.

Signed encounter tickets prevent route manipulation attacks at the cost of increased overhead.

SECURE DTN ROUTING Encounter-based routing in DTNs is used to intelligently select forwarding nodes based on their contact history, which works well, assuming recurring mobility patterns. At the same time, it makes the network susceptible to blackhole attacks, where an attacker lies about past encounters to appear as a reliable forwarder. Previous works have proposed to use signed encounter tickets that are exchanged upon contact [117, 141]. Unfortunately, exchanging and verifying these tickets introduces communication and computational overhead. In Chapter 9, we use epidemic routing to thwart all routing attacks and mitigate the problems of increased message replicas using effective buffer management.

3 EMERGENCY SCENARIO

Emergency communication is an application scenario for distributed wireless networks (DWNs) that we use as a running example in this thesis. During severe natural disasters, communication infrastructure is often destroyed or overloaded. To restore communication locally, smartphones and other ubiquitous mobile devices can leverage their built-in wireless neighbor communication technologies to form backup networks. Networking performance is highly dependent on node, and therefore, user mobility. To allow for realistic evaluations of new communication protocols, we have designed a realistic mobility model based on expert knowledge [175] that we summarize in this chapter. In the following, we give a brief introduction to natural disasters and response and present our mobility model.[1] Readers already familiar with the scenario may directly skip to the core contributions of this thesis on neighbor, island, and archipelago communication in Parts ii and iii.

Disaster response efforts can be supported by smartphone-based backup networks.

3.1 NATURAL DISASTERS AND RESPONSE

Around the globe, we observe a continuous increase in natural disaster occurrences [115]. When a disaster strikes, the communication infrastructure is often destroyed or unavailable in the immediate aftermath, which hinders effective disaster relief work [70, 99]. From several recent natural disasters (Table 4), we chose to re-create human mobility of the 2013 Typhoon Haiyan as media coverage and response was high, and communication infrastructure was dysfunctional during the first days. Based on the information gathered for this and other specific disasters, we extract recurring behavioral patterns of the various entities involved in disaster relief work. To this end, we define *roles* and role-specific *activities*. The information presented in this section was gathered from 126 IDR experts within 71 disaster response organizations (DROs) who provided reports and guidelines. Also, we conducted on-site and remote interviews with 15 of those experts.

We interview IDR experts to derive a mobility model.

3.1.1 *2013 Typhoon Haiyan*

Typhoon Haiyan lasted from November 3 to 11, 2013, and was one of the strongest tropical cyclones ever recorded [81]. Even though Ty-

1 The information provided in this chapter is highly condensed. For more details about our methodology and the model itself, we refer to the respective publication [175].

DISASTER	YEAR	FATALITIES	AREA (KM^2)
Hurricane Maria	2017	64	10 063
Nepal earthquake	2015	9 000	3 610
Cyclone Pam	2015	24	12 190
Ludian earthquake	2014	617	1 487
Typhoon Haiyan	**2013**	**6 300**	**71 503**
Christchurch earthquake	2011	185	1 426
East Africa drought	2011	260 000	2 346 466
Tropical storm Washi	2011	1 292	104 530
Tohoku earthquake	2011	15 894	83 955
Haiti earthquake	2010	316 000	27 750

Table 4: Selected natural disasters since 2010, including fatality count and affected area [75, 136]. We focus on Typhoon Haiyan in this chapter.

phoon Haiyan had devastating effects on large portions of Southeast Asia, for this work, we will focus on the aftermath of November 8 when Typhoon Haiyan hit the Philippines at 04:40 local time. Haiyan was the deadliest and most damaging Philippine typhoon on record. It left more than one million houses partially or completely damaged, killing at least 6 300 people and leaving many injured and homeless [139]. Typhoon Haiyan was ranked as a category five typhoon, the highest category by the definition of the Saffir–Simpson hurricane wind scale (SSHWS), implying that "catastrophic damage will occur" and "most of the area will be uninhabitable for weeks or months." After the storm had passed, widespread damage became visible with power lines cut off, roads blocked by fallen debris, and trees and buildings collapsed under the strong winds [97]. A 2013 preliminary estimate [139] calculated the total damage related to typhoon Haiyan to be around 2.86 billion US$.

Typhoon Haiyan caused catastrophic damage and destroyed parts of the local power and communication infrastructure.

Right after the disaster had hit the Philippines, many officials concluded that even though early warnings had been issued to the population of potentially affected areas, only a few people evacuated. The low evacuation rate was likely related to the high number of smaller typhoons the Philippines experience every year, which led to the population underestimating the severity of the coming typhoon. Warnings were broadcast on TV and radio but went largely unheeded. The typhoon was accompanied by the biggest storm surge ever experienced within the Philippines, reportedly reaching between four to six meters in height [167]. As a result, the typhoon brought fast-rising tides and surge water, which led to many additional fatalities [113].

Only a fraction of the affected population evacuated in time.

Using an extended version of this report [175] together with additional information gathered from operational reports and expert interviews, we derive a human mobility model for natural disasters.

3.2 HUMAN MOBILITY MODEL

Generic mobility models such as random waypoint (RWP) have been used to study the performance of mobile wireless networks such as disruption-tolerant networkings (DTNs). However, these models do not capture the non-randomness of human mobility (as presented before) and, therefore, produce questionable results when trying to understand network performance in realistic scenarios. Based on expert knowledge, from which we derived roles, activities, and scenario specifics, we implement a mobility model and make it freely available (Appendix C.5). Based on simulation results, we present the characteristics of this model.

We need a realistic mobility model to assess the performance of DWNs in the emergency context.

3.2.1 *Roles*

We identified the main stakeholders in natural disaster-struck areas and defined the following seven roles with distinct behavioral patterns: (1) the healthy local population, (2) the injured local population, (3) disaster response teams (DRTs) from DROs, (4) dedicated urban search and rescue teams (USRTs), (5) scientists (e. g., storm chasers and typhoon experts), (6) United Nations (UN) officials, and (7) government officials.

We differentiate between local population and IDR-related staff.

3.2.2 *Activities*

To create a model, we further need to define activities that regularly occur in disaster areas attributable to specific roles. In the following, we provide a detailed yet non-exhaustive description of activities that we identified during the interviews with IDR experts. We implement all of those activities in our model.

EVERYONE In the evening, everyone goes to their respective base camp, home, or shelter to sleep. Those arriving via the airport (e. g., DROs), on the day of arrival, first go to the reception/departure center (RDC) for registration, then visit the on-site operations coordination center (OSOCC), and finally set up the base camp or sleeping place.

All roles follow a day-night cycle.

DROS AND DRTS After arriving and registering at the airport, they go to the OSOCC or the town hall for a situation briefing and then start to help the affected population with one of the following activities: (1) collect dead bodies and organize burials; (2) patrol the main streets of the city and clean streets from debris, such that supplies can be

Some roles have different activities during the day.

delivered; or (3) go to food and water distribution centers to serve the locals until the end of the day. Besides that, they regularly visit the UN hotel, the OSOCC site, the base camp, or town hall to meet with officials and other DROs.

Officials meet and coordinate at central locations.

UN AND GOVERNMENT OFFICIALS Officials regularly (at least daily) visit the OSOCC, the town hall or base camp, for a situation briefing and meet other officials and DROs. During the day, they perform reconnaissance missions to get a situation overview, such as infrastructural damage. This information is used to provide help. Also, they organize the disaster relief efforts with other officials and DROs, such as setting up food and water distribution spots and organizing burials.

Scientists might leave the site once they completed their job.

SCIENTISTS Within the first two to three days, they collect scientific evidence from the disaster site before the cleaning of the rubble and debris starts. Upon completing their job, they either depart via the airport or volunteer to help the DROs.

Search-and-rescue missions systematically move through an area.

USRT After arriving and registering at the airport, they go to the OSOCC for a situation briefing and search and rescue operation (SRO) planning. When starting an SRO, USRTs go to the chosen location in the morning and then search every house in that street. When done, they repeat with the next street in the direct neighborhood. Usually, SROs are stopped after one week as the chances to find survivors diminish, and USRTs fly back home.

Locals mostly occupy themselves with helping in their neighborhood or move to a food and water distribution point.

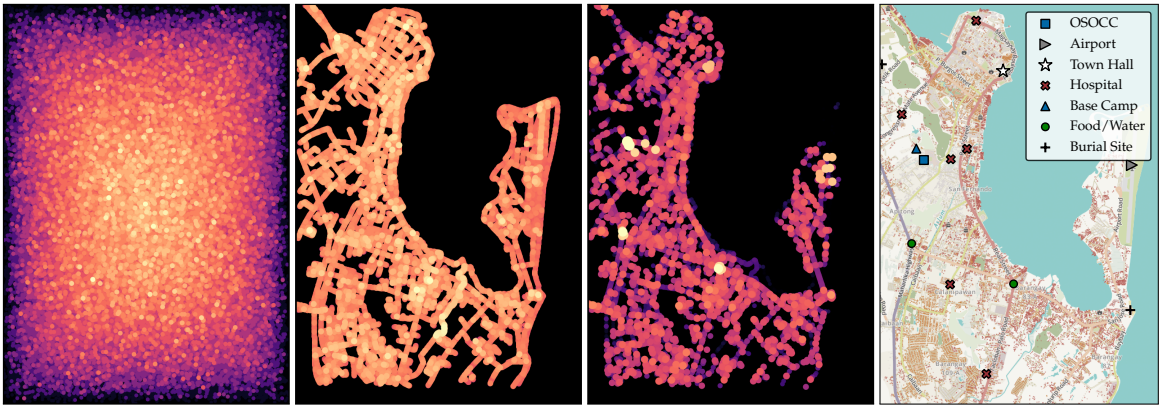
HEALTHY LOCAL POPULATION According to eyewitnesses, most locals stay at home or try to find friends and family members within the immediate surroundings after the disaster has struck. Later on the first day, they stay in the proximity of their homes to assess the damage and to help their neighbors. Then, they start to look for food and water, for example, at distribution centers where they will return daily. The rest of the day, they either volunteer for cleaning operations (we model this by slowly roaming around the city), or as replacement of security personnel to patrol the area.

The injured either stay at home or a hospital.

INJURED LOCAL POPULATION The injured either head to the closest hospital or stay at home if they are unable to move. Upon arrival, they stay at the hospital if the hospital's capacity is not exhausted and, otherwise, leave to find another one.

3.2.3 *Characteristics*

We want to visually validate our natural disaster (*ND*) and compare it with two other widely-used models: the random waypoint (*RWP*)



(a) RWP (b) Map-based RWP (c) Natural Disaster (d) Reference map

Figure 2: Spatial node distribution in different mobility models averaged over seven days. We sample the node counts from a grid of 10 × 10 m squares. Circle sizes and colors (dark to bright) scale logarithmically with node count to highlight hot spots.

mobility model [29] and a map-based RWP model (*Map*) [16] where waypoint selection is still random, but node movement is confined to a street grid. For the following evaluation, we rely on the ONE simulator v1.6.0 [104]. We average the experimental results over ten independently seeded runs. The model's source code and our experiment data set are available online (Appendix C.5)

We implement a mobility model based on this expert knowledge in a simulator.

We are interested in the spatial node distribution and encounters that occur during a disaster since they both affect the applicability of a DTN. Node hot spots can function as communication hubs where messages are quickly exchanged, while nodes that have many encounters can act as "data mules" and transport messages over larger distances.

We want to characterize the mobility patterns of our model.

SPATIAL NODE DISTRIBUTION We visualize the spatial node distribution of the three mobility models using a scatter-plot heatmap in Figure 2. In Figure 2a, we identify the typical non-uniform center-weighted distribution [29] of the *RWP* model. From a practical perspective, this means that nodes are moving across inaccessible areas, for example, a bay. In contrast, *Map* and *ND* (Figures 2b and 2c) essentially "redraw" the underlying street grid. Here, the nodes' movements are confined to streets and paths and are thus no longer moving across water. However, node distribution in *Map* across streets appears generally uniform. There are only minor hot spots at street intersections, which we expect since movement trajectories cross there. In general, nodes are located with similar probabilities at any point in the map. Figure 2c shows that *ND* exhibits characteristic hot spots that are mappable to certain point of interests (POIs) in the street map (Figure 2d), where many nodes stay for a longer period. Most prevalent are the locations of the OSOCC and the base camp as DRTs and officials frequent them. Also, we identify other hot spots at the city hall and the food and water distribution points. Unfortunately, the ONE does not support

The RWP mobility model yields unrealistic node distributions as expected.

Our model has frequently visited POIs.

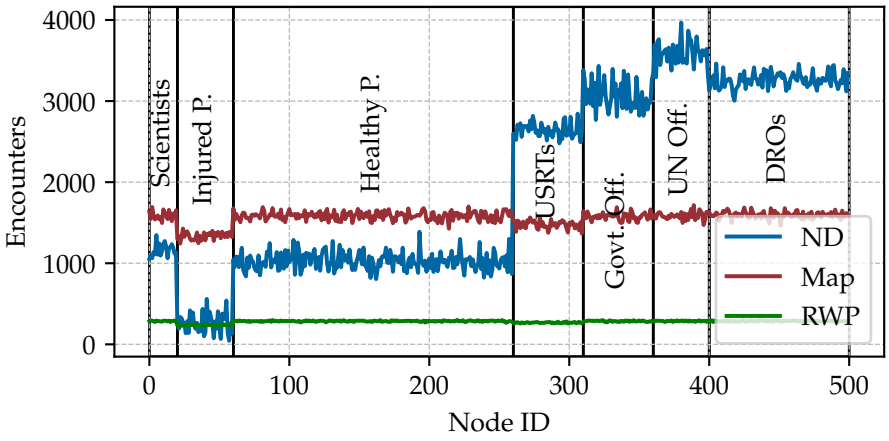


Figure 3: Number of encounters per node over one week.

removing nodes from a running simulation. As a workaround, inactive nodes, i. e., those that have not yet arrived and those that have already left, stay at the airport. The hot spots around the airport can, therefore, be considered an artifact.

An encounter is an event where two nodes move into transmission range of one another.

ENCOUNTERS An *encounter* is a transmission opportunity that occurs if two nodes move in each other's transmission range. DTNs performance highly depends on the number of encounters a node makes while moving around. For example: if a node encounters the destination of any currently carried message, it can directly deliver it. The advantage of direct delivery is that it prevents the replication overhead to intermediate nodes in the form of radio transmissions and storage consumption. So, in a scenario where communicating parties are generally physically close to one another, or at least meet regularly, a DTN deployment could exclusively rely on direct deliveries.

The encounter frequency is highly dependent on the nodes' roles.

Therefore, assessing the encounter characteristics of the underlying mobility model is essential to understand which protocols are suitable for a natural disaster scenario. In Figure 3, we observe that in *ND*, the local population groups (healthy and injured) make significantly fewer contacts than the other groups. Especially the injured encounter very few other nodes. On the other hand, DRO teams and government and UN officials make significantly more contacts due to regular meetings at the OSOCC, in the town hall, and the base camps. In the *RWP* and *Map* models, the number of encounters solely depends on the average velocity of the user role. For example, injured as well as heavily equipped USRTs move slower than the other groups. The low node density can explain the generally low number of encounters of *RWP* in combination with the low transmission range: as the nodes freely move around the large area, nodes only infrequently move into each other's transmission range.

Node in RWP experience generally few encounters as their movement is not confined to the street grid.

Part II

NEIGHBOR COMMUNICATION

4

A HACKER'S GUIDE TO APPLE'S WIRELESS ECOSYSTEM

The goal of this chapter is to provide a structured way to conduct reverse engineering[1] of Apple wireless protocols while using practical examples from our analysis of Apple Wireless Direct Link (AWDL) and AirDrop. First, we show useful vantage points and present services in Apple's wireless ecosystem. We explain the binary analysis methodology and share our insights on dynamic analysis. Then, we explain how to access security key material of Apple services and, finally, discuss the applicability of our methodology to other protocols in Apple's ecosystem. All services that we analyzed in this thesis are available on both macOS and iOS. Since we found macOS to be much more open and accessible than iOS, we used macOS as the platform that we analyzed.

We provide actionable advice on how to approach reverse engineering.

4.1 VANTAGE POINTS

We approach protocol analysis from different *vantage points* that we depict in Figure 4. Static (1) *binary* analysis is extremely hard to conduct as each protocol is implemented across multiple components (frameworks and daemons). Therefore, during the initial stages, it is useful to monitor the (2) *system* as a whole to identify key components

Multiple vantage points allow us to change perspective.

1 A *hacker* is a curious individual who wants to understand the technical details of a (potentially proprietary and closed-source) system to achieve interoperability or conduct a security analysis.

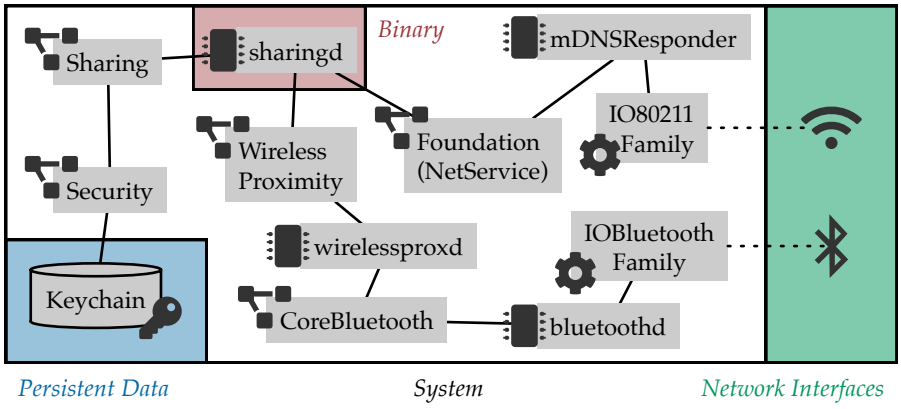


Figure 4: Vantage points that we used during our analysis. We provide a simplified view of components and their interactions such as daemons (), frameworks (), and drivers () that are used by AWDL and AirDrop.

that can thoroughly be examined subsequently. Also, data transmitted via (3) *network interfaces* is easily accessible using monitoring tools and is tremendously useful for dynamic analysis. We found that the ability to retrieve and use (4) *persistent data*, especially from the system's keychain, is essential for building prototypes and, thus, validating findings. Finally, any available (5) *documentation* (not shown in Figure 4) such as patents [184, 185] or Apple's iOS security white paper [12] can be helpful for an initial assessment and understanding some design elements of the service. Having those multiple vantage points at hand enables us to harness more information, to change perspective if we get stuck (e. g., when encountering encrypted traffic), and to resume analysis at a later point (e. g., after extracting the decryption keys). We elaborate on the four vantage points in Figure 4 in the following.

4.2 BINARY ANALYSIS

We analyzed numerous binaries related to AWDL to find those parts that implement the protocol finally. We first illustrate our selection process and then discuss the two-part Wi-Fi driver, which implements most of the AWDL protocol stack. We focus our analysis on macOS and assume that the architecture is, in principle, similar to that of iOS as the two operating systems (OSs) share a large common codebase [11].

4.2.1 *Binary Landscape*

Understanding and navigating the binary landscape of macOS is important to find and relate components of interest.

FRAMEWORKS AND DAEMONS Apple excessively uses *frameworks* and *daemons* in its OSs. Consequently, numerous dependencies result in a complex binary selection process.

Frameworks offer an API to their corresponding singleton daemons and can be used by other daemons and processes. Daemons and their respective frameworks typically have a similar name (e. g., *sharingd* and *Sharing*) or share a derived prefix (e. g., *searchpartyd*, *SPFinder*, and *SPOwner*). We list the locations in the file system in the following. */System/Library/Frameworks* contains frameworks with public documentation[2] such as Security. */System/Library/PrivateFrameworks* contains other frameworks such as Sharing. */usr/libexec* and */usr/sbin* contain most daemons such as sharingd, however, some are also shipped in their respective framework. */usr/lib* and */usr/lib/system* contain low-level libraries such as CoreCrypto.

A single service implementation can be scattered across multiple daemons *and* frameworks.

DRIVERS The Wi-Fi driver is a kernel extension and, therefore, resides in */System/Library/Extensions*. The driver is split up into a

2 https://developer.apple.com/documentation

KERNEL EXTENSION	SIZE (KB)
com.apple.iokit.IO80211Family	**1012**
com.apple.filesystems.apfs	1032
com.apple.driver.DspFuncLib	1284
com.apple.driver.AppleIntelBDWGraphicsFramebuffer	1624
com.apple.driver.AirPort.BrcmNIC	**7920**

Table 5: Largest kernel extensions in macOS 10.14.6 loaded on a MacBook Pro 13" (Late 2015) according to the output of `kextstat`.

generic component (`IO80211Family`) and chip-specific plugins (such as `AirportBrcmNIC`).

4.2.2 *Binary Selection*

The purpose of the initial selection process is to identify binaries that may contain relevant code and, thus, sets the scope for the analysis project. To start this process, we can use the system's logging facility (see Section 4.3.1) to monitor for processes that become active when starting a certain system function (e. g., AirDrop). If we identify at least one daemon process, we can crawl through its dependencies recursively by running `otool -L` to find related frameworks and libraries.

Finding all relevant binaries involves crawling through dependencies.

We show part of the discovered dependencies and interactions found for AWDL and AirDrop in Figure 4. While there are user-facing binaries such as the `sharingd` daemon which implements the AirDrop protocol, the most relevant binaries regarding AWDL reside in the kernel, in particular, the generic Wi-Fi driver `IO80211Family` and the device-specific plugin `AirportBrcmNIC`. Each of them includes hundreds of AWDL-related functions, suggesting that the bulk of the protocol stack is implemented here. We found that `IO80211Family` takes care of most of the AWDL frame parsing and creation as well as maintaining the AWDL state machine. The device-specific driver handles time-critical functions such as synchronization.

AWDL and AirDrop are implemented in different parts of the system.

4.2.3 *Interesting Functions and Code Segments*

Due to the size of most binaries that we analyzed, such as the Wi-Fi driver (see Table 5), it is infeasible to analyze the entire program. Instead, it makes sense to identify functions of interest, e. g., those that implement parts of AWDL. Fortunately, Apple does not strip symbol names from (most of) their binaries, such that the symbol table provides useful information and, e. g., lists function names including "awdl." Some of those symbols additionally contain "parse" in their

We look for function names and debug strings to identify interesting code segments.

name (e. g., `parseAwdlSyncTreeTLV`), which helped us understand the calculation of some type-length-value (TLV) fields. Furthermore, debug log statements give hints about the purpose of a code segment inside a function. Therefore, we can search for debugging strings (using `strings`) and their cross-references to find details such as the misalignment threshold in Section 5.2.3.

4.2.4 *Leaked Source Code*

We found several AWDL-related C structs in leaked driver source code.

As one source of information, we used the leaked source code of a dated Broadcom Wi-Fi driver [36]. While leaked code is often not available, we can opportunistically use it to advance our analysis. We found several references to AWDL in the source code, but none of the core functionality. We suspect that Broadcom uses a modular firmware concept with one central repository for a wide range of features. Individual features such as AWDL are made available selectively to their customers, such as Apple. More important than the references to AWDL are some C structs found in the source code. These include key structures such as the Synchronization Parameters TLV and Channel Sequence TLV (more in Section 5.1). The leaked code also contains the source code for the `wl` utility, which provides debugging features for the driver and is further discussed in Section 4.3.

4.2.5 *Dissecting Structures*

To understand the driver's functions, we needed to reconstruct the underlying data structures. The leaked source code shows that most of the AWDL-related functions use an `awdl_info` struct as a first parameter. The `wlc_dump_awdl` function prints internal data in a readable format and, thus, was an ideal target to reconstruct the internal structures from individual `bcm_bprintf` statements as shown below:

```
1 bcm_bprintf(a2, "AWDL master home channel = %d\n",
2             awdl_info->master_home_channel);
```

4.3 SYSTEM LOGGING

macOS comes with several powerful logging facilities.

The complete protocol operation is difficult to comprehend with binary analysis alone. We complemented our static analysis with a dynamic approach to understand, e. g., the semantics of synchronization and election in AWDL. In this section, we discuss dedicated macOS logging and debugging facilities that helped during our analyses. In particular, we used the *Console* application, the `ioctl` interface, Broadcom's leaked `wl` utility, as well as Apple's undocumented *CoreCapture* framework.

4.3.1 *Console*

The *Console* aggregates all system and application logs since macOS 10.12 and includes debug messages from the kernel. Alternatively, one can use the `log` command-line tool to access the same information.

FILTERING FOR INTERESTING OUTPUT It is possible to filter logging output, e. g., by process or subsystem. The *predicate-based filtering* is described in detail on the man page of `log`. For example, we can use

```
log stream --predicate "process == 'sharingd' AND
                        category == 'AirDrop'"
```

to get information about AirDrop, e. g., it involves AWDL and Bonjour.

```
sharingd: [com.apple.sharing:AirDrop] Bonjour discovered
          e275c6497b6c over awdl0 in 12793 ms
```

INCREASING LOG LEVEL While the `--level debug` flag will increase the log verbosity of processes that make use of `os_log`, some components use other means. To receive verbose output from the Wi-Fi driver, we increased the log level using custom boot arguments, which we found by searching for references to the `PE_parse_boot_arg` function in the Wi-Fi driver. We found that the following boot arguments maximize the driver's debug output:

There are several ways to increase the logging verbosity.

```
nvram boot-args="debug=0x10000 \
                 awdl_log_flags=0xffffffffffffffff \
                 awdl_log_flags_verbose=0xffffffffffffffff \
                 awdl_log_flags_config=1 wlan.debug.enable=0xff"
```

With the increased log level, `log` shows additional information such as state transitions ("low power" mode), the access point (AP) channel that the device is connected to (100), and the current channel sequence:

```
kernel: (IO80211Family) com.apple.p2p: AWDL ON: [infra(100) 72%],
        (6/44/44) [44 0 0 0 0 0 0 0 6 44 44 0 0 0 0 0] Low Power
```

4.3.2 *CoreCapture*

CoreCapture is Apple's primary logging and tracing framework for Wi-Fi on iOS and macOS. CoreCapture combines raw protocol traces with traditional log entries and provides snapshots of the device and driver state. CoreCapture is undocumented but was referenced in a `dumpPacket` function that we found in the driver. Since the framework outputs (among other logs and memory dumps) numerous PCAP trace files with a custom header format, David Kreitschmann wrote a Wireshark dissector for CoreCapture that is available online (Appendix C.1). In addition, David published a manual for CoreCapture [109].

Apple's CoreCapture *is a dedicated Wi-Fi logging and debugging framework.*

4.3.3 *Broadcom ioctl Interface*

`ioctl` system calls are a standard way to communicate with devices on Unix-based systems. Apple uses `ioctl`s to configure wireless interfaces such as associating with an AP or creating an independent basic service set (IBSS). Apple provides the header files with the request format, the available request types, and the data structures for macOS 10.5. These old header files can be brought up to date using information from the binary analysis. The `apple80211VirtualRequest` method contains calls to all handler functions. Out of the available 72 request IDs, 40 relate to AWDL. These requests can set several parameters in the driver. Especially useful is the card-specific `ioctl`. It allows wrapping a Broadcom-specific `ioctl` inside an Apple `ioctl`, providing us with a direct interface with the Broadcom driver.

We can access AWDL state information at runtime using `ioctl`.

The Broadcom `wl` utility found in the leaked source code (Section 4.2.4) uses those `ioctl`s to provide several methods to access private information about AWDL operations, which are directly related to the structures found during binary analysis. Although the AWDL-specific driver code was missing in the leaked source code, the `wl` source code contains AWDL related commands and structures. The tool was adapted for macOS by Matthias Schulz using the vendor-specific `ioctl` command in Apple's ioctl interface. In essence, `wl` facilitates querying the current AWDL driver status using commands such as `dump awdl_advertisers`, which shows information about neighboring nodes, including received signal strength indication (RSSI).

Note that it is no longer possible to send Broadcom-specific `ioctl` commands since Apple fixed our reported vulnerability. It enabled *any* local user to issue `ioctl`s (Appendix B.9). The driver now checks for a private *entitlement* security permissions [7] (`com.apple.driver.AirPort.Broadcom.ioctl-access`) which requires a binary signed with an Apple private key. We were able to restore `ioctl` access and, thus, circumvent Apple's check. This involved overwriting the respective permission-checking function in the driver using a kernel extension patching framework [188] and disabling *System Integrity Protection* [6] on macOS.

4.4 NETWORK INTERFACES

Traffic analysis provides a quick look at the protocol stack used by a particular service.

Monitoring the Wi-Fi and Bluetooth network interfaces are a quick way to gather information about a particular service. For example, we can identify known protocols, whether encryption is used, or determine whether we are dealing with an undocumented protocol. In addition, we can learn the active wireless communication channels, the timings of packet transmissions, generally monitor the dynamics of a protocol. In the following, we discuss those tools that we have found to be particularly useful for this purpose.

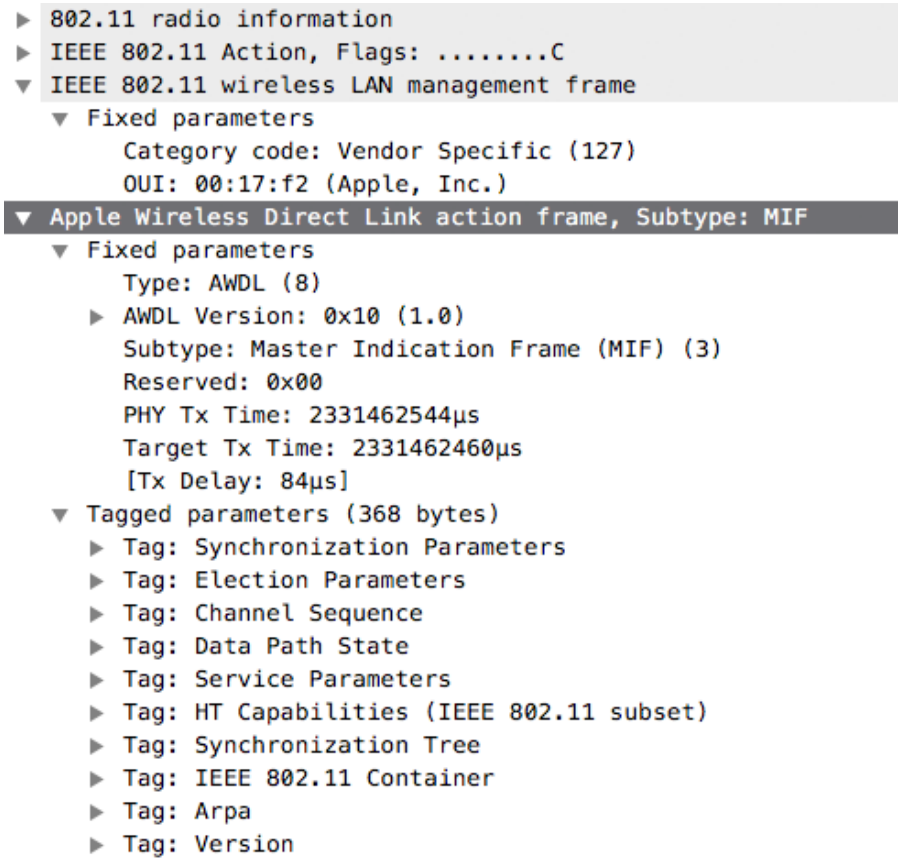


Figure 5: Screenshot of our AWDL Wireshark dissector.

4.4.1 *Wireshark*

Writing custom dissectors while reverse engineering has many benefits.

Wireshark[3] is an open-source network protocol analyzer and supports many standardized but also proprietary protocols. While Wireshark identifies known protocols from network traces, it is also possible to implement custom dissectors. We found that writing such a custom dissector in parallel to the reverse engineering process serves multiple purposes: (1) We iteratively document and validate our findings. (2) It helps to deduce the semantics of certain fields that become apparent in longer time series traces such as monotonically increasing counter or timing fields. (3) It can be used to evaluate experiments such as those in Chapters 5 and 7 by exporting time series data via `tshark`. And (4) at the end of the project, we can directly publish the code to allow reuse by others, which is what we did with our Wireshark dissector for AWDL (Appendix C.1). We show a screenshot of the final dissector in Figure 5.

3 https://www.wireshark.org

4.4.2 *Bluetooth Explorer and Packet Logger*

Apple ships two Bluetooth debugging tools in the *Additional Tools for Xcode* package.[4] The *Bluetooth Explorer* displays nearby Bluetooth Low Energy (BLE) devices and their advertisements in real-time. Apple devices use these advertisements excessively to announce the availability of services such as AirDrop [129]. *PacketLogger*, on the other hand, creates network traces for Bluetooth HCI commands. Wireshark supports PacketLogger-recorded `.pklg` files, which allow for convenient analysis of Bluetooth traces.

4.4.3 *InternalBlue*

InternalBlue can access the lower layers of the Bluetooth protocol stack.

The InternalBlue experimentation framework [127] allows accessing the lower layers of the Bluetooth protocol stack and, thereby, exceeds the capabilities of Packet Logger. InternalBlue operates by patching the firmware of the Bluetooth chip and runs on Linux, Android, and jailbroken iOS devices. While we have not used InternalBlue in this thesis, we list it here as it could support analyzing proprietary Bluetooth extensions.

4.4.4 *Machine-in-the-Middle Proxy*

HTTPS MitM proxies are not always successful.

Encrypted traffic can prohibit us from examining the interesting parts of the protocols. While we could instrument the daemon process and extract packets before transmission (which requires identifying functions that perform those operations), it can be easier to employ machine-in-the-middle (MitM) proxy tools to open the end-to-end encryption, such as of HTTPS [45]. Unfortunately, a MitM proxy is not always successful in intercepting a connection with self-signed certificates, e. g., when certificate pinning is used, so it can be helpful to extract private keys and certificates from the system's keychain.

4.4.5 *Custom Prototypes*

Custom prototypes support the reverse engineering and security analyses by interacting with target devices.

In an advanced stage of the process, we have collected sufficient information to re-implement (part of) the protocol and, thus, are able to interact with the target devices actively. In particular, a custom prototype enables us (1) to validate the correctness of our findings, e. g., if other devices start interacting with our prototype, we can conclude that the frame format is correct, (2) to find out more details about the protocol, e. g., we could determine in which availability windows (AWs) nodes would transmit unicast and multicast frames (Section 5.2), and (3) to conduct protocol fuzzing as part of the security analysis, e. g.,

4 https://developer.apple.com/download/more/?=additional%20tools%20xcode

we found two parsing-related vulnerabilities in AWDL (Section 7.3). The repository links to our AWDL and AirDrop implementations can be found in Appendices C.2 and C.3, respectively.

4.5 KEYCHAINS

Access to private keys and other secure data that is used by a particular service or protocol is highly useful to make educated assumptions about what security mechanisms might be employed. In addition, extracting key material is important to build and test prototypes that prove or disprove working hypotheses, e. g., verifying the requirements for an authenticated AirDrop connection (Section 6.3).

4.5.1 *Login and iCloud Keychains*

In macOS 10.14, there are two types of keychains known as *login* and *iCloud* keychain, respectively. The former is only stored locally on the computer. Also, we believe that Apple is going to deprecate this keychain soon as Apple has made efforts to unify their macOS and iOS codebases [11] and has been moving keychain items for AirDrop from the login to the iCloud keychain. The iCloud keychain was first introduced in iOS and has since been ported to macOS as well. This keychain provides more features such as protection classes, optional synchronization between devices, and improved access control [12]. The *Keychain Access* application is a GUI for displaying and working with either keychain. However, we have found that not all keychain items (e. g., those used by some system services) are displayed.

macOS currently still uses two different keychains.

4.5.2 *Security Framework*

Fortunately, Apple provides a documented API for accessing keychains via the *Security* framework, which additionally is open-source.[5] For our purposes, the `SecItemCopyMatching` function[6] is particularly interesting as it allows retrieving items such as keys from the keychain. The function requires some query parameters to narrow down the items it should return. To get the relevant query parameters of a target program, we can either statically analyze the binary by searching for references to `SecItemCopyMatching` or monitor the process and extract the parameters at runtime using a debugger. In the case of AirDrop, the query consists of three keys: `kSecClass`, `kSecReturnRef`, and `kSecValuePersistentRef`. The value of the latter is a serialized object containing all information required to locate a particular item in the keychain.

The public Security *framework API is used to access private keys and certificates stored in the keychain.*

5 https://opensource.apple.com/source/Security/
6 https://developer.apple.com/documentation/security/1398306-secitemcopymatching

SERVICE	DAEMON	BLE	AWDL	INFRA
AirDrop	sharingd	✓	✓	✗
Wi-Fi Password Sharing		✓	✗	✗
Instant Hotspot		✓	✗	✗
Auto Unlock	sharingd	✓	✓	✗
AirPlay and AirPlay 2		✗	✓	✓
Handoff	useractivityd	✓	✓	✓
Universal Clipboard	useractivityd	✓	✓	✓
iPhone Cellular Calls		✓	✗	✓
Find My Offline	searchpartyd	✓	✗	✓

Table 6: Selection of Apple's *Continuity* services, the main daemons that implement them, and employed wireless technologies. A blank field indicates 'unknown.'

4.5.3 *Accessing Keys of Apple Services*

We can extract key material from Apple services by disabling some security mechanisms.

As a security measure, programs not signed by Apple will not get any results even when using the correct query parameters as Apple uses code signing to implement access control to keychain items. To circumvent this measure, we (1) need to set the correct `keychain-access-group` entitlement (`com.apple.sharing.appleidauthentication` in case of AirDrop or simply the $*$ wildcard) during code signing and (2) disable Apple Mobile File Integrity (AMFI) which prevents program with restricted entitlements from starting by setting the following as a boot argument: `amfi_get_out_of_my_way=1`.

4.6 DISCUSSION AND SUMMARY

We have started to apply our methodologies to other Apple protocols.

Apart from AWDL and AirDrop, we have also successfully applied the vantage points presented in the chapter to other services[7] within Apple's wireless ecosystem. In the following, we give a brief outlook on our current work-in-progress projects. We present the initial results of identifying the key binaries and wireless technologies used within other Apple protocols in Table 6. In particular, we want to highlight that *Auto Unlock* [105] (providing automatic macOS login when an Apple Watch is nearby) uses the sharingd daemon process to bootstrap the unlock procedure. However, it implements a custom distance bounding protocol in the Wi-Fi driver where we could make use of the logging facilities such as CoreCapture. Another example is our ongoing work on the privacy-preserving *Find My* service introduced in iOS 13 and macOS 10.15 that implements a crowd-sourced location

7 https://www.apple.com/macos/continuity/

tracking system. By monitoring the Bluetooth interface, we were able to determine the address randomization interval. Also, we know that the searchpartyuseragent daemon connects to Apple servers, so we can make use of an HTTPS MitM proxy to intercept the messages. Our systematic approach on how to apply novel and existing methods and tools has proven effective. We believe and hope that others will benefit from our experience.

5 APPLE WIRELESS DIRECT LINK

Apple Wireless Direct Link (AWDL) is a proprietary protocol deployed in Apple's main product families such as Mac, iPhone, iPad, Apple Watch, and Apple TV—effectively all recent Apple devices containing a Wi-Fi chip. Apple does not advertise the protocol but only vaguely refers to it as a "peer-to-peer Wi-Fi" technology [12, 13]. Yet, it empowers popular applications such as AirDrop that transparently use AWDL without the user noticing.

AWDL has a key role in many of Apple's wireless services.

AWDL is based on the IEEE 802.11 standard and makes use of vendor-specific extensions that allow custom protocol implementations. Each AWDL node periodically emits custom action frames containing a sequence of availability windows (AWs) indicating its readiness to communicate with other AWDL nodes. An elected master node synchronizes these sequences. Within these AWs, nodes can communicate with their neighbors using a dedicated data frame format. Outside the AWs, nodes can tune their Wi-Fi radio to a different channel to communicate with an access point, or turn it off to save energy.

Synchronized channel hopping enables "concurrent" infrastructure and direct neighbor connection.

In this chapter, we present the results of analyzing and re-implementing this protocol using the methods described in Chapter 4. We first present the frame format and explain the operation in detail. Then, we introduce our implementation and conduct an experimental evaluation of the protocol. Finally, we discuss our findings.

5.1 FRAME FORMAT

We discovered two general frame types used by AWDL: *action* and *data* frames that enable coordination and direct data transfer, respectively. We elaborate on the format of these frame types in the following. Our Wireshark dissector contains more details (Appendix C.1).

AWDL introduces two custom frame types.

5.1.1 *Action Frames*

AWDL uses IEEE 802.11 vendor-specific action frames (AFs), which generally allow vendors with an organizational unique identifier (OUI) to augment IEEE 802.11 frames with arbitrary payloads [95]. The AWDL vendor-specific extension consists of a fixed-sized header and multiple variable-length tags, as shown in Figure 6. A tag has a type-length-value (TLV) structure that consists of a 1-byte *type* field, followed by a 2-byte *length* field that indicates the length of the subsequent *value* byte string. The fixed header includes the AWDL-specific

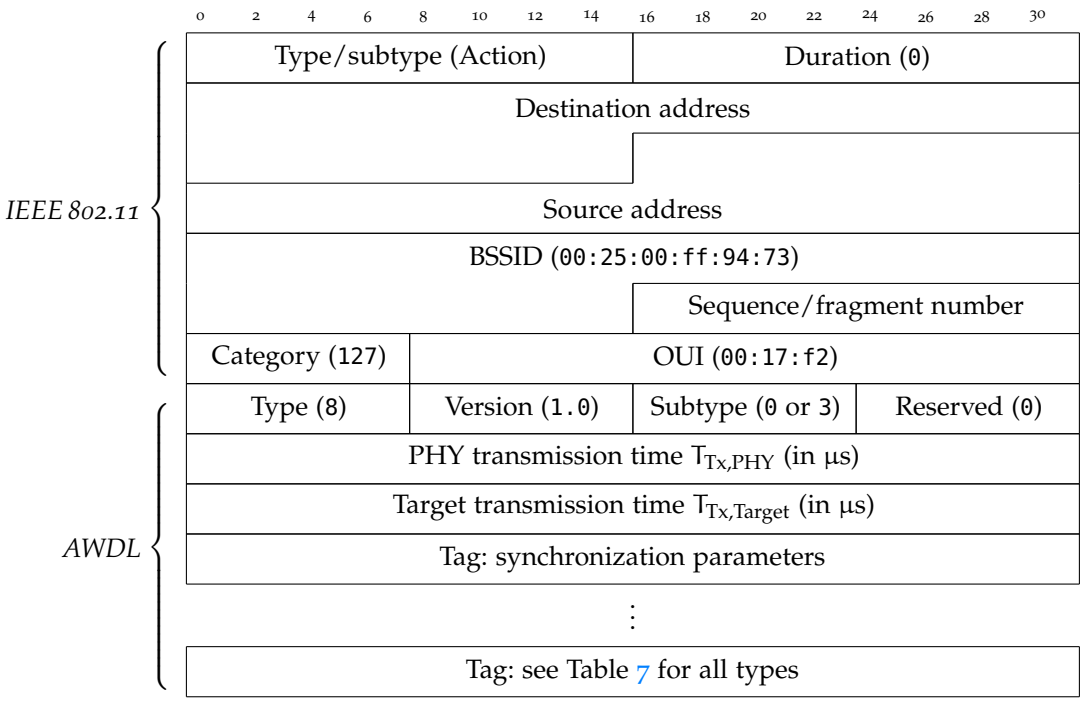


Figure 6: AWDL action frame format.

Two timestamps in the header compensate for the transmission delay.

basic service set identifier (BSSID), OUI, version, and subtype. The timestamp $T_{Tx,Target}$ indicates when the frame was created and, therefore, at which time the included information was up-to-date; while $T_{Tx,PHY}$ is the time when the frame was queued for transmission. Their difference approximates the sender's transmission delay and is used for synchronization purposes. There are two AWDL AF subtypes: periodic synchronization frames (PSFs) and master indication frames (MIFs). These frame types have the same fixed header and differ only in the included set of tags. We show the frame format, excluding the frame check sequence (FCS) at the end of the frame in Figure 6. We first explain the purpose of the subtypes and then discuss tags used in AWDL.

PERIODIC SYNCHRONIZATION FRAME The PSF (subtype `0`) allows for neighbor discovery and synchronization that we further explain in Section 5.2.3. We found the name in a patent [184]. If all participating devices support the 5 GHz band, the PSF is the only frame type also seen on the 2.4 GHz band. PSFs transmissions are scheduled at a fixed time interval that is unaligned to AWs.

MASTER INDICATION FRAME The MIF (subtype `3`) has multiple purposes, such as election (Section 5.2.2) and service discovery (Section 5.2.5). It includes more tags and is sent by all devices in the network regularly. Also, MIFs transmissions are aligned to AWs, as we show in Section 5.4.4.

NAME	TYPE	PSF	MIF	PURPOSE
Synchronization parameters	4	✓	✓	Election and synchronization (Sections 5.2.2 and 5.2.3)
Channel sequence	18	✓	✓	
Election parameters	5	✓	✓	
Election parameters v2	24	✓	✓	
Synchronization tree	20	✓	✓	
Data path state	12	✓	✓	Data transfer (Section 5.2.4)
HT capabilities	7		✓	
VHT capabilities	17		✓	
Service parameters	6	✓	✓	Service discovery (Section 5.2.5)
Service response	2		✓+	
Arpa (reverse DNS)	16		✓	
Version	21	✓	✓	Compatibility

Table 7: Tags used in AWDL. We give the name and type value of a tag grouped by purpose, indicate whether it is included in PSFs or MIFs (✓) and whether it may be present multiple times (+).

TAGS Tags contain the actual control information. The different types are attributable to one of the following purposes: *election and synchronization*, *data transfer*, and *service discovery*. In addition, the *version* tag provides a 1-byte version number that presumably supersedes the version field in the fixed header (see Figure 6). We summarize all tags in Table 7 and discuss them briefly in the following. We found the names in function names and debugging strings during binary analysis. We discuss only some tags in detail in this paper and refer to our Wireshark dissector for the full specification. We omit some *type* values found in the binaries (e. g., 1, 3, and 8) as the anaylzed AWDL versions did not use them and, thus, appear to be deprecated.

There are at least 24 different tag types—but only some of them are used in practice.

The *election and synchronization* processes handle the cooperation of the devices. The respective tags determine, e. g., which node takes the master role and which channels are to be used. Curiously, the synchronization parameters tag includes its own channel sequence, so the separate channel sequence tag appears to be redundant. However, it was always transmitted on current operating system (OS) versions. The *data transfer* components are used to negotiate the parameters for direct connections between devices. For example, the HT/VHT capabilities tags include supported PHY rates and are similar to the ones introduced in the IEEE 802.11n and 11ac amendments [95]. In the data path state tag, each peer announces the BSSID of the Wi-Fi network that it is currently connected to and the real medium access control (MAC) address of the Wi-Fi chip. We believe that this information could be used to offload an AWDL connection to an infrastructure

Some tags contain redundant information.

We are unable to understand every design decision.

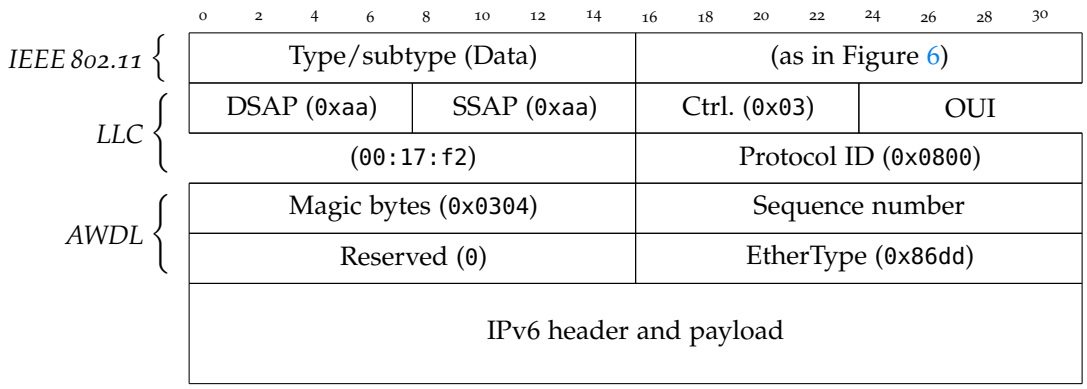


Figure 7: AWDL data frame format.

network if both peers are connected to the same network. However, this would require additional reachability tests due to network policies such as client isolation, and we did not observe such behavior in practice. The *service discovery* components offload multicast DNS (mDNS) and DNS-SD functionality to the AFs. For example, the Arpa tag contains the hostname and a service response tag that can be a PTR, SRV, or TXT resource record. The *version* tag includes the AWDL version (half a byte for major and minor version number each) as well as a device class ID. We found that v4.0 is used in macOS 10.15 and iOS 13; v3.x in macOS 10.13 and iOS 11; and v2.x in macOS 10.12 and iOS 10 (and potentially prior iOS versions). AWDL v1.x is used in macOS 10.11, which does not support the *version* tag. The device class seems to indicate the OS type of the node, e. g., macOS (1), iOS or watchOS (2), or tvOS (8).

There are four major AWDL versions.

5.1.2 *Data Frames*

AWDL uses IEEE 802.11 data frames for data transmission. The To-DS and From-DS flags in the IEEE 802.11 header are set to zero, similar to independent basic service set (IBSS). Consequently, these frames use direct addressing, and the three address fields contain the destination, source, and BSSID. We depict the AWDL data frame format in Figure 7. The BSSID in AWDL frames is always `00:25:00:ff:94:73`, which belongs to the OUI `00:25:00` that is assigned to Apple [96]. The LLC header contains a different Apple OUI (`00:17:f2`) and a protocol ID in the SNAP part. These headers are part of the IEEE 802 standard [94] and allow vendors to implement custom protocols on higher layers. The actual AWDL data header essentially consists of a sequence number and the EtherType of the transported protocol. We identified IPv6 as the only protocol used with AWDL.

AWDL adds a vendor-specific header to IEEE 802.11 frames.

5.1.3 *Addressing for Higher-Layer Protocols*

AWDL is used in conjunction with higher-layer protocols. Therefore, it needs some way to address AWDL nodes via a network layer protocol. This is especially important because AWDL implements privacy-enhancing MAC randomization, i. e., instead of using the Wi-Fi chip's fixed MAC address, it generates a random address every time the interface is activated. In IPv6, address resolution is usually achieved via the Neighbor Discovery Protocol (NDP). Apple, however, does not use NDP for AWDL, but instead generates link-local IPv6 addresses from the source address field contained in the AFs (Figure 6) using the method described in RFC 4291 [82, Appendix A]. This method constructs a link-local IPv6 address based on the MAC address of the network interface. In particular, given a 48-bit MAC address $o_0{:}o_1{:}o_2{:}o_3{:}o_4{:}o_5$, the corresponding link-local IPv6 address is constructed as:

IPv6 addresses of neighbors are derived from their MAC addresses which makes NDP expendable.

Deriving an IPv6 address from a MAC address is a standardized method.

```
1 fe:80::o0^0x02:o1:o2:ff:fe:o3:o4:o5
```

where ^ is the XOR operator. Nodes use this standardized method to add their neighbors to the neighbor table immediately after receiving the first AF without the need or overhead of an additional address resolution protocol such as NDP or Address Resolution Protocol (ARP).

5.2 OPERATION

Based on our analysis, we formulate *hypotheses* regarding the goals and decisions of AWDL's design: (1) leverage existing hardware (Wi-Fi chip), thus building the protocol on top of IEEE 802.11; (2) conserve energy, especially on mobile devices, hence synchronizing and putting the Wi-Fi chip into a power-saving mode during idle times; (3) allow seamless operation of direct and infrastructure-based communication, so enable synchronized channel hopping without disconnecting from an AP; and (4) enable fast service discovery, thus offloading DNS-SD to Wi-Fi frames. Commodity Wi-Fi chips usually have a single RF chain and are, therefore, restricted to a single wireless channel at any given time. For using multiple channels, an adapter needs to switch channels and cannot use the regular wireless connection for short periods of time. This is expected behavior for roaming (scan for available networks while being connected to a network) and power saving features (switch off the radio). As the periods are short, devices need a method for discovery and coordination when to meet on which channel. In the following, we explain the five AWDL phases (1) *activation*, (2) *master election*, (3) *synchronization*, (4) *data transfer*, and (5) *service discovery*.

5.2.1 *Activation*

An external trigger activates AWDL.

Apple uses AWDL as an on-demand communication technology. This means that AWDL is inactive by default, but applications can (temporarily) request activation. There are several possible triggers. For example: AirDrop uses Bluetooth Low Energy (BLE) for activation by sending truncated hashes of the user's contact information (which we discuss in detail in Chapter 6). AirPlay receivers (Apple TV) constantly announce their presence via AWDL. And third-party application may activate the interface indirectly by advertising services via the `NSNetService` API [13].

5.2.2 *Election*

Nodes from a cluster with a single master.

Apple uses three social channels (6 in the 2.4 GHz band and 44 or 149 in the 5 GHz band, depending on the country) for all communication. A node starting its AWDL interface monitors the social channels for some time to discover other nodes in range. If AWDL AFs are received, the node can adopt an existing master, thus, joining an existing cluster. If no frames are received, it assumes the master role itself. In the following, we explain the master election process and the tree-based synchronization structure. In particular, we focus on the mechanisms that make AWDL robust to master nodes leaving or joining the cluster.

The master node emits a clock signal.

ROLE OF THE MASTER NODE AWDL relies on roughly synchronous clocks of all participating nodes in a cluster to enable data transfer. To achieve this, it is paramount that there is exactly one node in the cluster that has the responsibility of emitting a "clock signal." This is the one and, as far as we know, only role of the *master* node. All other nodes in the cluster are called *slaves* and should adopt this signal. In a simple scenario with only two nodes, one node will be the master and another a slave. In larger scenarios, slave nodes might be more than one hop away from the master node. In such cases, intermediate slave nodes will take the role of *non-election masters*, which have the responsibility to repeat the master's clock signal. The synchronization tree tag includes the intermediate master nodes. In particular, each node announces the path to the root, which we call *top master*. In any case, there is only one top master in a cluster.

Each node tries to synchronize before competing in the master election.

MASTER METRIC Who wins the election is decided based on a *metric* field which is included in the election parameters (v2) tag. The node that announces the largest metric value will become the master of that cluster. Apple's patent [185] claims that these metrics could be based on, e. g., available energy resources, CPU load, or signal strength. In practice, however, the metric is just chosen at random. A node that activates its AWDL interface initially sets its metric field to

60 and listens on the social channels for an existing master for two seconds and tries to synchronize. Then, it draws a random number from a predefined range and sets this as its metric. We have found that this range depends on the AWDL version, e. g., 405 to 436 in v2.x and 505 to 536 in v3.x. We assume that this is done for backward compatibility. It guarantees that the master node is running the most up-to-date version, and future protocol extensions can be supported.

MERGING CLUSTERS WITH DIFFERENT MASTERS When two already established AWDL clusters with different master nodes move into proximity of one another, they need to merge such that nodes in the different clusters will be able to discover each other. In AWDL, the process is straightforward as all nodes advertise their current master metric in the election parameters tag. If two nodes with different masters discover each other, they receive the top master metric of the other cluster and can immediately adopt the master with the higher metric. The remaining nodes in the "losing" cluster then follow as soon as the first node advertises the new master metric.

Clusters can merge seamlessly.

LOOP PREVENTION When creating such an election tree hierarchy with multiple levels of sub-masters, loops may occur. To prevent loops and limit the maximum depth of the election tree, each AF contains a list of all nodes up to the top master in the synchronization tree tag. Each node can then make sure that it does not adopt a non-election master if it is already in that node's path. However, it is unclear whether Apple actively uses this feature because (1) we observed in network traces that the synchronization tree often contains invalid node addresses such as `00:00:00:00:00:00` and (2) our AWDL implementation does not use this tag despite being able to communicate with Apple devices.

RE-ELECTION A master leaving the network simply stops sending AFs. There is no sign-off message. Therefore, a missing master can only be detected by other devices after a certain *no-master* timeout that is fixed to 96 AWs ($\approx$ 1.5 s). Another node will then take the place of the old master. As this node was already in sync with the old master, other slave nodes do not need to re-synchronize but simply adopt the new master. In other words, AWDL is robust to master churn, i. e., a leaving master does not interrupt communication, and a new master is seamlessly adopted.

AWDL is robust to masters joining or leaving the cluster.

THE ROLE OF RSSI The received signal strength indication (RSSI) values of received AFs are used to filter out possibly unstable or asymmetric connections. In particular, AWDL nodes drop frames when the RSSI is below a so-called *edge sync* threshold set to -65 dBm (or -78 if AirPlay is used). Frames from the current master node are

Two RSSI thresholds prevent "master flapping."

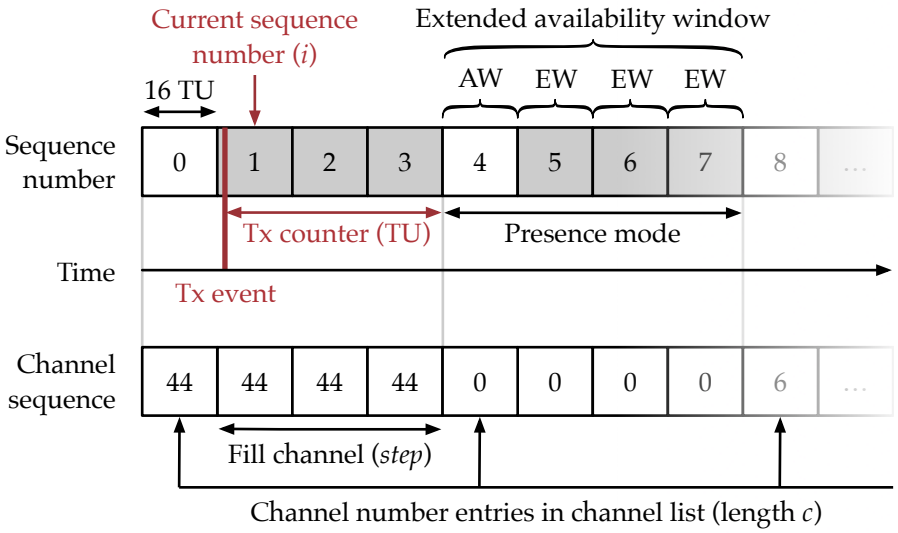


Figure 8: Structure of AWs and mapping to channel sequence.

accepted with a lower RSSI. These frames receive a grace *slave sync* threshold of 5 dBm. Lowering the threshold for the master frames allows for a certain variance in the RSSI. We assume that this was done to reduce "master flapping," where a node frequently adopts a new master because it regularly ignores frames until the *no-master* timeout occurs.

5.2.3 *Synchronization*

Synchronized nodes can meet on the same the channel to communicate.

Synchronization is tightly coupled with the election process since nodes always try to synchronize with their elected master. In the following, we describe how AWDL structures time and how nodes align their time reference with that of their master so that they can communicate when they are on the same channel. We introduce the concept of AWs, i. e., short fixed-length time slots during which communication is possible. These windows have a static length, but can be extended using extension windows (EWs). Finally, we show how the start of an AW is determined using fields from the synchronization parameters tag. We visualize the key concepts and variables in Figure 8.

AVAILABILITY WINDOW AWs indicate the times during which a device will be available for communication. These windows must be synchronous in a cluster such that every node starts an AW at the same time. Timing in AWDL is based on time units (TUs) where 1 TU = 1024 µs [95, page 141]. In the AWDL implementation, an AW is always set to be 16 TUs long. The length of an AW and all other values presented in this section are contained in the synchronization parameters tag shown in Figure 9. In theory, different configurations are possible, but we found that only fixed values are used.

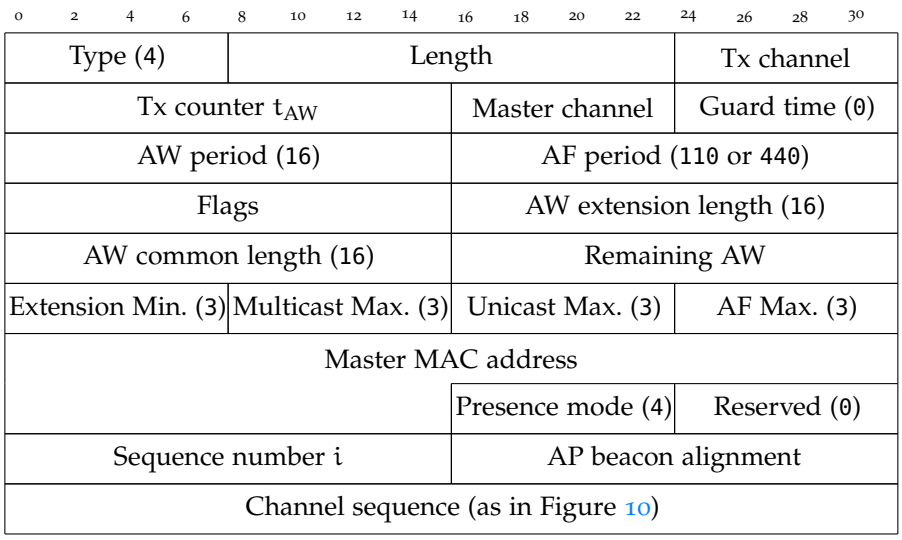


Figure 9: AWDL synchronization parameters format.

PRESENCE MODE AND EXTENSION WINDOWS For reduced power consumption, a peer can indicate that it is not listening in every AW. A presence mode p of 4, which is the only value used in Apple's AWDL implementation, means that a peer is only listening for every fourth window. If a node is transmitting or receiving data, it may extend its time spent on the channel. This is called an EW. A presence mode of 4 leaves space for three EWs of 16 TUs. Also, AWDL allows configuring different numbers of unicast, multicast, and AF EWs, but these fields are currently always set to 3 and, thus, align with the presence mode. Figure 9 shows the parameters transmitted in the synchronization parameters tag. Given the static configuration, the effective smallest time unit in use is four consecutive AWs/EWs. For the remainder of this paper, we use the term extended availability window (EAW) to refer to such a 64 TU time slot.

The AWDL protocol allows for complex AW configurations—but only a fixed one is used in practice.

CALCULATING THE START OF AN AVAILABILITY WINDOW Each slave node needs to synchronize its clock to that of its master node. For achieving this, the master node announces the *start of the next AW*. When transmitting an AF, the master includes the number of TUs to the next EAW t_{AW} as well as the sequence number of the current AW or EW i. We mark these values in red in Figure 8.

In each AF, the master indicates its current AW sequence number and start of the next AW.

As the driver sets these values when creating the frame, some time passes until the frame is finally transmitted via the Wi-Fi interface. AWDL tries to compensate for this transmitter delay by including two additional timestamps in the fixed header of each AF: the PHY and target transmission times $T_{Tx,PHY}$ and $T_{Tx,Target}$, respectively. Ideally, $T_{Tx,Target}$ is set when the frame is created, and $T_{Tx,PHY}$ just before the frame is transmitted via the interface. In the macOS driver, both timestamps are set in the Wi-Fi driver and, therefore, do not account for delays induced by the distributed coordination function (DCF) that

controls medium access [95]. In the leaked Broadcom source repository [37], we found a configuration constant (`D11AC_TXC_AWDL_PHYTT`) for the D11 real-time core [163] indicating that the D11 core is capable of setting $T_{Tx,PHY}$ just before transmission. However, we did not verify whether macOS makes use of this mechanism.

In any case, a device receiving an AF from its master at time T_{Rx} can approximate the start of the next AW T_{AW} as follows:

$$T_{AW} = t_{AW} \cdot 1024 - (T_{Tx,PHY} - T_{Tx,Target}) + t_{air} + T_{Rx}. \quad (1)$$

However, AWDL ignores the airtime t_{air} since it is in the order of sub-µs in a typical close-range Wi-Fi scenario, and the accepted synchronization error is 3 ms.[1] We experimentally evaluate the achievable accuracy in Section 5.4.

5.2.4 *Data Transfer*

Before sending a data frame, AWDL needs to make sure that the receiver is listening on the same channel.

Apple exposes a dedicated `awdl0` network interface to the system that enables transmitting Ethernet frames to other AWDL nodes. AWDL uses a vendor-specific frame format header when transmitting those frames over the air. When sending to a particular neighbor, a node needs to calculate the AWs during which both nodes are tuned to the same channel and only transmit frames during those AWs. Also, AWDL adapts its channel sequence according to the current outgoing traffic load. In this section, we explain the AWDL channel sequence announcement that builds upon the synchronized AWs and indicates whether a node is available for communication and, if so, to which channel its radio is tuned to. In particular, we explain the AWDL channel encoding and channel-to-AW mapping.

AWDL uses three different channel encodings.

CHANNEL ENCODING AWDL announces its channel sequence redundantly as part of the synchronization parameters tag as well as in the dedicated Channel Sequence tag. The latter appears to be used in recent versions of AWDL as the contained channels use the IEEE 802.11 *operating class* encoding that supports channels with bandwidths above 40 MHz, thus enabling communication with devices supporting VHT data rates. In contrast, the *legacy* encoding used in the synchronization parameters tag only supports channels with a maximum bandwidth of 40 MHz. We have found a third encoding during binary analysis that simply uses the channel number and is exclusively used by the Apple Watch, which only supports the 2.4 GHz band.

MAPPING THE CHANNEL SEQUENCE TO AVAILABILITY WINDOWS The channel sequence maps channel numbers to AW sequence num-

1 In the function `awdl_recv_action_frame`, a misalign metric is increased if the difference between a projection from a previous calculation and a new calculation of T_{AW} is larger than 3 ms.

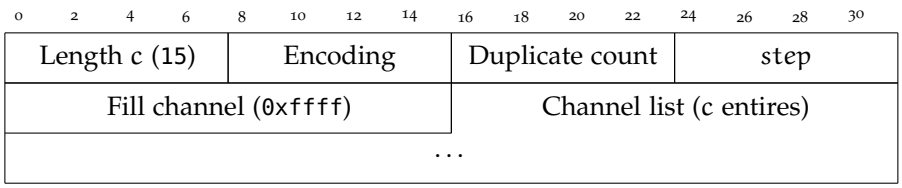


Figure 10: AWDL channel sequence format.

bers. Figure 10 shows a channel sequence tag with a fixed number of $c + 1 = 16$ channel entries. While the number of entries is fixed, the sequence is expandable with the step field similar to the presence mode in the synchronization parameters, so that one channel entry can span multiple AWs and EWs.[2] Setting step to 1 means that the channel will be active for one additional AW. However, Apple always sets this field to 3, meaning that the channel will be active for four AWs or one EAW. Thus, the channel sequence aligns fully to the presence mode in the synchronization parameters tag. Given an encoded channel sequence and an AW sequence number i, an AWDL node can calculate the currently active channel C for any peer based on the following calculation:

The channel sequence aligns with the EAW sequence.

$$C = i \mod ((c + 1) \cdot (step + 1)) \tag{2}$$

As Apple uses fixed values for c and step, the announced channel sequence covers $(15 + 1) \cdot (3 + 1) = 64$ AWs which takes about one second ($64\,\text{AW} \cdot 16\,\text{TU/AW} = 1048576\,\mu\text{s} \approx 1\,\text{s}$) and is repeated periodically.

5.2.5 *Service Discovery*

Apple relies on DNS service discovery (DNS-SD) [76], also known as *Bonjour*, to discover services. Curiously, Apple decided to support two different ways to integrate DNS-SD with AWDL: piggybacking service announcements on AFs and using the IPv6 data path to send mDNS frames, both of which we explain in the following.

There are two options for service discovery in AWDL.

PIGGYBACKING ONTO ACTION FRAMES AWDL nodes can piggyback DNS-SD responses such as SRV, PTR, or TXT records directly onto its AFs in type 2 tags (Table 7). While this approach violates the ISO/OSI layer model, it has the advantage that service changes are immediately visible to nearby peers. It appears that piggybacking is optional as our AWDL implementation (Section 5.3) does not implement type 2 tags while still being compatible with Apple devices. Instead, our implementation exclusively relies on the IPv6 data transport for service discovery.

Piggybacking violates the ISO/OSI layer model.

2 Note that the extension with step works only with the *fill channel* field set to 0xffff, which was the case in all our captured frames.

USING IPV6 DATA TRANSFER DNS-SD is also supported by sending mDNS directly over AWDL's IPv6 network interface. While this approach appears to be the most straightforward way to implement service discovery, there is a caveat. AWDL uses dedicated multicast EAWs. Recall that data frames need to be scheduled on a particular AW and that mDNS frames are multicast, so that a sender cannot calculate a common AW with a single receiver. In fact, we have found that Apple schedules multicast frames onto two fixed EAWs: 1 and 10. These EAW are set in all channel sequences, which makes sure that nearby nodes receive all service announcements. From an implementation perspective, this mandates the use of different transmission queues for multicast and unicast frames, which is what we use in our AWDL implementation.

AWDL uses dedicated AWs for transmitting multicast frames.

5.3 RE-IMPLEMENTATION

We present our AWDL prototype called *Open Wireless Link (OWL)*. We implement our prototype in plain C for performance reasons and to facilitate porting the code to other platforms. In the following, we present a high-level architecture and discuss supported platforms as well as future work.

5.3.1 *Architecture*

We designed OWL to provide a similar system integration as Apple's implementation. In particular, we wanted to expose a dedicated network interface and maintain the neighbor table based on the reception of AFs (see Section 5.1.3). We depict the resulting architecture and integration of our AWDL daemon in Figure 11. At its core, the daemon uses an event loop (*libev*[3]) that (1) listens on the Wi-Fi interface for new IEEE 802.11 frames using *libpcap*,[4] (2) updates neighbor information, (3) periodically schedules the transmission of AFs that carry information used for peer discovery, synchronization, and election procedures, (4) listens on a virtual Ethernet interface for new traffic from the host system, and (5) schedules data frame transmissions in the correct AWs.

We mimic the system integration of Apple's implementation.

When a new Wi-Fi frame is received on the monitoring interface, we check whether the frame is an AWDL action or data frame. Regular IEEE 802.11 frames are dropped by a BPF filter so that the card only forwards frames with the AWDL-specific BSSID `00:25:00:ff:94:73`. If we receive an AF, we derive the link-local IPv6 address from the source Ethernet address and add both to the system's neighbor table. Based on the included tag fields, we run the election and synchronization mechanisms, as described in Section 5.2. If we receive a data frame,

3 http://libev.schmorp.de
4 https://github.com/the-tcpdump-group/libpcap

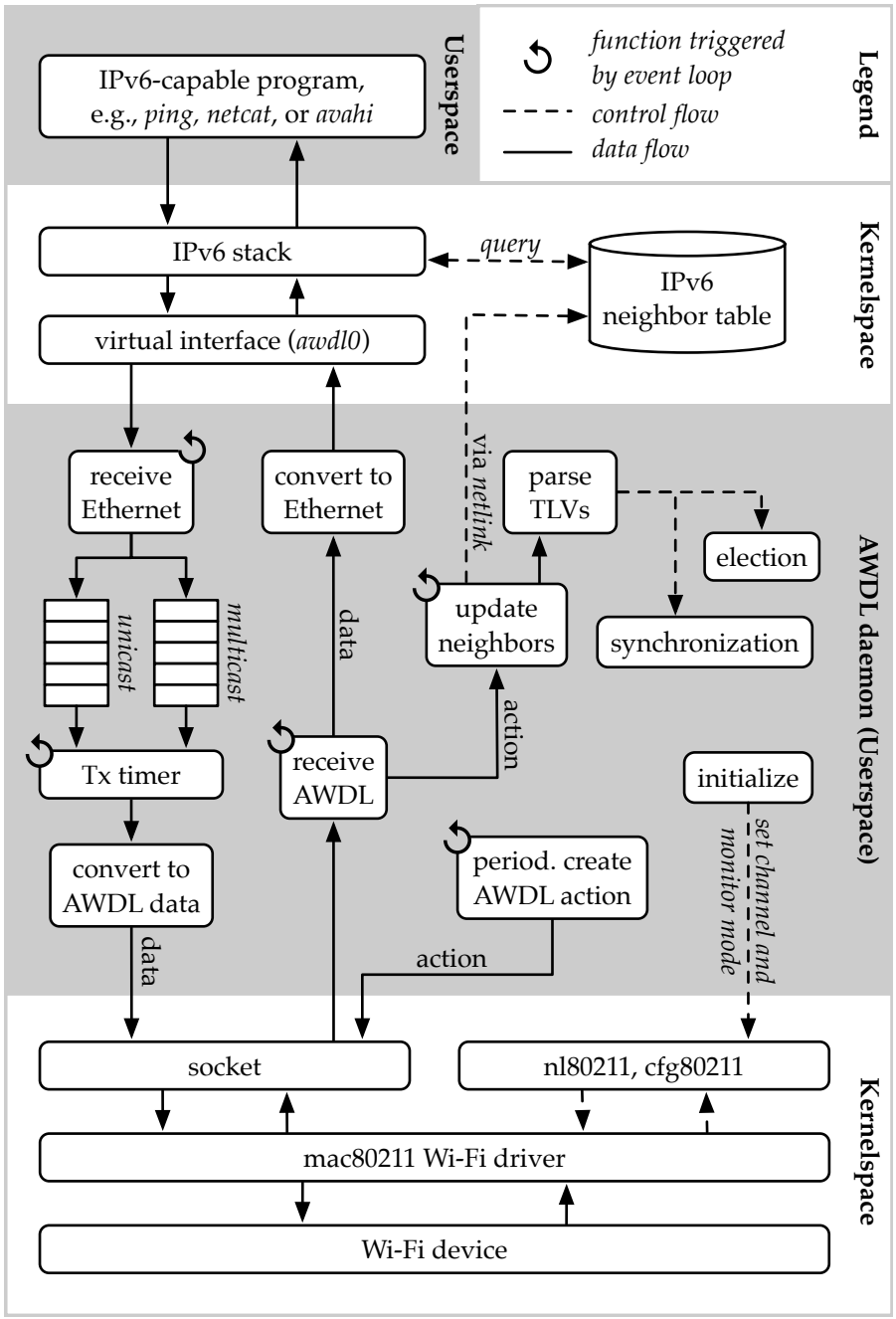


Figure 11: Architecture of our AWDL prototype and its integration with the Linux networking stack.

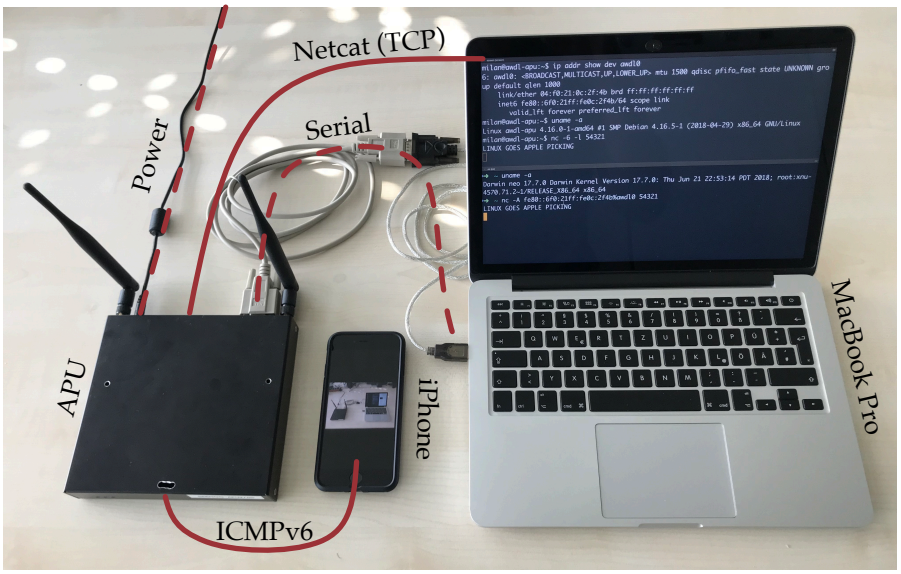


Figure 12: AWDL demonstrator setup consisting of a Linux-based APU board, an iPhone 8, and a MacBook Pro. The terminal on the MacBook's screen shows a working TCP-over-AWDL connection between the APU board and the MacBook.

we strip the AWDL data header, replace it with a regular Ethernet header, and forward the frame to the virtual `awdl0` interface. We do the inverse for Ethernet frames that we receive from `awdl0` and add the AWDL sequence number from an internal counter. Unicast and multicast frames are put in their respective queues, and a timer takes care of sending the frames out on the correct EAW. In addition, the daemon periodically emits AWDL action frames that it builds from its internal synchronization and election state.

5.3.2 *Supported Platforms and Future Work*

OWL currently supports Linux and macOS.

OWL currently runs on Linux and macOS. On macOS, it can act as a drop-in replacement for Apple's own AWDL implementation. On Linux, we require a Wi-Fi card that supports active monitor mode and frame injection, such as a Qualcomm Atheros AR928X. Figure 12 shows a hardware setup that demonstrates cross-platform communication between Linux and Apple devices.

With a few hardware and platform requirements, Android and Windows could be supported in the future.

Since our prototype is written in C, it should be possible to port the code to other OSs. However, we have the following dependencies that each target platform needs to provide: (1) a Wi-Fi card supporting active monitor mode with frame injection to be able to receive and send IEEE 802.11 frames, (2) a means to change the Wi-Fi channel such as *nl80211*, (3) access to the system's IPv6 neighbor table, and (4) a facility to create virtual network interfaces such as TUN/TAP. In principle, this should allow implementations on Windows or Android.

On the latter, monitor mode and frame injection can be enabled using the Nexmon framework [163].

Our prototype currently lacks a channel-switching mechanism that would be required to follow nodes to a different channel. However, since AWDL devices usually meet on one social channel (6, 44, or 149), our prototype still works by continually listening on a fixed channel.

5.4 EXPERIMENTAL EVALUATION

We analyze the runtime behavior of AWDL in different scenarios to (1) validate our findings of the previous sections and (2) assess the performance of the protocol. First, we describe our test setup. Then, we look at the master election and synchronization accuracy in an idle scenario without data transmissions. We further analyze the channel hopping behavior and throughput performance.

5.4.1 *Test Setup*

We record frames on all AWDL channels simultaneously.

Our test setup consists of one monitoring device and several Apple devices. Our monitor device is an APU board [149] equipped with two Qualcomm Atheros QCA9882 Wi-Fi cards to support simultaneous sniffing on two different channels that are tuned to AWDL's primary (44) and secondary (6) channel. Both Wi-Fi cards support hardware timestamping, which mitigates variable delays in the receiver's OS stack. As each Wi-Fi chip has its own clock, the timestamps in both recorded traces are misaligned. Therefore, we start each experiment with a calibration phase: we tune both chips to a common channel and let them record multiple frames. Post-experiment, we calculate the timestamp difference of frames that were received by both cards on the common channel. We use the median difference to correct the clock offset and align both traces. All the following experiments were conducted inside a Faraday tent to minimize interference. Our test devices include an iPhone 8 (iOS 11.2.2), an iPad Pro 10.5" (iOS 11.0.3), an iMac (Late 2012, macOS 10.12.6), and a MacBook Pro (Late 2015, macOS 10.12.6).

5.4.2 *Master Election*

We monitor the dynamics of the election process.

In our first experiment, we analyze the master election process. We observe an AWDL cluster in an idle state, meaning that no data transmission takes place, and the only observed frames are AFs. We use a setup consisting of an iPhone, iPad, iMac, and MacBook. We activate the AWDL interface by selecting the *sharing panel* in one device that causes a BLE scan and activates the AWDL interface of other devices in range (on iOS, this only works if the device is

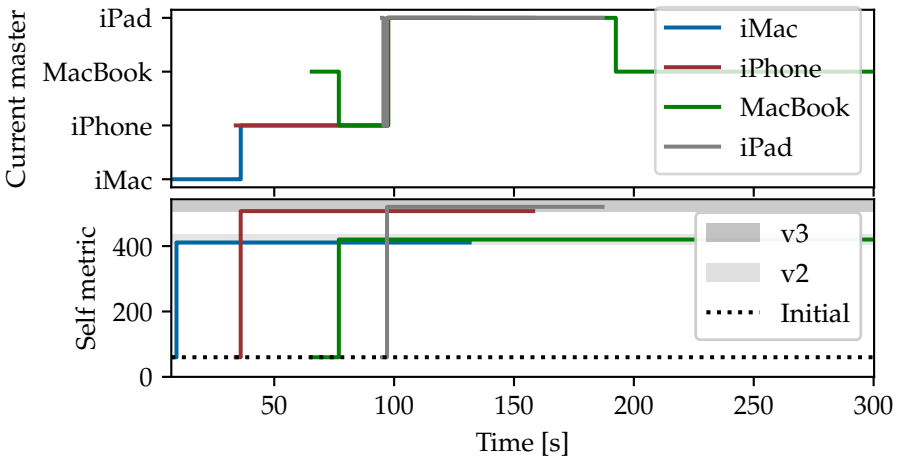


Figure 13: Master selection (top) and self metric (bottom) over time. The grey shaded areas show the value ranges used in the different versions of AWDL.

unlocked). To get more interesting results, we let the different devices join approximately 30 s after one another.

Figure 13 shows the currently selected master of each node. First, the iMac creates the AWDL cluster and consequently selects itself as the master. As soon as the iPhone joins, it takes over the master role, and the iMac adopts it. The MacBook runs the same version as the iMac and, thus, after having discovered the AWDL cluster, it also adopts the iPhone as the master node. The iPad briefly adopts the existing master, but then immediately takes over this role as it selects a higher self metric than the iPhone: Figure 13 also shows the current self metric of each node over time. We show the initial value of 60 and the implemented ranges for the different versions of AWDL. Finally, all nodes successively leave the cluster (Wi-Fi turned off) until only the MacBook remains. Since the iMac and the MacBook run an older version of AWDL, they are only selected as master if none of the newer versions are present in the cluster.

We expected most of these results. What is interesting, however, is that an already existing master node can be "overtaken" by another node running the same version of AWDL. This indicates that Apple's AWDL implementation is rather simplistic. Each node keeps the initial self metric only for a short time and then selects a higher random value from the version-dependent range *irrespective of whether it has found an existing master or not*.

5.4.3 *Synchronization-to-Master Accuracy*

We want to evaluate how well AWDL's master election and synchronization mechanism work. To this end, we monitor the PSF and MIF exchanges between several different nodes. We run another idle ex-

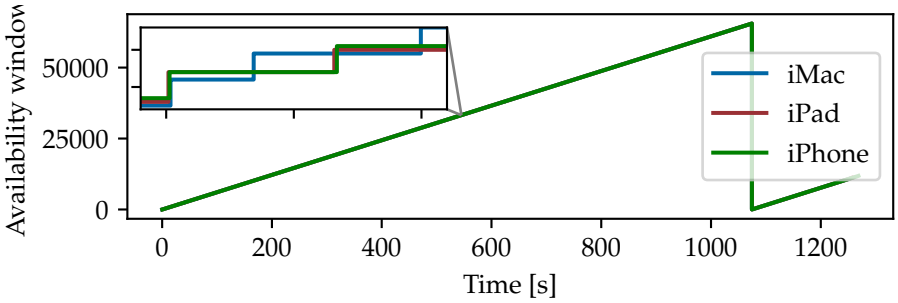


Figure 14: AW sequence number. Shows the sequence number wrap after approximately 18 min ($\approx 2^{16}\,AW \cdot 16\frac{TU}{AW} \cdot 1024\frac{\mu s}{TU}$).

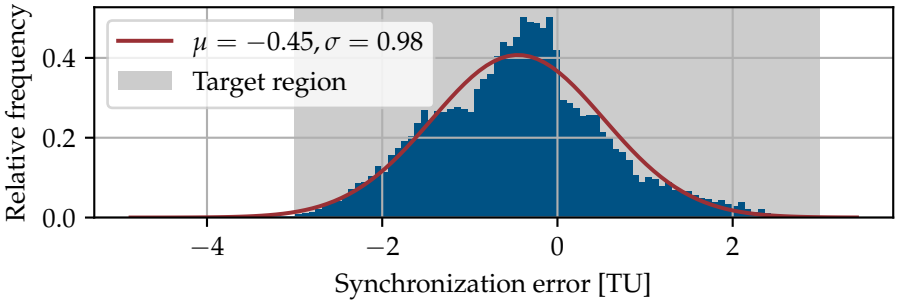


Figure 15: Distribution of synchronization error ξ.

periment over a longer period (20 min) with three nodes. Figure 14 shows the AW sequence number each node advertises. While Figure 14 indicates that synchronization works in principle (all nodes follow the same AW sequence number incline), we can see that the AW sequence number steps are not perfectly aligned. We are interested in the magnitude of this synchronization offset. We adapt Equation (1) to compute the synchronization error ξ between a slave $\mathcal{S}$ and its master $\mathcal{M}$. Assuming a constant airtime t_{air} and given two AFs from $\mathcal{S}$ and $\mathcal{M}$ with a sequence number in the same EAW recorded at the sniffer at time $T^{\mathcal{M}}_{Rx}$ and $T^{\mathcal{S}}_{Rx}$, respectively, we calculate ξ as

AWDL accepts a synchronization error of 3 TU which it manages to achieve.

$$\begin{aligned}
\xi &= T^{\mathcal{M}}_{AW} - T^{\mathcal{S}}_{AW} \\
&= \left(t^{\mathcal{M}}_{AW} \cdot 1024 - \left(T^{\mathcal{M}}_{Tx,PHY} - T^{\mathcal{M}}_{Tx,Target}\right) + t_{air} + T^{\mathcal{M}}_{Rx}\right) - \\
&\quad \left(t^{\mathcal{S}}_{AW} \cdot 1024 - \left(T^{\mathcal{S}}_{Tx,PHY} - T^{\mathcal{S}}_{Tx,Target}\right) + t_{air} + T^{\mathcal{S}}_{Rx}\right) \\
&= \left(t^{\mathcal{M}}_{AW} - t^{\mathcal{S}}_{AW}\right) \cdot 1024 - \left(t^{\mathcal{M}}_{Tx} - t^{\mathcal{S}}_{Tx}\right) + T^{\mathcal{M}}_{Rx} - T^{\mathcal{S}}_{Rx}, \\
&\quad \forall i_{\mathcal{S}}, i_{\mathcal{M}} \text{ with } \lfloor \frac{i_{\mathcal{S}}}{p} \rfloor = \lfloor \frac{i_{\mathcal{M}}}{p} \rfloor.
\end{aligned} \tag{3}$$

In Figure 15, we can see that the synchronization error approximates a Gaussian distribution with a mean value of -0.45 TU and a standard deviation of 0.98 TU. Figure 15 also shows that the target maximum synchronization error of 3 TUs is met in more than 99 % of all cases.

While the results are within the target region, the relatively large synchronization error leads to the conclusion that only a fraction of

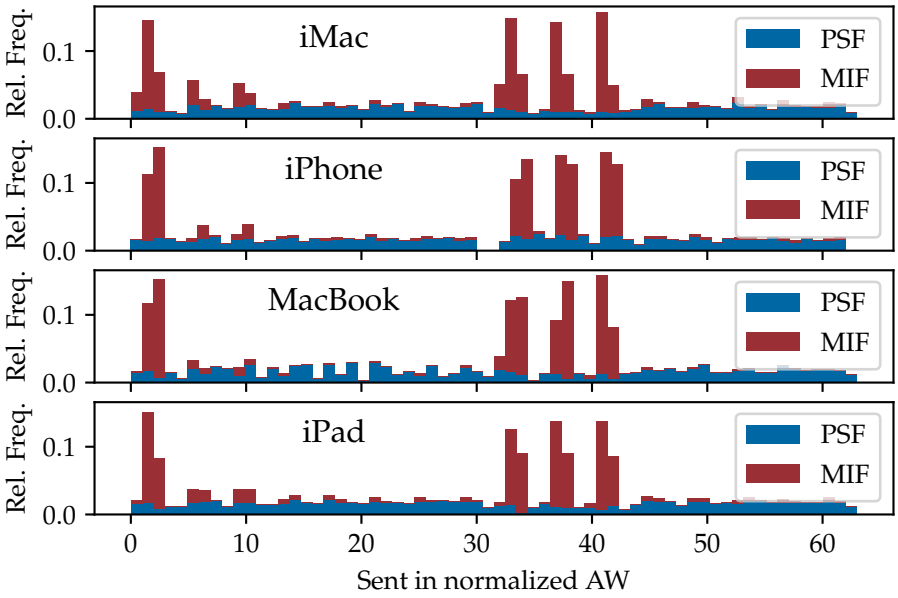


Figure 16: Activity in a full channel sequence period.

each EAW can reliably be used for communication, and the 3 TUs should be used as a guard interval. Assuming a guard interval on either side of the EAW, this means that only $1 - \frac{2 \cdot 3\,\mathrm{TU}}{64\,\mathrm{TU}} \approx 90.6\,\%$ of the interval can be used for communication. We believe that the main source of synchronization error lies in the calculation of the transmission delay t_{Tx}. Equation (1) assumes that $T_{Tx,PHY}$ is set exactly at the moment when the frame is being transmitted via the Wi-Fi radio after the frame has already been enqueued and additional DCF back-offs have expired. However, we have found that in macOS, $T_{Tx,PHY}$ is set in the driver right after the AF is created and before the DCF has been run, even though the Wi-Fi firmware appears to be able to set $T_{Tx,PHY}$ closer to the actual transmission using the D11 real-time core.

A guard interval would help to mitigate the synchronization error.

5.4.4 *Channel Activity*

We want to find out when AFs are usually transmitted. For this, we consider the *idle* scenario from Section 5.4.2 again. Figure 16 shows when frames (MIF and PSF) are transmitted during an EAW by the different nodes. Each bin represents a single AW (16 TU). We notice that MIFs are mostly sent at the beginning of the first and second half of the entire sequence. We confirm this by looking at the advertised channel list: Figure 17 shows the relative frequency of the different channel numbers in the advertised channel list. We can see that slot 9 is always reserved for channel 6.

We also notice that there is a distinct difference in the sending behavior of MIFs and PSFs. While MIF transmissions adhere to the advertised channel sequence, PSFs are sent at arbitrary times. This

PSF and MIF transmissions follow different schedules.

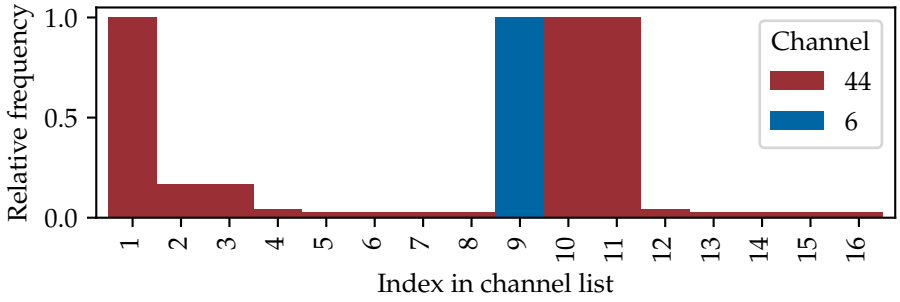


Figure 17: Advertised channel list in *idle* scenario.

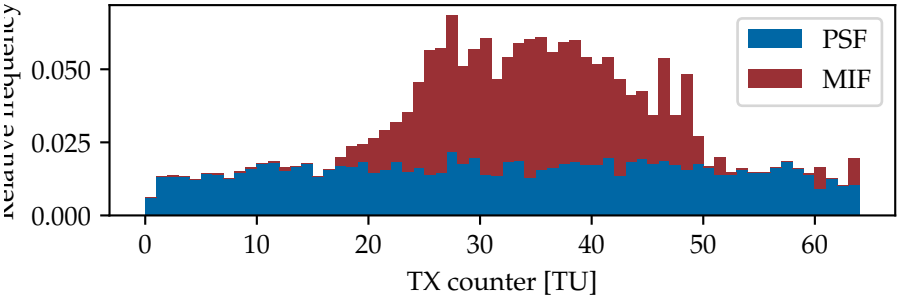


Figure 18: Activity within a single EAW.

is due to the AF period in the synchronization parameters tag (see Figure 9) that is either set to 110 or 440 TU and does not align with the 64 AWs that cover one channel sequence. We believe that this design decision was made to accelerate the bootstrapping of new nodes that have not yet synchronized to a master node. We find supporting evidence as *all* nodes send PSFs, no matter if they are master or not, which will increase the chance that a new node will find an existing AWDL cluster.

Figure 18 shows the transmission times within a single EAW at TU resolution. We can see that MIFs are primarily sent during the middle of one EAW. While it might increase contention on the MAC layer, sending in the middle of an EAW increases the chance that a node receives a transmission even if they are not perfectly synchronized.

5.4.5 *Throughput and Channel Hopping*

We know that AWDL makes use of the highest possible PHY data rates for unicast frames if both participants support them, and if the signal strength is high enough. We want to evaluate the impact of AWDL's channel hopping on the throughput of a TCP connection. Unfortunately, Apple drops packets for regular TCP and UDP servers that directly bind to the `awdl0` interface. This meant that running measurement software such as `iperf` was not immediately possible. As a solution, we built an AWDL–TCP proxy via the `NSNetService` API [13] that whitelists the advertised port. In essence, the proxy

We wrote an AWDL-TCP proxy to use `iperf` via AWDL.

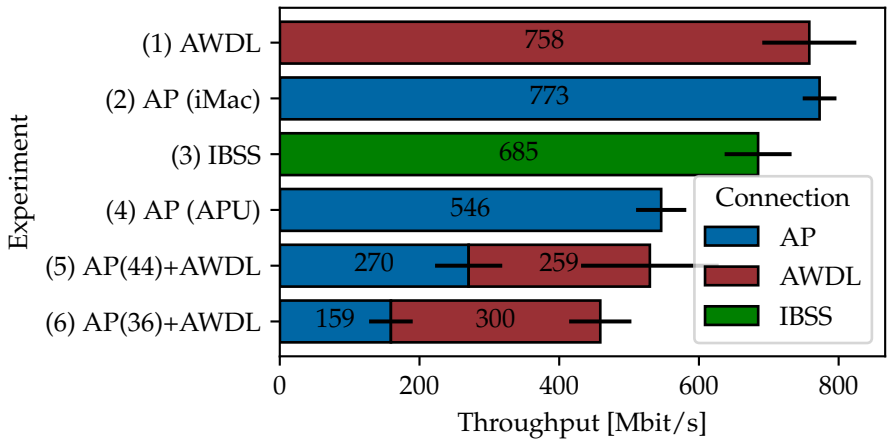


Figure 19: Throughput measurements.

server advertises a service via DNS-SD and listens for incoming TCP connections. The proxy client component connects to it. Both proxy endpoints also allow TCP connections via the loopback interface such that regular TCP services can connect to the loopback interface without modification, and forward the TCP traffic via the `NSNetService` connection. The proxy tool is available at [168].

TCP THROUGHPUT We evaluate the TCP throughput performance of AWDL compared with an access point (AP) connection and IEEE 802.11 IBSS mode. We measure the throughput using a custom AWDL–TCP proxy and `iperf` using the MacBook and iMac. We repeat each of the following 10-second experiments 50 times and show the results in Figure 19. The error bars indicate the standard deviation.

We compare AWDL to other Wi-Fi modes of operation.

In (1), the iMac acts as an AP, and the MacBook connects as a client, while (2) shows the throughput when using an AWDL connection between the same devices. Evidently, (1) and (2) result in similar throughput, demonstrating that bandwidth is only limited by the hardware capabilities of the communicating nodes. (3) shows the performance of IBSS which performs 10–12 % worse than (1) and (2): we observed that the MCS selection mechanism for IBSS on macOS is erratic and does not always choose the maximum supported values even when the signal-to-noise ratio is high.

Channel hopping reduces overall throughput by about 13 %.

Next, the APU operates as an AP. The APU's Wi-Fi card supports a maximum PHY data rate of 866.7 Mbit/s (MCS 9, two streams, 80 MHz bandwidth) while iMac and MacBook both support three streams. Thus, we first measure the maximum achievable throughput using only the APU AP (4) and see that, indeed, the maximum bandwidth is reduced by approximately 30 %. Finally, we are interested in the impact of AWDL's channel hopping on the throughput of concurrent connections. In (5) and (6), the MacBook creates one connection each to the APU (acting as an AP) and to the iMac (via AWDL). In (5), the AP operates on AWDL's primary channel 44. Here, the cumulative

STATE	AIRTIME	CHANNEL LIST (c = 16)
Low Power	25.0 %	p s p p
Idle	37.5 %	p p p s p p
Data+Infra	50.0 %	p p p p i i i i s p p p i i i i
	75.0 %	p p p p p p i i s p p p p p i i
Data	100.0 %	p p p p p p p p s p p p p p p p

Table 8: A subset of AWDL states and corresponding channel lists where p and s are the primary (44) and secondary (6) AWDL channels, respectively, and i is the channel of the AP.

throughput is similar to (4), while the bandwidth between the two connections is uniformly distributed. When (6) the AP operates on a different channel (36), the cumulative throughput drops by about 13 %. This confirms the intuition that channel switching reduces throughput.

CHANNEL HOPPING We found that AWDL adopts its channel sequence according to the traffic volume on the interface. When there is no traffic (such as in the *idle* scenario), AWDL allocates at least 25 % of the channel sequence to the social channels (see slots 1, 9, 10, and 11 in Figure 16). As the load increases, AWDL may allocate all EAWs for itself. We depict the various channel allocation states in Table 8.[5] The table shows that (1) at least 25 % of the time is allocated for AWDL (*low power* state), (2) there is *always* a switch to channel 6 in slot 9 possibly for backward compatibility, and (3) at least 25 % of the time is reserved for the AP connection if the node is connected to an AP. In our throughput experiment, either the *data* or the *data+infra* (50 %) state was active.

Channel allocation dynamically adjusts to network load.

5.5 DISCUSSION AND SUMMARY

In this section, we discuss AWDL's robustness, complexity and overhead, energy efficiency, and conduct an initial security assessment of AWDL and its OS integration.

5.5.1 *Robustness*

We found that compared with its direct competitor Wi-Fi Direct, AWDL is extremely robust to nodes joining and leaving the cluster. The group owner (GO) in Wi-Fi Direct is the equivalent of the master node in AWDL. However, GO essentially acts as an AP and, therefore, has more responsibilities such as relaying data between two

AWDL achieves robustness by giving the master limited responsibilities.

5 We found string references for 25 states in total (including a *real-time* mode and different combinations) during binary analysis, which we will not further discuss in this dissertation.

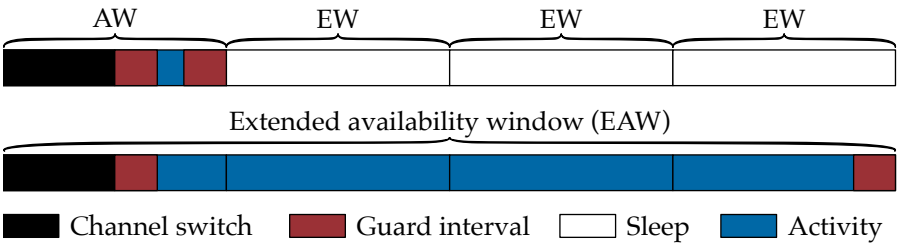


Figure 20: Time spent on channel switching, guard interval, and resulting airtime that can be used for communication when using AWs/EWs vs. EAWs.

non-GO nodes. In AWDL, the master's only purpose is to emit the clock signal for synchronization. Once synchronized, a cluster could survive without a master for some time, assuming that the clocks do not drift apart quickly. Also, active connections between any pair of nodes will not be disturbed by a leaving master and a new node seamlessly taking over. In contrast, Wi-Fi Direct requires a complete group re-establishment when the GO leaves, which disrupts any ongoing communication.

5.5.2 *Complexity and Overhead*

AWDL has a complex protocol definition that supports various configurations using AWs and EWs. We were surprised to see that Apple settled for a static and rather simple configuration, making the complex concepts obsolete. In addition, we found redundant information that bloats the size of the AWDL AFs.

Using only EAWs might mitigate practical limitations of the Wi-Fi driver.

(EXTENDED) AVAILABILITY WINDOWS AWDL, as implemented in current OSs, allows for highly configurable operation configurations (see synchronization parameters in Figure 9). However, all current implementations use a fixed channel sequence length of 16 and do not differentiate between AWs and EWs but exclusively use the longer EAWs (compare Figure 8). The reason why Apple prefers EAWs might have to do with the time that is required to perform a channel switch operation in the Wi-Fi chip. We found that a channel switch operation takes at least 8 ms ($\approx$ 8 TU) using `dump chanswitch` of the `wl` utility. In combination with a guard interval that is necessary to cope with the accepted synchronization error of 3 TU, this would leave only 2 TU airtime for communication, assuming that the EWs are reserved for an energy-conserving *sleep* state. When using EAWs, the temporal efficiency increases from about 12.5 % to more than 78 % while sacrificing opportunities to save energy. We visualize this difference in Figure 20.

REDUNDANCY AWDL AFs contain redundant information such as the current master address that is announced in the synchronization parameters, election parameters, and election parameters v2 tags. Also, the service response tag often encodes the same information multiple times, such that the service instance string and device name can be seen three times in a frame when AirDrop is active. We assume that Apple added features incrementally but kept old fields to retain backward compatibility.

5.5.3 *Energy Efficiency*

Our working hypothesis was that energy efficiency was one of the primary design goals of AWDL. The insights obtained from our experimental analysis do not support this hypothesis. We have found that even in the so-called *low power* state, AWDL is active for at least 25 % of the time during which the Wi-Fi chip is active. We suspect that Apple sacrificed energy efficiency for more reliable operation: the exclusive use of long EAWs makes the system more robust against synchronization error.

5.5.4 *Security*

AWDL does not offer any encryption and, thus, traffic remains unprotected. However, Apple employs a default filter on sockets that prevents unaware processes from listening on the AWDL interface accidentally.

OPEN AWDL CONNECTION We have found that AWDL connections do not feature any security mechanism. All action and data frames are sent in plain text and without authentication. AWDL delegates security functions to the transport and application layer, e. g., AirDrop uses TLS 1.2 (Chapter 6). The approach appears to be an informed decision to implement application-dependent policies: a device might be trusted for sending an image file via AirDrop, but not for remote-controlling a Keynote presentation.

Apple leaves security to the upper layers.

DEFAULT SOCKET FILTER While an AWDL connection can be considered insecure, Apple made sure that AWDL-unaware services are *not* advertised via the `awdl0` interface that would otherwise be accessible by unauthenticated nearby adversaries. Developers need to explicitly use a dedicated API (i. e., `NSNetService`) to opt-in for the use of AWDL, which we did to implement our TCP proxy. Alternatively, programs can set an XNU-specific `SO_RECV_ANYIF`[6] socket option to make the socket listen on the `awdl0` interface. Also, the `awdl0` interface is activated only on demand and deactivated once no more traffic is

There is an opt-in mechanism for using AWDL.

6 https://opensource.apple.com/source/xnu/xnu-4570.41.2/bsd/sys/socket.h

registered, thus, minimizing the time window for an attack. This could be considered an "accidental" security mechanism because the main reason for the timeout has likely to do with energy conservation.

5.5.5 *Summary*

We reconstructed the frame format and the operation of AWDL, a complex undocumented protocol, and complemented our findings with a Wireshark dissector and an open-source AWDL implementation. We believe that public knowledge of this wide-spread proprietary protocol is vital to allow independent security audits, to stimulate innovation and research below the application layer, and to enable high-throughput cross-platform communication on the neighbor scope. Also, we experimentally evaluated AWDL and showed that the mean synchronization accuracy is about -0.45 ms. The maximum achievable throughput is only limited by the devices' supported PHY data rates if the nodes are not actively using an infrastructure network. If channel switching is required, the cumulative throughput of two concurrent connections drops by about 13 %.

6 APPLE AIRDROP

AirDrop is a system service that allows iOS and macOS users to exchange files between devices using Apple Wireless Direct Link (AWDL) as transport. In this chapter, we present the protocol based on a reverse engineering analysis using the methods described in Chapter 4. In particular, we discuss the different discoverability settings from a user perspective. Then, we describe the technical protocol flow and explain the difference between authenticated and unauthenticated connections. Finally, we briefly present our open-source implementation (Appendix C.3) and summarize our findings. The analysis is based on macOS 10.13 and iOS 12. However, our implementation continues to interoperate with macOS 10.15 and iOS 13, suggesting that our findings are still up-to-date.

6.1 DISCOVERABILITY USER SETTING

When opening the sharing pane on an iOS device (see top left screenshot in Figure 21), nearby devices appear in the user interface depending on their discoverability setting [10]. In particular, devices can be discovered (1) by *everybody* or (2) by *contacts only*. Alternatively, (3) the *receiving off* setting disables any AirDrop connection requests. AirDrop requires Wi-Fi and Bluetooth to be enabled. By default, Wi-Fi and Bluetooth are enabled, and AirDrop is set to *contacts only*. In addition, we found that devices need to be unlocked to be discovered. Based on a user study that we present in Appendix A, we found that 80 % of the participants enable AirDrop (59.4 % in *contacts only* and 20.6 % in *everybody* mode) while the other 20 % disabled it. In the remainder of this thesis, we assume that a target device has AirDrop enabled and is unlocked.

AirDrop supports an everybody *and a* contacts only *mode.*

6.2 PROTOCOL WORKFLOW AND USER INTERACTIONS

The AirDrop protocol consists of three main phases (discovery, authentication, and data transfer), which we reverse engineered by employing a machine-in-the-middle (MitM) HTTPS proxy and analyzing the `sharingd` daemon and `Sharing` framework on macOS 10.14.5 where AirDrop is implemented. In the following, we describe each protocol phase in detail and visualize the complete protocol flow, including user interactions in Figure 21.

AirDrop's protocol stack involves BLE, AWDL, Bonjour, TLS and HTTP.

The *sender* initiates the discovery procedure and transfers the data while the *receiver* responds to requests. We walk through the indi-

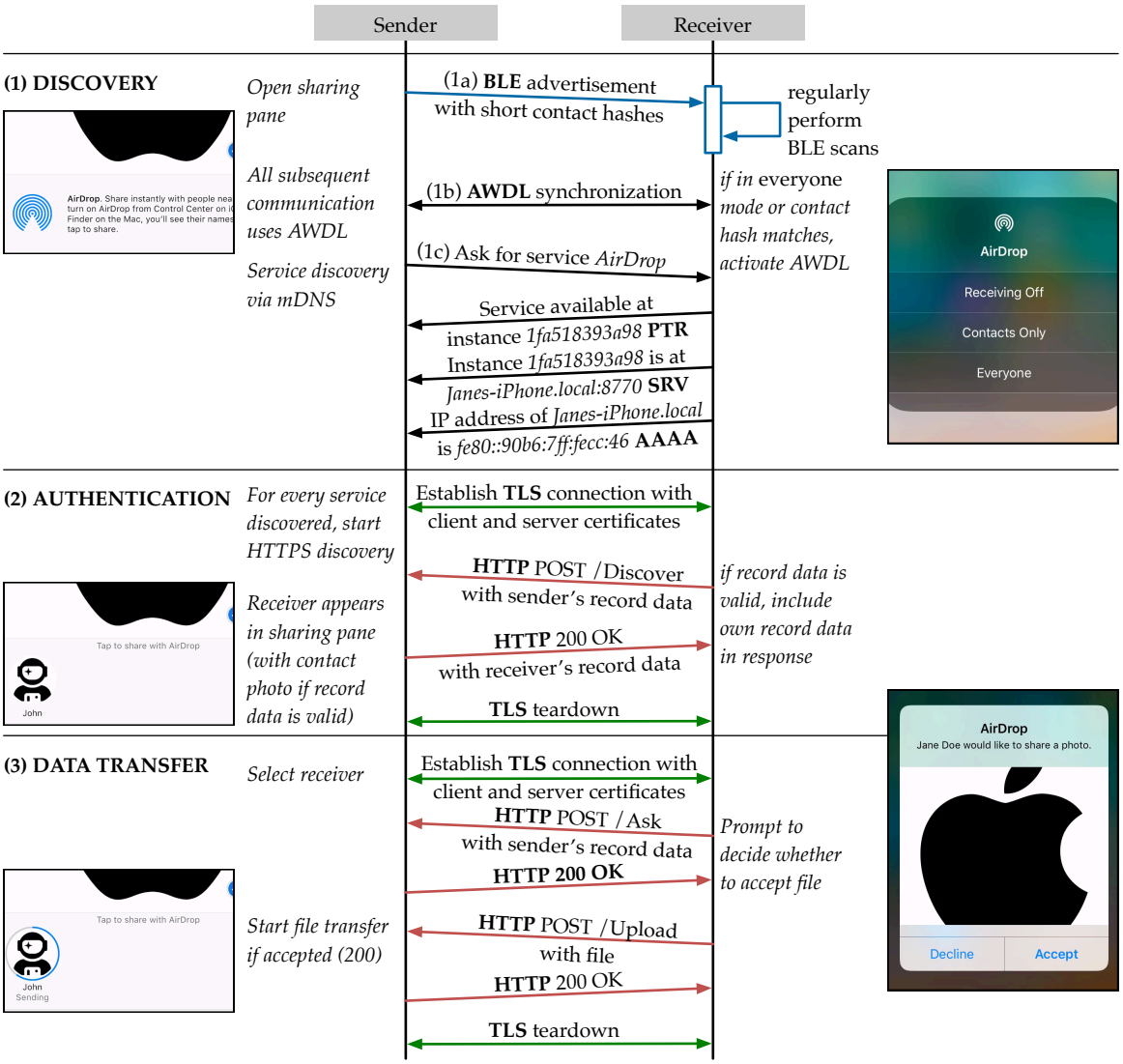


Figure 21: Typical AirDrop protocol workflow including user interactions.

vidual phases of *discovery*, *authentication*, and *data transfer*. (1a) The sender emits Bluetooth Low Energy (BLE) advertisements, including its hashed contact identifiers (see Section 7.3.1 for details), while the prospective AirDrop receiver regularly scans for BLE advertisements. (1b) The receiver compares the sender's contact hashes with contact identifiers in its address book if set to *contacts-only* mode. If there is at least one match or if the receiver is in *everyone* mode, the receiver activates its AWDL interface. (1c) Using multicast DNS (mDNS) and DNS service discovery (DNS-SD), the sender starts to look for AirDrop service instances via the AWDL interface. (2) For each discovered service, the sender establishes an HTTPS connection with the receiver and performs a full authentication handshake (*Discover*). If authentication is successful, the receiver appears as an icon in the sender's UI. (3) When the user selects a receiver, AirDrop sends a request containing metadata and a thumbnail of the file (*Ask*). The receiver decides whether they want to accept it. If so, the sender continues to transfer the actual file (*Upload*).

Next, we discuss the client and server TLS certificates and explain their usage in combination with the sender's and receiver's record data to establish authenticated connections.

6.3 (UN)AUTHENTICATED CONNECTIONS

AirDrop always tries to establish what we call an *authenticated* connection. An authenticated connection can only be established between users with an Apple ID and that have each other in their address books. Authentication involves multiple certificates and CAs that we depict in Figure 22. In order to authenticate, a device needs to prove that it "owns" a certain contact identifier c_i such as email address or phone number associated with its Apple ID, while the verifying device checks whether it has c_i in its address book. When establishing a TLS connection, AirDrop uses a device-specific Apple-signed certificate σ_{UUID} containing a universally unique identifier (UUID). The UUID does not correspond to any contact identifiers, so AirDrop uses an Apple-signed *Apple ID validation record (VR)* containing the UUID and all contact identifiers $c_1, \ldots, c_n$ that are registered with the user's Apple ID in a hashed form. The validation record and the certificate σ_{UUID} are retrieved from Apple when the user logs in to their iCloud account on their device for the first time and is then used in any subsequent AirDrop connection. Formally, *VR* is a tuple:

Client and server have to prove that they are each others' contacts.

Authentication involves Apple-signed TLS certificates and Apple ID validation records.

$$VR = (UUID, SHA\text{-}256(c_1), \ldots, SHA\text{-}256(c_n)) \quad . \tag{4}$$

The signed validation record VR_σ additionally includes a signature and a certificate chain (Figure 22):

$$VR_\sigma = (VR, sign(\sigma_{VR}, VR), \sigma_{VR}, \sigma_{AAI_2}) \quad , \tag{5}$$

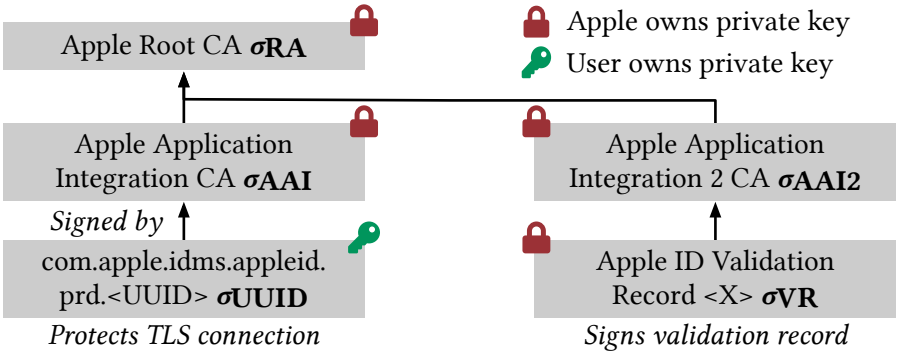


Figure 22: Certificates and CAs involved in AirDrop. Boxes contain the certificates' *common names*.

where $sign(\sigma, X)$ is the signature of σ over X. When authenticating, a node computes SHA-256 over each normalized[1] contact identifier in its address book, compares them with the hashes contained in the presented VR_σ, and verifies that the UUID matches the certificate of the current TLS connection. The latter effectively prevents a replay attack where an attacker would use VR_σ with a different TLS certificate.

Authentication "failures" are masked transparently to support non-contacts.

As AirDrop is supposed to work with non-contacts as well, AirDrop transparently treats a connection as *unauthenticated* if the sender or receiver fails to provide an Apple-signed TLS certificate or matching validation record. Consequently, AirDrop establishes unauthenticated connections with devices that use a self-signed certificate and provide no validation record. AirDrop's authentication mechanism appears to be cryptographically well-designed. However, we show in Section 7.2 how to downgrade an authenticated connection to an unauthenticated one by mounting a denial-of-service (DoS) attack and launch a MitM attack on the data transfer.

6.4 RE-IMPLEMENTATION

We re-implement the AirDrop protocol in Python.

We implement an AirDrop protocol prototype in the Python programming language that we call *OpenDrop* (Appendix C.3). In its current version, OpenDrop supports the announcement and discovery of AirDrop services via mDNS/DNS-SD and implements the complete HTTPS authentication and data transfer. When running on Linux, we require our AWDL implementation (Appendix C.2). On macOS, OpenDrop can also directly use Apple's AWDL implementation.

OpenDrop exposes a command-line interface that allows *finding* other devices to *send* files to as well as *receive* files from other devices. The command stores the results in a temporary file, so that a subsequent send command can use it. A typical workflow for sending files looks like this:

1 Phone numbers are hashed in a normalized form, e. g., the phone number string `+49 123 4567890` would be hashed as `491234567890`.

```
$ opendrop find
Looking for receivers. Press Ctrl+C to stop ...
Found  index 0  ID eccb2f2dcfe7  name John's iPhone
Found  index 1  ID e63138ac6ba8  name Jane's MacBook Pro
^C
$ opendrop send -r 0 -f /path/to/some/file
Asking receiver to accept ...
Receiver accepted
Uploading file ...
Uploading has been successful
```

Our prototype is not complete. In particular, OpenDrop currently does not send out BLE beacons during discovery and does not perform proper authentication, i. e., it does not verify that UUID in the certificate matches the one in the validation record.

6.5 DISCUSSION AND SUMMARY

Analyzing AirDrop has been more straightforward compared to investigating AWDL.

AirDrop was the second protocol within Apple's wireless ecosystem that we reverse-engineered. Compared to AWDL, the process was much easier for several reasons. (1) AirDrop uses a standardized protocol stack such as mDNS and HTTPS, so analyzing traces with Wireshark was possible out of the box. (2) We already knew which binaries contained the implementation from our AWDL analysis. (3) The AirDrop HTTPS protocol uses requests and responses, making the flow easy to follow, whereas AWDL's synchronization and channel hopping are much more complex.

From a security perspective, we think that AirDrop is generally well and cleverly designed. Tying a TLS certificate to a validation record provides authentication without requiring an active Internet connection. However, in Chapter 7, we show that subtle weaknesses in the BLE discovery mechanism and UI design can be exploited to mount DoS and even MitM attacks.

7

DOS ATTACKS AND MITIGATIONS FOR AWDL AND AIRDROP

We have analyzed and re-implemented Apple Wireless Direct Link (AWDL) and AirDrop in the two previous chapters. With a deep understanding of those two protocols, we can conduct a security analysis. In this chapter, we present three denial-of-service (DoS) attack vectors. In particular, we found (1) a DoS attack aiming at the election mechanism of AWDL to deliberately desynchronize the targets' channel sequences effectively preventing communication, (2) a machine-in-the-middle (MitM) attack which intercepts and modifies files transmitted via AirDrop, effectively allowing for planting malicious files, and (3) two DoS attacks on Apple's AWDL implementations in the Wi-Fi driver that allow crashing Apple devices in proximity by injecting specially crafted action frame (AF). In addition, we found several privacy issues in AWDL that allow for long-term device tracking, works *despite* medium access control (MAC) address randomization, and may even reveal personal information such as the name of the device owner (over 75 % of experiment cases). The results of the privacy analysis are presented in Appendix A, which includes a user study that we reference throughout this chapter. We responsibly disclosed our findings and proposed mitigations to Apple, which have responded by issuing several updates for its operating systems (OSs). Appendix B provides a complete list.

We present three different DoS attacks vectors targeting AWDL and AirDrop.

7.1 DOS DESYNCHRONIZATION ATTACK ON AWDL

AWDL does not employ any security mechanisms. Instead, Apple decided to leave security mechanisms to the upper layers. Thus, while end-to-end confidentiality and integrity can be achieved using a secure transport protocol such as TLS, AWDL frames remain vulnerable to forgery, which renders any upper layer using AWDL susceptible to attacks on availability. In this section, we present a novel DoS attack that targets AWDL's synchronization mechanism to prevent two nodes from communication with each other. In the following, we first recap the synchronization procedure and describe a novel *desynchronization* attack aiming at minimizing the channel sequence overlap of two targets. Next, we evaluate the attack's performance and present an effective mitigation method. Finally, we compare this attack to reactive jamming.

AWDL frames can be forged as there is no authentication or integrity protection.

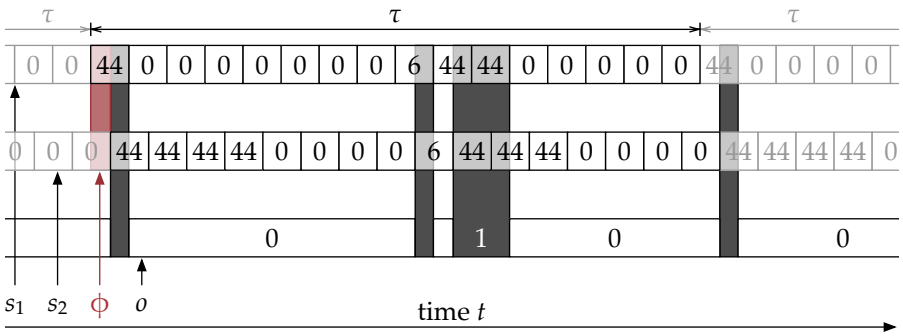


Figure 23: AWDL synchronization depicting period, phase offset, and the overlap function of two channel sequences.

7.1.1 *Modeling Channel Sequence Overlap*

We briefly explain the purpose of synchronized channel sequences (details in Section 5.2) and then derive the channel sequence overlap.

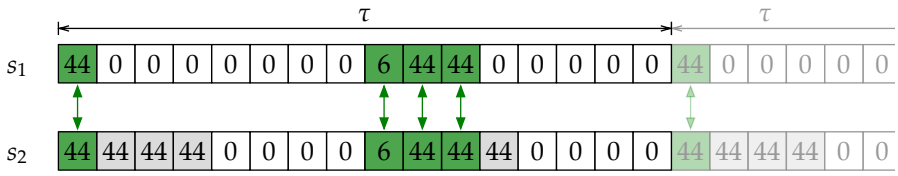
At its core, AWDL uses a channel hopping mechanism to enable "simultaneous" communication with an access point (AP) and other AWDL nodes on different channels. This channel hopping is implemented as a sequence of so-called extended availability windows (EAWs). For each EAW, a node indicates if it is available for direct communication and, if so, on which channel it will be. Each node announces its own sequence s consisting of 16 EAWs regularly. We call the length of such a full 16-EAW sequence a *period* τ. Each EAW has a length of 64 TU, so $\tau \approx 1\,\text{s}$. Figure 23 depicts these and the following concepts.

To allow nodes to meet and exchange data on the same channel, they need to align their sequences in the time domain. AWDL nodes elect a common master and use its AFs as a time reference to achieve synchronization. In each AF, the master node includes the current AW/EW sequence number i (0 to $2^{16}-1$) and the time until the next EAW starts based on its local clock t_{AW}. When receiving an AF from its master at time T_{Rx}, a slave node approximates T_{AW}, the start of the next EAW $i+4$ in local time as (simplification of Equation (1)):

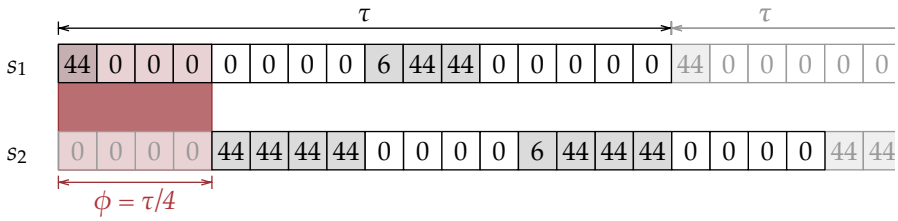
$$T_{AW} \approx t_{AW} + T_{Rx} \tag{6}$$

and corrects its clock accordingly. The phase ϕ denotes the effective clock offset between two nodes.

A node transmits data frames to another AWDL node during EAWs, where the channels of both nodes are the same. We denote the *overlap* as the communication opportunities normalized over one period: an overlap of one means that two nodes are on the same AWDL channel during all 16 EAWs, while zero means that they are never on the same channel. Formally, we define the *overlap* O as the integral over the overlap function o of two sequences s_1 and s_2 taking into account



(a) In normal operation, two channel sequences result in non-zero overlap, allowing two nodes to communicate. In this example, they can communicate during four out of 16 EAWs.



(b) A phase shift of a quarter period ($\phi = \tau/4$) results in zero overlap preventing the two nodes from communicating with each other.

Figure 24: Sketch of the desynchronization attack.

the phase where $s(t)$ is the τ-periodic continuation of a sequence, i. e., $s(t + n\tau) = s(t), \forall n \in \mathbb{N}$. Then,

$$o(s_1, s_2, \phi, t) = \begin{cases} 1 & \text{if } s_1(t) = s_2(t - \phi) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and

$$O(s_1, s_2, \phi) = \int_{t=0}^{\tau} o(s_1, s_2, \phi, t) \quad . \tag{8}$$

7.1.2 *Desynchronizing Two Targets*

The attack involves three steps.

We exploit AWDL's synchronization mechanism to reduce the channel overlap by inducing an artificial *phase* offset between two targets. In order to succeed, the attacker needs to (1) get recognized as the master by both targets, (2) communicate with each target separately in order to (3) induce a phase shift that results in zero (or at least minimal) channel overlap. Figure 24 depicts the non-zero overlap in normal operation and the zero overlap as the result of the desynchronization attack. We describe the three steps in the following.

The attacker needs to become the master.

(1) WINNING THE MASTER ELECTION The master election in AWDL is based on a numeric comparison of two values that are transmitted in the election parameters tag. The first value is called *metric*, and each node draws one randomly upon initialization. The numeric range of the *metric* is bounded and depends on the AWDL version that runs on the node. The second value is called *counter* and

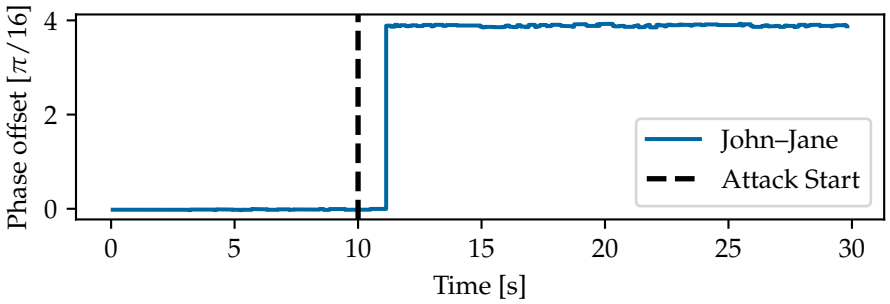


Figure 25: Phase offset between two targets before and after mounting a desynchronization attack that induces a phase shift of $\phi = \tau/4$.

is initialized to a random value and increases linearly over time while the node is elected as a master. Given the metric and counter values of two nodes A and B as (m_A, c_A) and (m_B, c_B), respectively, then, A wins the master election if

$$c_A > c_B \vee (c_A = c_B \wedge m_A > m_B) \tag{9}$$

and loses otherwise. To consistently win the election, the attacker sets c and m to their maximum values.

AWDL nodes accept unicast AFs.

(2) UNICASTING ACTION FRAMES The attacker needs to send different synchronization parameters to each target without the other one noticing. We have found that while AFs are typically sent to the broadcast MAC address `ff:ff:ff:ff:ff:ff`, AWDL nodes also accept unicast AFs. Therefore, the attacker can unicast their AFs to make sure that only the intended target receives them.

(3) INDUCING THE PHASE SHIFT To desynchronize two targets, the attacker needs to send incompatible synchronization parameters that will result in a controllable offset. We explain how the attacker calculates the relevant parameters i and t_{AW} for both targets. Let us assume that the attack starts at time T_s. An AF sent to the *first* target at some time T_{Tx} with $t = \left\lfloor \frac{T_{Tx} - T_s}{1024} \right\rfloor$ (in TU) will include the following parameters:

$$i = \left(\left\lfloor \frac{t \mod 64}{16} \right\rfloor + 4 \left\lfloor \frac{t}{64} \right\rfloor \right) \mod 2^{16} \quad \text{and,} \tag{10}$$

$$t_{AW} = 64 - t \mod 64 \quad . \tag{11}$$

For the *second* target, the attacker will calculate $t^\phi = \left\lfloor \frac{T_{Tx} - T_s - \phi}{1024} \right\rfloor$ and compute i^ϕ, t^ϕ_{AW} analogously to Equations (10) and (11) while replacing t with t^ϕ. We verify the correctness of these calculations experimentally and show the resulting phase offset between two targets for a target phase $\phi = \tau/4$ in Figure 25.

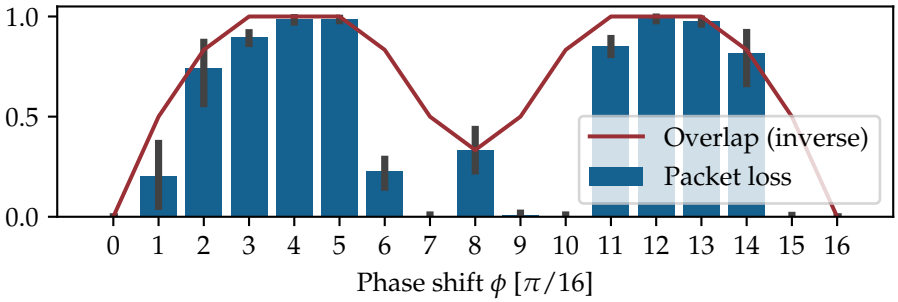


Figure 26: Packet loss for different phase shifts.

7.1.3 *Experimental Evaluation*

We evaluate the impact of our desynchronization attack by measuring the packet loss via the `ping` program. In particular, we use an APU board [149] equipped with a Qualcomm Atheros AR928X Wi-Fi card to act as an attacker that runs a modified version of *OWL* (Appendix C.2) to inject AWDL AFs. The ICMP echo requests are sent from a MacBook Pro (Late 2015, macOS 10.13) to an iPhone 8 (iOS 12). The attacker induces different phase shifts spanning one period. The sender emits 100 ICMP echo requests per experiment, which we repeat ten times and plot the resulting packet loss in Figure 26. The error bars indicate the standard deviation. In addition, we include the inverse channel overlap (Equation (8)) which is calculated for two identical sequences which we have observed were the most common ones during our experiment, i. e., $s_1 = s_2 = 44, 44, 44, 0, 0, 0, 0, 0, 6, 44, 44, 0, 0, 0, 0, 0$.

We modify our AWDL implementation to carry out the desynchronization attack.

At $\phi = 0$, there is no attack, while at $\phi = 16$, the targets are desynchronized by a full period. Due to the periodicity of the channel sequence, neither impairs communication reliability. The other results indicate that the desynchronization attack significantly degrades communication between the targets, peaking at phase shifts where the targets are off by a quarter ($\phi = 4$) and three-quarter period ($\phi = 12$), which is where the channel overlap has its minima (and the inverse overlap its maxima). With these settings, the packet loss is almost 100 %. Surprisingly, some phase shifts (e. g., $\phi = 6, 7, 9, 10, 15$) result in less packet loss than the overlap predicts. We suspect the reason to be retransmissions on the MAC layer (up to 7 times in Wi-Fi [95]) that, at the cost of longer latency, increase the chance that a frame will be received in a subsequent EAW.

A phase offset of a quarter or three-quarter period maximizes packet loss.

7.1.4 *Mitigation*

Devices can mitigate our desynchronization attack by discarding unicast AFs. Not accepting unicast frames is an extremely effective and practical countermeasure because it will cause all nodes in range to process the same information exclusively. While this does not pre-

Ignoring unicast AFs is an effective counter measure.

vent an attacker from winning the master election and, thus, sending invalid synchronization parameters, as all nodes process the same frames, it becomes much harder to create a deterministic offset between two targets. A more sophisticated attacker could employ attacks on the PHY layer (e. g., using directional antennæ) to achieve a similar effect as that of unicasting. However, such attacks are much more difficult to carry out in practice. In reaction to our report, Apple issued updates for all of its OSs to prevent desynchronization (Appendix B.4)

7.1.5 *Comparison to Reactive Jamming*

Desynchronization allows the attacker to intercept frames from its targets which is much harder for a jammer.

At first glance, our desynchronization attack achieves a similar effect as a *reactive jammer* [71, 121, 162]. However, the desynchronization attack can be more attractive for two reasons. First, desynchronization needs less energy than a jamming attack in principle. The desynchronization attacker only needs to emit one frame every 1.5 s to maintain their position as a master node because AWDL nodes elect a new master if they have not received an AF for more than 1.5 s from their current one. In contrast, a reactive jammer needs to emit a jamming signal for every packet that the target sends. Second, it allows intercepting frames from its targets, which enables mounting more sophisticated MitM attacks, as presented in Section 7.2. In contrast, a normal jammer *kills* the frame in transit, disallowing anyone (even the attacker themselves) to decode the frame [162]. There exist more sophisticated receiver designs that cancel out the jammer's signal, but this typically requires special hardware [71]. Our desynchronization attack only requires a system with an off-the-shelf Wi-Fi chip and, thus, could even be implemented in a smartphone [163].

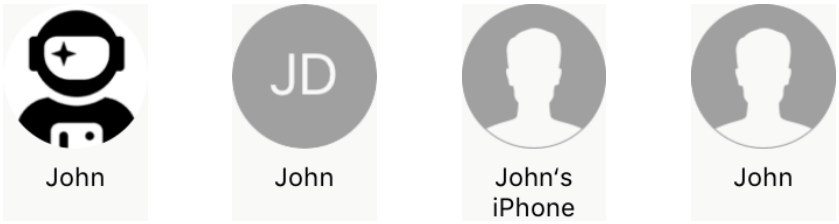
7.2 DOS-SUPPORTED MACHINE-IN-THE-MIDDLE ATTACK ON AIRDROP

DoS is the first step in mounting a MitM attack.

This section describes a MitM attack on the popular AirDrop service, which allows iOS and macOS devices to exchange files directly via AWDL. First, we assess the security of AirDrop and find that poor UI design choices enable an attacker to masquerade as a valid receiver. Then, we describe a complete MitM attack on AirDrop that prevents any sender from discovering a valid receiver using a DoS attack and subsequently can intercept and modify any AirDrop file transmission. Finally, we discuss possible mitigations for the attack.

7.2.1 *Ambiguous Receiver Authentication State*

We have observed that AirDrop employs two different kinds of connections that we term *authenticated* and *unauthenticated* (see Section 6.3). Further, the user can set its device to be discoverable by *contacts only*



(a) Sender has John's contact photo

(b) Sender does not have John's contact photo

(c) Sender does not have John as a contact

(d) Attacker spoofs John's identity

Figure 27: UI representation of an AirDrop receiver.

or *everyone*. Counter-intuitively, the discoverability setting *only applies to the receiver side.* In particular, while a receiver in *contacts-only* mode will only accept files from authenticated senders, a sender will see all discoverable receivers irrespective of whether they are authentic or not. This ambiguity has profound implications for security because it is up to the user of the sending device to decide whether a connection is authenticated or not, which can be non-trivial. The only visual cue to differentiate between an authenticated and unauthenticated connection is that an authenticated connection will show the receiver's name and photo from the sender's address book. Neither provides sufficient evidence to decide whether a receiver is authentic unambiguously. First, if no contact photo is available (users augment only 27 % of their contacts with a photo according to our survey in Appendix A), the icon contains the receiver's initials in a grey circle which is similar to that of an unauthenticated receiver (a grey circle with a head's silhouette). Second, the name that is displayed underneath unauthenticated receivers is the receiver's device name. Based on our results in Appendix A, a device name contains the user's given name in the majority of the cases (more than 70 % according to our experimental evaluation), which the attacker can exploit to create a trustworthy-looking device name. Unless users are sensitive to such subtle UI changes, an attacker can easily trick them into sending files via an unauthenticated connection. Figure 27 compares the different receiver icons, including a spoofed identity by the attacker. We want to highlight the similarity between an authenticated identity (Figure 27b) and a spoofed identity (Figure 27d).

The AirDrop UI does not clearly distinguish between authenticated and unauthenticated receivers.

The attacker controls the display name.

7.2.2 *Protocol Flow under Attack*

Our MitM attack on AirDrop is carried out in three phases. First, we break the discovery process to put ourselves in a privileged position. Second, we wait until the target receiver becomes discoverable by *everyone*, effectively forcing the user to downgrade the connection. Third, we relay and manipulate the actual data transfer to plant arbitrary

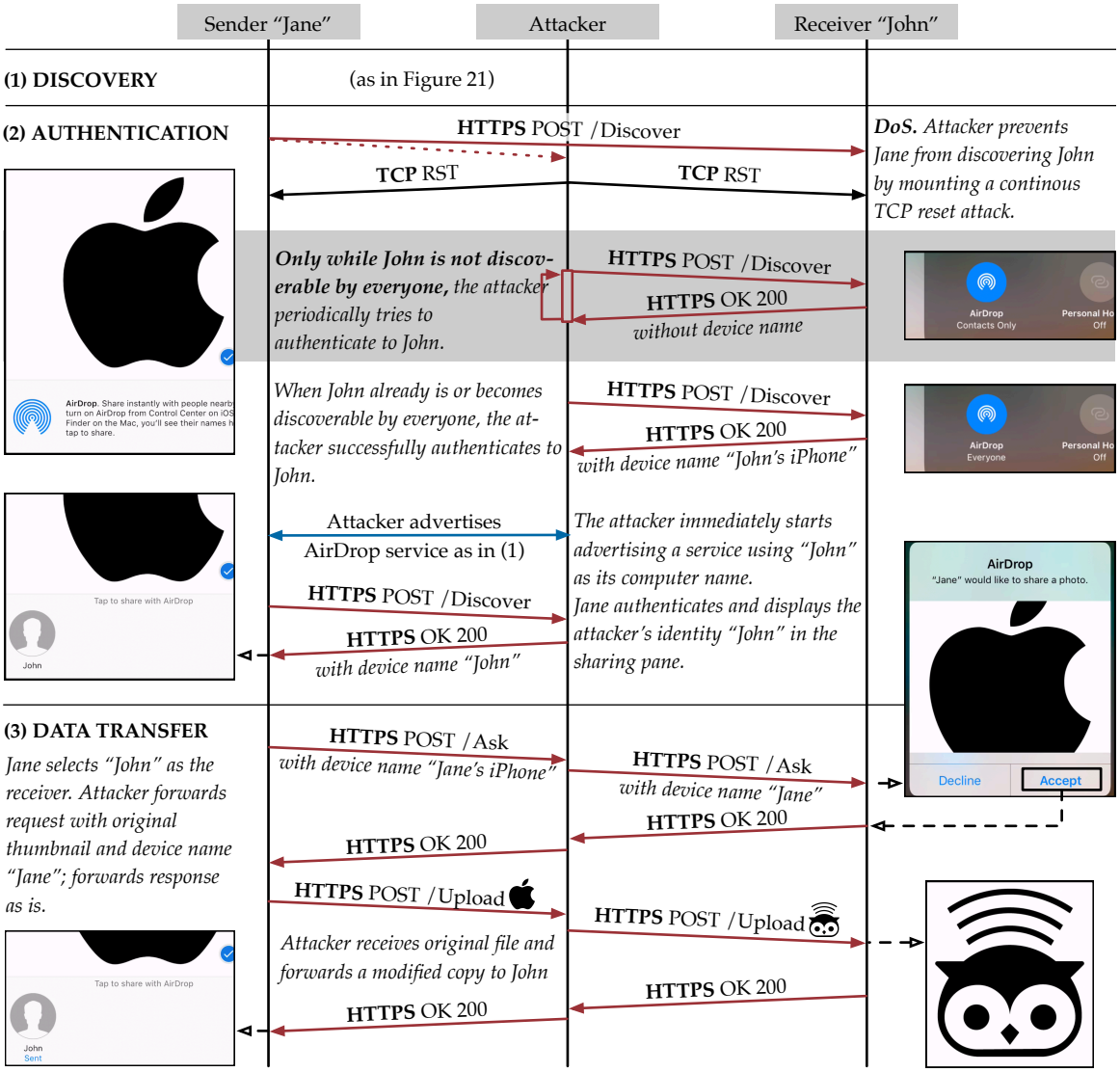


Figure 28: Protocol flow and user interaction of our MitM attack on AirDrop.

files at the receiver. We illustrate the attack in Figure 28 and explain each phase in more detail below.

The DoS attack disturbs AirDrop's discovery handshake.

(1) BREAKING DISCOVERY VIA DOS The most crucial part of the attack is preventing the sender from discovering the receiver such that it appears as an icon in the sharing pane. In particular, we need to prevent that the discovery handshake via HTTPS completes successfully. In principle, such a DoS attack could be carried out via our desynchronization attack (Section 7.1). However, we found that it could not reliably prevent the short HTTPS *discovery* requests and responses from being received. This is because AirDrop senders increase the channel allocation when starting the discovery process, thus, increasing the overlap with the receiver even when desynchronized. As an alternative, we used the well-known TCP reset attack, which sends TCP segments with an RST flag to the targets that, in turn, immediately drops the connection. For this attack, the attacker sends out an RST reply for every TCP segment that is not addressed to itself and effectively prevents any reconnection attempts from the sender to the receiver.

The DoS attack forces the receiver to set their device in everyone *mode.*

(2) DOWNGRADING AN AUTHENTICATED CONNECTION For a complete MitM attack, we need to authenticate with the receiver. Otherwise, it will deny any *ask* or *upload* requests. If the receiver is discoverable by *everyone*, this is trivial, since it accepts all authentication attempts, even those with a self-signed certificate which the attacker can easily generate (see Section 6.3). The receiver indicates a successful authentication attempt from a non-contact by including its device name in the discover response. However, we have found that in most cases (59.4 % in our survey), users set their device to *contacts only*. In such cases, we leverage the ongoing DoS attack to force the receiver to try the *everyone* setting.

The remainder of the MitM attack is straightforward.

(3) RELAYING AND MODIFYING DATA TRANSFER We can check when a receiver is discoverable by everyone by periodically sending *discovery* requests. Once the receiver becomes discoverable, we advertise our own AirDrop identity via mDNS and wait until the sender tries to perform the authentication handshake via HTTPS for discovery which we let succeed. We relay the sender's *ask* request to the receiver, including the original file thumbnail, to make the request appear valid. After the receiver accepts the transmission request, we relay the answer back to the sender, which—in turn—starts to send the actual file. We can now decide whether to relay a modified version of the file or send an entirely new one, possibly containing malware to the receiver.

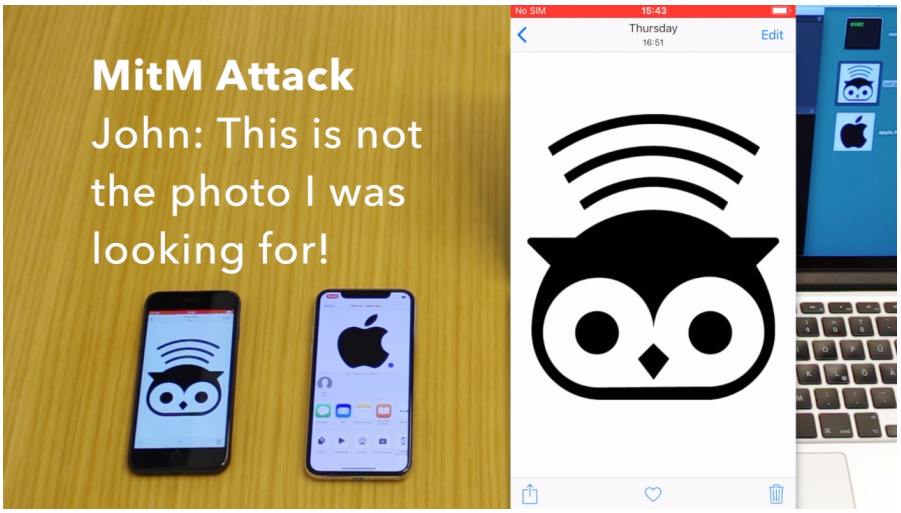


Figure 29: PoC of MitM attack on AirDrop (still image of video [170]).

7.2.3 *Proof-of-Concept*

We provide a video PoC of the complete MitM attack on AirDrop.

Our proof of concept (PoC) of the MitM attack consists of two components. First, we modify OWL (Appendix C.2) to overhear data frames not addressed to us and set an arbitrary source MAC address, which is required to mount a TCP reset attack. Second, we modify OpenDrop (Appendix C.3) to probe the discoverability status of the receiver target and finally implement the MitM attack as depicted in Figure 28. Using these implementations, we record a video of the attack [170]. A still image is shown in Figure 29.

7.2.4 *Mitigation*

We discuss possible mitigation strategies. We examine them according to implementation complexity, starting with the mitigation requiring the least number of changes to existing AirDrop implementations.

Apple has implemented stronger visual cues to assist users in differentiating between contacts and non-contacts.

PROVIDE STRONGER VISUAL CUES One of the core problems of the current design of AirDrop is that a user might have a hard time to differentiate between authenticated and unauthenticated receivers (see Section 7.2.1 and Figure 27). Currently, the only cues to decide whether a receiver is authenticated are the display of a contact photo and contact name. We have shown that the former is not commonly available (users augment 27.4 % of their contacts with photos), and the latter can be spoofed. Therefore, we propose to provide stronger visual cues that unambiguously tell the user if a receiver is authenticated or not. This is already customary in web browsers where HTTPS-protected websites are augmented with a green (lock) symbol telling the user that the website they are visiting is authentic. Based on our report, Apple has reworked the sharing pane UI in iOS 13 that

now groups authenticated and unauthenticated receivers, respectively (Appendix B.8).

RESET DISCOVERABILITY SETTING AFTER A TIMEOUT Users might set the discoverability of their device to *everyone* for convenience or if they used it for one occasion and then forgot to reset it. In either case, we believe that the *everyone* setting should only be used when it is required, i. e., if one wants to receive a file from a non-contact. To protect negligent users, we propose to use a timeout, after which the discoverability setting is reset to *contacts only*. Alternatively, one could reset the setting the next time the device is locked. This would also prevent past cases where people would receive inappropriate photographs from strangers in public places [26, 80], because, in *contacts-only* mode, devices will transparently reject all requests from unauthenticated senders.

Resetting to contacts-only *mode automatically would prevent users from unsolicited requests.*

INTRODUCE SECURE AIRDROP MODE FOR NON-CONTACTS Our last proposal involves deprecating unauthenticated connections and instead establish authentication with a non-contact via an out-of-band (OOB) channel. AirDrop could transmit one-time cookies with similar functionality as the *validation record* (see Section 6.3) during the initial HTTPS authentication handshake (see Section 6.2). The one-time cookies could be validated via an OOB channel such as NFC or via QR codes. After one transfer (or after a specific timeout), each device deletes its one-time cookie. By committing to the one-time cookie in the TLS handshake, a MitM attack on the OOB channel would be fruitless because the attacker could not establish a TLS connection with the same key. Unfortunately, such a mode would require one more manual step by both parties and, therefore, would impair usability.

Abandoning unauthenticated connections would greatly improve security but sacrifice usability.

7.2.5 *Previous Attacks on AirDrop*

Other attacks on AirDrop have been presented before. An impersonation attack [20] exploits DNS service discovery (DNS-SD) via multicast DNS (mDNS) to redirect file transmissions to an attacker for unauthenticated connections. In particular, the attack uses forged SRV and AAAA responses to redirect an AirDrop ID to the attacker. This attack only affects unauthenticated connections, while our attack also targets authenticated connections via a downgrade attack and we present a complete MitM attack that allows an attacker to send malicious files to the receiver stealthily. Finally, [20] proposes a conflict detection mechanism for mDNS to prevent their attack, which is based on the assumption that "disrupting two parties' communication through a Wi-Fi direct link or a local network is difficult for the adversary without access to the routing infrastructure of the network." In this work, we show that it is indeed practical to mount a DoS on the

Previous attacks on AirDrop targeted DNS-SD.

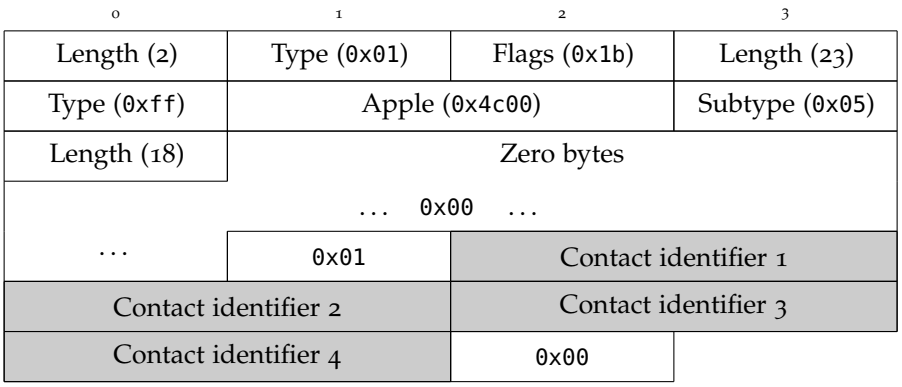


Figure 30: AirDrop BLE advertisement frame format showing semantics and values of individual bytes highlighting the hashed contact identifiers.

link layer since AWDL does not employ any security mechanisms. An earlier work [55] targeted a vulnerability in AirDrop's implementation that allowed the attacker to install files in arbitrary directories on the target's system. Apple fixed this bug in 2015.

7.3 DOS BLACKOUT ATTACKS ON AWDL

We need to activate nearby AWDL devices to make the attack practical.

We discovered two implementation bugs that allowed use to crash Apple devices by sending corrupted AWDL AFs. To demonstrate the seriousness of these vulnerabilities in practice, we require the targets' AWDL interface to be active, which is typically not the case since an application has to request activation explicitly (Section 5.2.1). Therefore, we exploit the Bluetooth Low Energy (BLE) discovery mechanism integrated with AirDrop (see Chapter 6) to activate all AWDL devices in proximity. In particular, devices in *everyone* mode will enable AWDL immediately after receiving any AirDrop BLE advertisement. To also target devices in the default *contacts-only* mode, we analyze the practicality of brute-forcing the truncated contact hashes in AirDrop's BLE advertisements. Then, we build a PoC leveraging a low-cost (20 US$) BBC micro:bit device and experimentally confirm that the attack is feasible in practice with a target response time of about one second for devices that have 100 contact identifiers in their address book. Finally, we introduce the actual vulnerabilities and discuss their mitigation.

7.3.1 *AirDrop BLE Advertisements*

Limited space in the BLE advertisements required Apple to truncate the contact identifier hashes.

We show the actual BLE advertisement frames [32, Vol. 3, Pt. C, Sec. 11] of AirDrop, including four contact identifier hashes in Figure 30. The advertisements are broadcast as non-connectable undirected advertising (`ADV_NONCONN_IND`). The frames use manufacturer-specific data fields that have fixed values except for the contact hashes. In fact, we found that the contact hashes are the first two bytes of the SHA-256

SYMBOL	DESCRIPTION
$\mathbb{S}$	Contact hash search space
$\mathbb{C}$	Contacts in the target's address book
w	Target's BLE scan window
$\mathfrak{i}$	Target's BLE scan interval
$\mathfrak{i}_{PHY}$	Attacker's BLE PHY injection interval
$\mathfrak{r}$	Effective contact hash brute force rate
$\mathfrak{n}$	Tried hash values per scan window
$\mathfrak{p}$ ($\mathfrak{p}_j$)	Hit probability after one (or j) scans

Table 9: Symbols used for the BLE brute force attack.

digest of the sender's contact identifiers that are also included in the validation record (see Section 6.3). If the sender has less than four identifiers, the remaining contact hash fields are set to zero. Due to the short length, it appears feasible to use brute force to try all possible values.[1]

7.3.2 *Brute Force Analysis*

We first determine the feasibility of brute-forcing analytically.

We assume that the attacker does not know the target's contacts and, thus, attempts to enable the target's AWDL interface using brute force. As the target has at least one contact identifier (the address book contains at least the user's own Apple ID), the attacker needs to try $\mathbb{S} = 2^{16} = 65\,536$ hashes in the worst case. Thus, the challenge for the attacker is to quickly send a large number of BLE advertisements *while* the target is conducting a BLE scan. In the following, we analyze *how fast* the attacker can deplete the search space and *how successful* they would be. We start investigating the results for a single BLE scan window and then extend our analysis to multiple scan intervals. We use the symbols described in Table 9 in the following.

ONE SCAN WINDOW Let the attacker inject BLE advertisement frames at the physical layer with an interval of $\mathfrak{i}_{PHY}$. Further, consider that the attacker has a single radio and that BLE uses three advertisement channels [32]. Also, recall that an AirDrop BLE frame has room for four contact hashes. Then, we calculate the attacker's effective brute force rate $\mathfrak{r}$ as:

$$\mathfrak{r} = \frac{4}{3 \cdot \mathfrak{i}_{PHY}} \,. \tag{12}$$

1 Note that the sender still has to provide the complete hash during the HTTPS handshake before the receiver accepts the data on an *authenticated* connection.

Now, we can compute the number of hash values n that the attacker can inject per scan window w [32] as:

$$n = w \cdot r \,. \tag{13}$$

The attacker needs to send advertisements while the target is scanning.

Let X be a random variable, and let $P(X = k)$ denote the probability that the target "sees" k out of C known contact hashes during one scan window. Since the attacker moves through the search space sequentially, we can formulate the problem using the *urn model* in *drawing without replacement* mode which results in a hypergeometric distribution. We get:

$$P(X = k) = \frac{\binom{n}{k}\binom{S-n}{C-k}}{\binom{S}{C}} \,. \tag{14}$$

In particular, the attacker only requires one hit to activate the target's AWDL interface whose probability we call p:

$$\begin{aligned} p &= P(X \geqslant 1) = 1 - P(X = 0) \\ &= 1 - \frac{\binom{n}{0}\binom{S-n}{C-0}}{\binom{S}{C}} = 1 - \frac{\binom{S-n}{C}}{\binom{S}{C}} \,. \end{aligned} \tag{15}$$

Using the Stirling's approximation $\binom{n}{k} \approx \frac{n^k}{k!}$ for $k \ll n$, we can simplify Equation (15) as:

$$\begin{aligned} p &\approx 1 - \frac{\frac{(S-n)^C}{C!}}{\frac{S^C}{C!}} = 1 - \frac{(S-n)^C}{S^C} \\ &= 1 - \left(\frac{S-n}{S}\right)^C = 1 - \left(1 - \frac{n}{S}\right)^C \,. \end{aligned} \tag{16}$$

We assume that the attack does not know when the target's scan window starts.

MULTIPLE SCAN INTERVALS BLE devices perform scans regularly at a fixed interval i [32]. Let Y be a random variable indicating that the attacker has at least one hit ($Y = 1$) or none ($Y = 0$) during one scan. We assume that the attacker does not know when the target's scan window starts and, therefore, that Y is independent and identically distributed between scans.[2] Let j indicate the target's jth scan since the attacker started their brute force attack. Then, the probability that the attacker had k hits after j scans is given by a binomial distribution:

$$P(Y = k) = \binom{j}{k} p^k (1-p)^{j-k} \,. \tag{17}$$

Again, the attacker needs at least one hit whose probability we denote as p_j (note that $p_1 = p$):

$$p_j = P(Y \geqslant 1) = 1 - P(Y = 0) = 1 - (1-p)^j \,. \tag{18}$$

2 If the attacker knew the start of each scan window, they could follow a better strategy by only sending advertisements while the target is performing a scan. This way, they would deterministically succeed after they had gone through S once.

```c
uint8_t *le_adv = airdrop_init_template()
for (uint16_t h = 0; /* loop */; h += 4)
{
  airdrop_set_hashes(le_adv, h, h+1, h+2, h+3);
  for (uint16_t chan = 37; chan < 40; chan++)
  {
    le_adv_tx(le_adv, chan);
    sleep(0.625 /* in milliseconds */);
  }
}
```

Program 1: C pseudo code of our BLE brute force attack.

With Equation (16), we get:

$$p_j \approx 1 - \left(1 - \frac{n}{S}\right)^{jC} . \tag{19}$$

We know that j depends on the time since the attack started and the target's BLE scan interval i (the target performs one BLE scan of length w per interval). Let t denote the attack duration, then $j \leqslant \lfloor t/i \rfloor$. Finally, we denote the success probability at time t as

$$p(t) \approx 1 - \left(1 - \frac{wr}{S}\right)^{tC/i} . \tag{20}$$

7.3.3 *Jailbreaking BLE Advertisements*

The send interval of BLE advertisements in off-the-shelf devices is restricted.

The Bluetooth standard imposes a minimum advertisement interval[3] of 100 ms for non-connectable undirected advertising [32, Vol. 6, Pt. B, Sec. 4.4.2.2], which we found is usually enforced in the BLE firmware. By complying with the standard, the attacker would need at least $2^{16} = 27$ minutes to iterate through the entire search space once. If the attacker had access to the BLE physical layer to control and schedule individual transmissions, they could circumvent the standard's restrictions and, thus, iterate through the search space much faster. To this end, we extend an open-source BLE firmware [34] for the Nordic nRF51822 [142] chipset to implement our brute force attack. In principle, our attack implementation is very simple and shown in Program 1. We use a send interval of $i_{PHY} = 0.625\,\text{ms}$ resulting in $r = 2133.3\,\text{s}^{-1}$ which allows the attack to iterate through S in only $2^{16}/f = 30.72\,\text{s}$. By using three BLE radios (one for each advertisement channel), we could reduce this time to 10.24 s. However, we show that using one radio is sufficient in practice.

3 The BLE advertisement interval accounts for a frame transmission on each of the three advertisement channels.

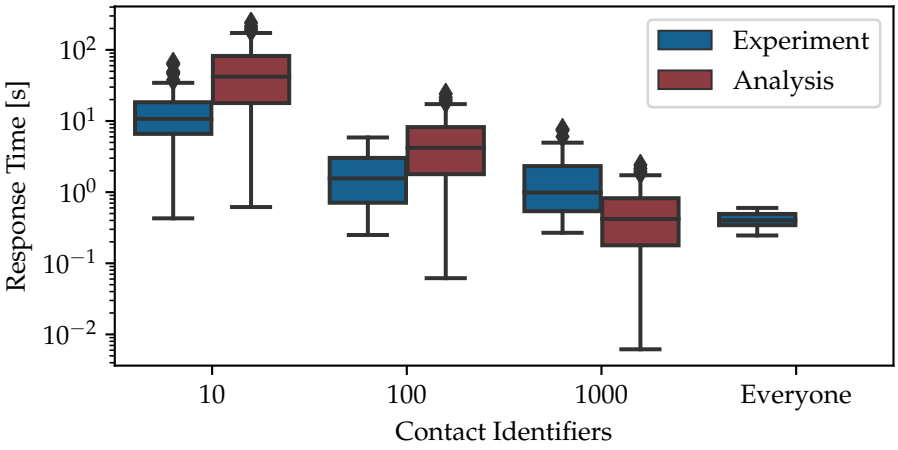


Figure 31: Time until target activates AWDL after being exposed to our brute force attack.

7.3.4 *Target Response Time*

We measure the time until our brute force attack is successful.

We measure the target response time, i. e., the time it takes for a target to turn on its AWDL interface when being exposed to our attack. In particular, we measure the response time for a *contacts-only* receiver that has 10, 100, and 1000 contact identifiers in their address book. Also, we include reference measurements for a receiver in *everyone* mode under the same attack.

Hardware as cheap as 20 US$ is sufficient to implement the attack.

SETUP For the experiment, we use a Wi-Fi sniffer (Broadcom BCM-4360) to receive AWDL AFs and a 20 US$ micro:bit device [132] to inject BLE advertisements. To get the response times, we start a brute force attack and measure the time until we receive the first AF from the target. We then stop the attack and wait until the target stops sending AFs which means that the AWDL interface has turned off. Then, we start over to collect 50 measurements per setting.

In a realistic setting, our brute force attack activates nearby AWDL devices in about one second.

RESULTS We show the results for an iPhone 8 (iOS 12) in Figure 31. The plot also includes the analytical response time distribution based on Equation (20), assuming a BLE scan window w and interval i of 30 ms and 300 ms, respectively.[4] We can make several observations. (1) Our analytical model does not capture our experimental results precisely but approximates them within an order of magnitude which is sufficient for our purposes. (2) The median response time of targets with only 10 contact identifiers in their address book is 10 seconds and decreases to about 1 second when more contacts are available. We found that a user has more than 136 contacts on average based on a user study that we describe in Appendix A.2. (3) This means that the

4 https://lists.apple.com/archives/bluetooth-dev/2014/Sep/msg00001.html

Figure 32: PoC of blackout attack on iOS devices (still image of video [169]).

brute force attack is feasible for scenarios where the target will be in the attacker's communication range for a few seconds.

7.3.5 *Crashing AWDL Devices in Proximity*

During our AWDL analysis and building an AWDL prototype, we found two implementation flaws in Apple's OSs that allow an attacker to crash devices in proximity.

Implementation flaws in AWDL enable us to crash nearby devices.

These flaws can be exploited by sending corrupt AFs. In particular, we can trigger kernel panics by setting invalid values in the synchronization parameters (affecting macOS 10.12) and the channel sequence (affecting macOS 10.14, iOS 12, watchOS 12, and tvOS 5), respectively. To showcase our findings, we provide a video of our PoC which exploits the second vulnerability on iOS devices [169] with a still image shown in Figure 32. The video demonstrates how an attacker mounts a targeted DoS attack that crashes a single device and a blackout DoS attack that crashes all devices in range of the attacker at the same time.

While not critical by themselves, the mere existence of these vulnerabilities creates a new class of threats to Wi-Fi devices as an attacker can exploit them *without any authentication towards the target*, i. e., they do not have to be on the same network. In light of the complexity that AWDL adds to existing Wi-Fi implementations and past discovered issues in standardized Wi-Fi procedures [15, 28], we think that a determined attacker can find vulnerabilities in AWDL that eventually lead to remote code execution.

7.3.6 *Mitigation*

Apple has responded to our responsible disclosure and released a two-step mitigation to the problem. First, Apple implemented a rate-

limiting mechanism (Appendix B.7) that throttles the contact hash matching process and, thus, makes our brute force attack less effective to activate nearby AWDL devices. Second, Apple has fixed a parsing problem we exploited in our PoC and affected all Apple OSs (Appendix B.1). Another reported vulnerability affecting only macOS 10.12 (Appendix B.2) remains unpatched.

Apple did not patch all vulnerabilities.

7.4 DISCUSSION AND SUMMARY

The deployment of open Wi-Fi interfaces enables new types of applications for mobile devices. They allow devices in proximity to communicate with each other without being connected to the same Wi-Fi network. On the downside, this also opens new opportunities for an attacker as they no longer have to provide any kind of authentication (e. g., access to a secure Wi-Fi network). In this chapter, we investigate the first protocol of this kind, i. e., Apple's proprietary AWDL. In particular, we find three distinct protocol-level vulnerabilities that allow for DoS, user tracking, and MitM attacks. In addition, we discovered two implementation bugs in Apple's OSs that cause DoS. Given the complexity of the protocol and implementations, we conjecture that more severe vulnerabilities will be found in the future. We have shared our findings with Apple, and they closed the vulnerabilities in several OS updates (see Appendix B for details). Finally, our findings have implications for the non-Apple world: Neighbor Awareness Networking (NAN), commonly known as Wi-Fi Aware, is a new standard supported by Android which draws on AWDL's design and, thus, might be vulnerable to the similar attacks as presented in this work. This is pending further investigation.

Deploying open wireless interfaces exposes users to new risks.

Part III

ISLAND AND ARCHIPELAGO COMMUNICATION

8

DOS-RESILIENT ISLAND COMMUNICATION

Internet of things (IoT) devices penetrate different aspects of our life, including critical services such as health monitoring, emergency communciation, and autonomous driving. Such safety-critical IoT systems often consist of a large number of devices and need to withstand a vast range of known denial-of-service (DoS) attacks to ensure reliable operation while offering low-latency information dissemination. As the first solution to jointly achieve these goals, we propose LIDOR, a secure and lightweight multihop communication protocol designed to withstand all known variants of packet dropping attacks. Specifically, LIDOR relies on an end-to-end feedback mechanism to detect and react on unreliable paths and draws solely on efficient symmetric-key cryptographic mechanisms to protect packets in transit. We show the overhead of LIDOR analytically and provide the proof-of-convergence for LIDOR that makes LIDOR resilient even to strong and hard-to-detect wormhole-supported greyhole attacks. In addition, we evaluate the performance via testbed experiments. The results indicate that LIDOR improves reliability under DoS attacks by up to 91 % and reduces network overhead by 32 % compared to the state-of-the-art Castor protocol [64] that is our benchmark.

LIDOR provides comprehensive security guarantees while maintaining low communication latency.

8.1 OVERVIEW

In this section, we first provide the system model and a high-level overview of the LIDOR protocol. Next, we describe the protocol in detail, and, finally, highlight differences from the benchmark protocol.

8.1.1 *System Model*

Our security assumptions consist of a trust model and an adversary model. We further require each node to perform certain basic cryptographic operations. We elaborate on them in the following.

TRUST MODEL LIDOR constitutes an end-to-end communication protocol. Hence source and destination nodes need to trust each other and be able to share a cryptographic key. The key establishment can be mediated by a trusted third party that certifies public keys, via secure device pairing methods, or manually by the user. In the IoT scenario, the device manufacturer could pre-deploy certified keys such that devices from the same manufacturer could perform key derivation without an active third party. In an emergency scenario, we could use

We require established end-to-end trust relationships between nodes that want to communicate.

the certificate infrastructure proposed in Chapter 9. However, we do not assume trust relationships between relay nodes. Thereby, LIDOR is resilient to individual nodes that the adversary compromises, e. g., in case that certain device models expose vulnerabilities.

ADVERSARY MODEL We consider the DoS attacks presented in Section 2.2. We consider an insider adversary, i. e., an entity controlling a portion of authenticated nodes within the network. They can consequently take part in normal network operations and can mount, e. g., jamming, spoofing, and dropping attacks. We refrain from discussing flooding attacks in this chapter as Castor [64], our benchmark protocol, has a mitigation against this attack that LIDOR can use as well. However, the adversary cannot break cryptographic primitives, and we assume that there is at least one adversary-free path between any source-destination pair that wishes to communicate. Without this restriction, an adversary in a favorable position could partition the network and completely prevent communication. In order to remove this restriction, nodes could attempt to detect non-functioning flows and switch to an archipelago-scope protocol. This would, however, sacrifice latency, and we leave such a mechanism as an open research topic.

A mitigation for flooding attacks has been previously proposed and can be directly integrated in LIDOR.

NODE CAPABILITIES We require that each node (1) has access to a pseudo-random number generator $\mathtt{rng}$, (2) can compute a cryptographic hash function $\mathtt{hash}(\cdot)$, (3) has access to a stream cipher $\mathtt{prf}(K, n)$ which takes a key K and some nonce n as inputs, and (4) can compute authentication tags $\mathtt{tag}(K, \cdot)$ based on a shared key K. We discuss practical candidate functions in Section 8.5.2.

We need to find suitable candidate functions for some cryptographic primitives.

8.1.2 *Protocol Summary*

LIDOR leverages established concepts [64, 160, 161] to provide DoS-resilient communication. At its core, LIDOR (1) uses an acknowledgment-based feedback mechanism to rate the reliability of neighbors and effectively detect faulty links, (2) lets intermediate nodes individually decide whether to conduct path exploration (broadcast) or exploitation (unicast) to react to changes in reliability quickly, and (3) separates the state of different flows to prevent adversarial state pollution. With its generic design, LIDOR is agnostic to the *cause* of disruptions but will detect the *existence* of failures and react to them. Thereby, LIDOR *comprehensively* thwarts any type of dropping attack. Further, LIDOR solely relies on *lightweight* symmetric cryptographic primitives that can be efficiently calculated on computationally constrained nodes. In particular, we rely on a Merkle tree-based commitment scheme. All packet IDs are committed to when constructing the tree. Then, the destination reveals the secret only after receiving it

LIDOR builds on acknowledgments, node-local routing decisions, and isolated flow state.

We only use lightweight cryptographic primitives.

in the form of an acknowledgment. Since all intermediate nodes can verify the secret, they can be sure that the destination has received the packet if they receive the acknowledgment.

8.1.3 *Comparison to Castor's Design*

Before we describe our LIDOR protocol in detail, we discuss the distinct differences to Castor [64], which we will refer to as our benchmark protocol for the remainder of this chapter. In particular, LIDOR differs from the benchmark in the following key points.

Compared to our benchmark Castor, LIDOR reduces bandwidth overhead and protects against replay and wormhole-supported greyhole attacks.

- We employ an effective construction and in-network compression of the Merkle tree. Especially with the in-network compression, we can reduce the average packet overhead from a logarithmic to a *constant* factor (with respect to the tree size) in a static setting. We provide the proofs in the overhead analysis in Section 8.3.
- We provide proper protection against replay attacks. In particular, we keep a list of seen packet identifiers that survives the acknowledgment timeout in the form of a space-efficient bit vector, as described in Section 8.2.2.
- We design a reliability metric that will converge even in the presence of a strong wormhole-supported greyhole attack. We provide a proof for non-convergence of the benchmark and a proof of convergence for LIDOR in Section 8.4 for a static network topology.

8.2 PACKET PROCESSING

LIDOR nodes process packets in five steps.

Next, we elaborate on the protocol workflow of LIDOR. The main processing steps are (1) packet generation, (2) packet verification, (3) packet forwarding, (4) packet reception, and (5) acknowledgment handling. We depict the various stages in Figure 33 and summarize the symbol notations in Table 10.

8.2.1 *Packet Generation*

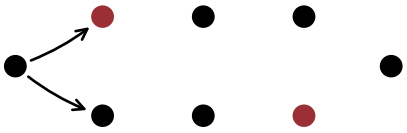
LIDOR establishes flows for end-to-end communication, similar to Castor [64]. The cryptographic material securing each flow is drawn from a Merkle tree, an accepted cryptographic tool for secure multi-hop communication [64, 88, 160, 161]. We first introduce the packet format and then discuss the peculiarities of Merkle tree usage and construction.

There are only two types of packets: PKTs and ACKs.

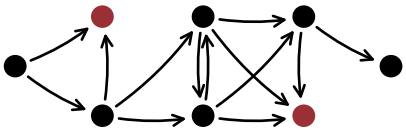
PACKET FORMAT The LIDOR packet (PKT) in Equation (21) contains source s and destination d identifiers, a flow identifier H, which is the root of a Merkle tree of height l, the kth PKT identifier b_k,

SYMBOL	NOTATION
s	source node
d	destination node
H	flow identifier (root of the Merkle tree)
l	height of Merkle tree
n	nonce
b_k	kth packet identifier
$K_{s,d}$	key
m	packet digest
σ	authentication tag
f_k	flow authenticator for the kth packet
a_k	ACK authenticator (preimage of b_k)
T_{ACK}	ACK timeout
$O_k^B(O_k^L)$	overhead for the kth packet for the benchmark (LIDOR) protocol
$O_B(O_L)$	total overhead from all the packets of the flow for benchmark (LIDOR) protocol
ΔO	difference between total overhead in benchmark and LIDOR protocol
η	number of hash values required
t_k	sending time of the kth packet
λ_n	number of packets for which nonce is re-transmitted
p_n	probability of retransmission of nonce
$\mu_{x,j}^H$	average reliability estimator for the jth neighbor of the node x for the flow H
$\mu_{x,j}^{a,H}(\mu_{H,j}^{f,H})$	reliability estimator for the jth neighbor of the node x for the flow H for all ACKs (first ACK)
$\alpha_{x,j}^{a,H}(\alpha_{x,j}^{f,H})$	counts all ACKs (first ACK) received successfully from the jth neighbor of the node x for the flow H
$\beta_{x,j}^{a,H}(\beta_{x,j}^{f,H})$	counts all ACKs (first ACK) not received from the jth neighbor of the node x for the flow H
δ	reliability estimator adaptivity parameter controlling the decay of α and β
ϵ	suitable threshold on reliability
B_x^M	successive broadcast of M packets by the node x
B^M	successive broadcast of M packets by all the nodes in the network
U_x^y	unicast of a packet from the node x to the node y
Δ_M	the reliability estimator of a node after M packets transmitted successfully
$M_{min}(M_{max})$	Minimum (maximum) number of packets required to achieve convergence
A	number of attacker nodes in a relay layer
N	number of non-attacker nodes in a relay layer
G_ν	minimum number of broadcasts required in terms of Γ and Υ
I	number of hops
ξ_I	number of packets unicast for I hops

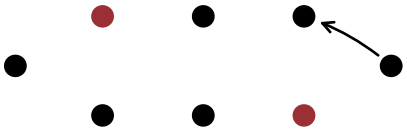
Table 10: Symbols and notations for LIDOR.



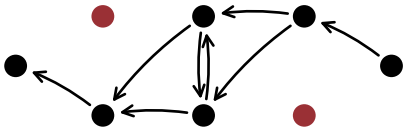
(a) The source (1) generates a packet and (3) forwards it to its most reliable neighbor. If reliability is low, it probabilistically broadcasts to all neighbors to explore new paths.



(b) Receiving nodes (2) discard duplicates and verify that the packet belongs to a certain flow. Then, they (3) make forwarding decisions the same way the source node does.



(c) The destination (4) verifies the authenticity of the received packet and replies with an acknowledgment on the reverse path.



(d) All receiving nodes (5) verify its authenticity and update the reliability rating of their respective neighbors. Neighbors that do not return an acknowledgment receive a penalty.

Figure 33: Overview of LIDOR's protocol workflow showing which operations are made in which stage: (1) packet generation, (2) packet verification, (3) packet forwarding, (4) packet reception, and (5) acknowledgment handling. Attacker nodes are marked in red and drop all packets in this example.

and an authentication tag σ. A nonce n is included until the first acknowledgment (ACK) of the flow is received. The user payload $\mathcal{P}$ may be encrypted using the stream cipher $prf(K_{sd}, hash(n||k))$. σ is computed over all fields except the flow authenticator f_k and length l'. Additional metadata (packet type, length of hash values, and length of the entire packet) is excluded for brevity.

$$PKT = \left(s, d, H, b_k, f_k^{l'<l}, n, \mathcal{P}, \sigma\right) \tag{21}$$

PURPOSE OF MERKLE TREE LIDOR utilizes Merkle hash trees for packet labeling, flow authentication, and proof of packet reception. In particular, the idea is to use the input values for the tree's leaf nodes as packet identifiers b_k and to commit to them with the root H that is used as a flow identifier. The packet identifiers, in turn, are computed from a secret a_k as $b_k = hash(a_k)$. Since PKTs are end-to-end authenticated, the destination node will only reveal the preimage a_k of the packet identifier b_k for authentic packets in the form of an ACK (Section 8.2.4). Upon reception of a_k, intermediate nodes can deduce that the destination must have received an authentic packet with b_k.

The Merkle tree is a key building block of LIDOR's design.

NONCE-SEEDED MERKLE TREE CONSTRUCTION We construct the LIDOR Merkle tree as follows. (1) We use a unique and random nonce

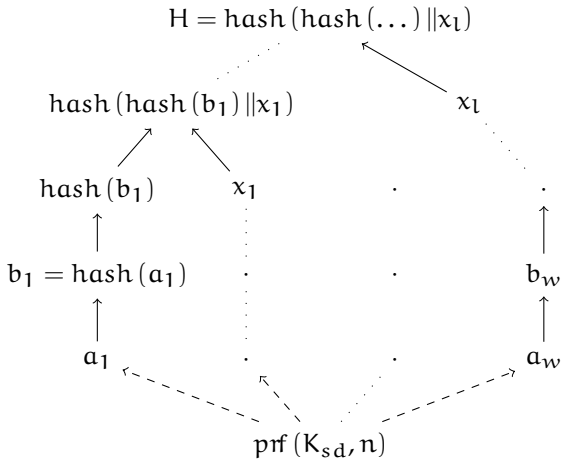


Figure 34: LIDOR Merkle tree generation. The leaf seeds a_k are drawn from a stream cipher $prf(\cdot,\cdot)$ that, in turn, is seeded by a secret key K_{sd} and a public nonce n. The intermediate tree nodes $x_1, \ldots, x_l$ are the elements of the flow authenticator f_k.

n and use it together with K_{sd} to seed a cryptographically secure pseudo-random number generator $prf(\cdot,\cdot)$, e. g., a stream cipher. (2) We "chop" the output of $prf(K_{sd}, n)$ into w blocks[1] of size $|hash(\cdot)|$ to create a_k for $k = 1, \ldots, w$ and construct the Merkle tree as shown in Figure 34. Our unique approach of seeding the Merkle tree with a nonce n enables the source to share all secret values a_k with the destination by just communicating n. Together with the shared key K_{sd}, the destination can repeat the Merkle tree construction process and retrieve all a_k. Without (1), the source would need to communicate all a_k individually in a confidential manner, which would waste bandwidth as done in [64].

The Merkle tree is seeded by a shared key and a nonce.

When creating the Merkle tree from n and K_{sd}, we need to assert that we never reuse n for any source-destination pair. Otherwise, replay attacks are possible. Reasonable candidates for n are timestamps or randomly chosen values drawn, e. g., from a system-provided rng function. The drawback of choosing timestamps as nonces is the additional attack vector on time synchronization services such as NTP [125] or GPS [147]. When choosing n purely at random, n must be large enough to avoid nonce reuse due to the well-known *birthday attack* [69]. We choose to implement the second option with a random 192-bit nonce.

Nonces need sufficient entropy to mitigate the birthday attack.

In the optimal case, the tree size w is chosen such that it is equal to the number of packets a source node wishes to transmit for a certain flow. If this number is known a priori, LIDOR can use w as an optimization. In all other cases, LIDOR has to rely on a default tree size. However, choosing the default tree size incurs a trade-off. (1) The length of the flow authenticator included in every packet grows

The choice of the tree size has a profound impact on efficiency.

1 Note that due to the nature of a binary tree, w must be a power of 2.

logarithmically with the tree size. However, (2) very small trees cause frequent flow restarts, i. e., whenever all b_k have been used, a new tree must be created, and the route exploration process restarts. In Section 8.2.3, we propose a countermeasure for (1) in the form of an in-network compression mechanism that can reduce the average overhead of the flow authenticator to a constant factor.

8.2.2 *Packet Verification*

During packet verification, a node filters out the PKTs that either have already been forwarded (i. e., duplicates) or contain an invalid flow authenticator. Also, it computes the minimal authenticator length for the next-hop node.

DUPLICATE DETECTION Duplicate detection consists of two steps. First, a node calculates a packet digest m using a collision-resistant hash function of the incoming PKT (excluding the variable-length field f_k). This serves to identify unique PKT copies that might have the same packet identifier b_k. If the node has already seen the pair (m, b_k), the PKT is dropped. For preventing replay attacks, each node keeps a per-flow state to memorize which PKTs have already been acknowledged for preventing replay attacks. A very low-complexity and space-efficient implementation of such a data structure is a zero-initialized bit vector. Setting bit k in the bit vector signifies that the kth PKT of a certain flow is acknowledged and, thus, future copies of b_k can be ignored and are not forwarded again. Specifically, a node checks whether PKT k of the indicated flow has already been acknowledged (kth bit set) and, if yes, discards it. We demonstrate the effectiveness of our replay protection mechanism in Section 8.6.

Detecting duplicates is important for both efficiency and security.

FLOW AUTHENTICATION The Merkle tree assures that all b_k can be authenticated to a single value, i. e., the root H that serves as a flow identifier. Intermediate nodes can validate that b_k belongs to H by traversing the tree from b_k to the root H' using intermediate tree nodes f_k and checking that $H' = H$. The flow authentication procedure has been described in [64] and assures that only PKTs belonging to the flow will be forwarded.

A relay node makes sure that a received PKT belongs to the indicated flow before forwarding it.

8.2.3 *Packet Forwarding*

We describe the forwarding decision that is based on a reliability metric. We further discuss the in-network Merkle tree compression to reduce network overhead and explain the purpose of the PKT timer.

RELIABILITY METRIC In contrast to the per-destination routing state used in classic MANET protocols [43, 150], LIDOR keeps the

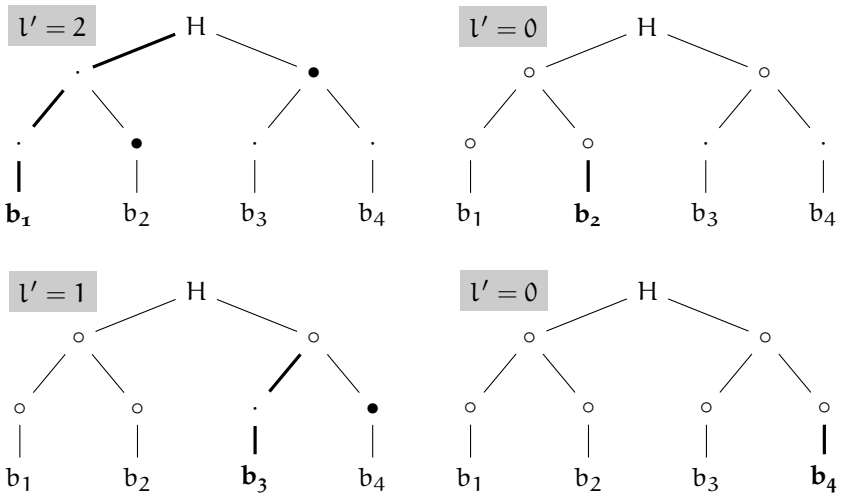


Figure 35: Exemplary Merkle tree ($w = 4$) visualizing optimal flow authenticator lengths l' for different PKT identifiers b_k. Bullets (•) indicate tree nodes that have to be included in PKT k. Circles (○) are known tree nodes sent in previous PKTs. Dots (·) are unknown nodes but are not required to authenticate b_k. Thick lines indicate the verification path.

forwarding state per flow. In addition, LIDOR nodes maintain separate per-neighbor reliability estimators for every encountered flow. The reliability estimator $\mu^H_{x,j} \in [0,1]$ of a node x for its neighbor j for the flow H is computed as a running average of the PKT delivery rate (i. e., the number of valid ACKs returned from a neighbor). It has been shown in [64, 146] that this approach provides lightweight protection against any accidental and deliberate packet loss, including hard-to-detect selective packet dropping, i. e., greyhole attacks. In short, the metric is a running average of successfully delivered PKTs via a particular neighbor. In Section 8.4, we discuss the calculation of the reliability estimator in detail. We further prove that previous approaches [64] are not secure, i. e., they might not converge towards an attacker-free path if attackers are present in the network. Also, we prove that LIDOR's approach converges.

LIDOR adopts the reliability estimator from previous works but applies it in a way that achieves provable convergence.

FORWARDING DECISION A node makes a probabilistic forwarding decision that is based on its reliability estimators. The intuition is that we use broadcast for *route exploration* if no reliable path exists, and otherwise unicast for *route exploitation* to keep using a working path. A node unicasts a PKT with probability μ^H_x to the most reliable neighbor where $\mu^H_x = \max_j \mu^H_{x,j}$, i. e., the reliability estimator of the most reliable neighbor. Should two or more neighbors have the same reliability estimator, we use the average round-trip time (RTT) to break the tie. Otherwise, the PKT is broadcast to all neighbors.

Should two neighbors be equally reliable, we choose the one that achieved the lowest RTT.

Input: k, l, h
1 $l' \leftarrow 0$
2 **while** $l' < l$ **do**
3 $k_{left} \leftarrow (k \oplus (1 \ll l)) \wedge (-1 \ll l)$
4 $k_{right} \leftarrow k_{left} + (1 \ll l) - 1$
5 **for** $k' \in \left[k_{left}, k_{right}\right]$ **do**
6 **if** h has acknowledged k' **then**
7 **return** l'
8 **end if**
9 **end for**
10 $l' \leftarrow l' + 1$
11 **end while**
12 **return** l

Program 2: Calculation of the minimal flow authenticator length.

IN-NETWORK MERKLE TREE COMPRESSION The size of the flow authenticator f_k has a significant impact on the protocol overhead. f_k grows linearly with the tree height l. In previous works [64, 88, 100], all sibling nodes in the tree from the leaf to the root (tree nodes $x_1, \ldots, x_l$ in Figure 34) are included in each packet. For large trees, this naïve approach generates significant overhead. For instance, a tree of height $l = 8$ allowing to send $2^8 = 256$ PKTs under a single flow requires the header to include eight hash values in f_k. In absolute terms, this results in $8 \times 16 = 128$ bytes overhead *per PKT* when using a collision-resistant hash function with a 16-byte output.

Transmitting all intermediate Merkle tree nodes for authentication creates a significant overhead.

LIDOR employs a more efficient method. LIDOR nodes incrementally construct the Merkle tree as they receive new b_k and f_k values (note that intermediate nodes cannot construct the entire tree from n since they do not possess K_{sd}). Starting from the second received PKT, l' (with $l' < l$) new tree nodes are required to authenticate the flow. The idea is visualized in Figure 35. In a stable network, the lower-bound average of l' is *constant* with $(2^l - 1)/2^l < 1$, which leads to an eight-fold overhead reduction compared to sending the full authenticator length as in the above example.

We require nodes to cache parts of the tree to reduce redundancy.

In order to devise a practical distributed algorithm to calculate l', nodes need to keep track of the current Merkle tree state of their neighbors. LIDOR nodes do this by leveraging the ACKs received from their neighbors: when receiving a_k from neighbor h, a node knows that h has the kth leaf of the Merkle tree, as well as the authenticated path from this leaf to the root. Otherwise, h would have been unable to authenticate and forward the kth PKT in the first place. To determine minimal l', i. e., the shortest possible flow authenticator length for which the next-hop node will still be able to authenticate b_k, we use Program 2. Note that the algorithm is robust against (adversarial)

We devise an algorithm that calculates which tree nodes have to be included in a given PKT.

packet dropping as it always includes the full flow authenticator if the neighbor has not acknowledged a packet of that flow. When broadcasting a PKT, we need to make sure that all neighbors can authenticate b_k. Therefore, we set l' to the maximum among all neighbors, i. e., $l' = \max_h l'_h$ with l'_h being the minimal flow authenticator length for neighbor h.

This scheme assures that (1) a node can always authenticate any PKT received from another correct node; and (2) the transmitted flow authenticator does not convey redundant information, i. e., it is exactly as long as it needs to be for minimal PKT overhead. Note that our scheme is agnostic to packet loss and packet reordering.

A node can request a full flow authenticator if it lost state.

In the rare case that a node loses state and is unable to authenticate the flow because f_k is too short, it may "bounce" the PKT back to the sender with a unicast to request for a retransmission with the complete f_k. The receiving node then removes the flag and returns the complete f_k of length l to the requester. This retransmission may only be done once per neighbor and PKT to prevent DoS attacks where an attacker would effectively circumvent the duplicate check.

If no ACK is returned, the reliability estimator of that neighbor decreases.

AWAITING RESPONSE When forwarding a PKT, a node starts a timer for each new b_k that expires after a timeout T_{ACK}. In addition to starting the timer, the tuple (m, b_k, H) together with the forwarding decision is added to a collection of previously seen PKTs. This tuple serves the purpose of authenticating ACKs (as described in Section 8.2.5), and it is discarded after the PKT timer expires. If the timer expires and no ACK has been received, the reliability estimator for the next-hop node decreases. To avoid premature false positives (timer expires before ACK was returned) or late true positives (lost ACK is detected too late), we employ an adaptive TCP-inspired timeout calculation for T_{ACK} following the same approach as in [161].

8.2.4 *Packet Reception*

The destination node verifies that the PKT payload is authentic and, if yes, generates an acknowledgment.

In addition to packet verification, the destination node checks the PKT's authentication tag σ. PKTs with an invalid σ are discarded. For the first PKT of a flow, the destination locally computes the full Merkle tree using the nonce n as described in Section 8.2.1. For every received PKT, the destination selects a_k from the Merkle tree. It then generates the appropriate acknowledgment (ACK), as shown in Equation (22), which consists of the packet digest m and the ACK authenticator a_k. The ACK is then returned to the sender.

$$\text{ACK} = (m, a_k) \tag{22}$$

8.2.5 *Acknowledgment Handling*

ACKs act as secure proofs of delivery and allow for updating the neighbor reliability estimators. Upon ACK reception, a node calculates $b_k = hash(a_k)$ and checks whether the ACK belongs to a valid PKT, i. e., whether any PKT with m and b_k has been forwarded before. If not, the ACK is dropped. Otherwise, and if the sending node matches the previous forwarding decision, the reliability estimator for the sending node is increased. The ACK is then forwarded to the neighbors from which the node received copies of the corresponding PKT. If a node receives multiple ACKs, only the first one is forwarded. In addition, a valid ACK updates the bit vector used for duplicate detection and neighbor Merkle tree state, as described in Section 8.2.2.

When receiving a valid ACK, we increase the reliability of the sending node.

8.3 OVERHEAD ANALYSIS

In this section, we present a comparative analysis between the overhead of the benchmark and LIDOR protocol for a converged scenario. For this analysis, we consider the number of hash values in the flow authenticator f_k as the protocol overhead. The overhead is added for each hop. In the converged scenario, a stabilized path exists between the source and the destination. Thus, all the nodes in the stabilized path will unicast the packet to their most reliable neighbor. This implies that the recipient of consecutive packets from a node remains the same.

We compute the overhead for the case that LIDOR has converged.

Let I denote the number of hops between the source and destination nodes. Let l be the height of Merkle Tree. Therefore, the number of packets transmitted for a flow of Merkle tree is given by 2^l. Let $|hash(\cdot)|$ denote the size of one hash value in bytes.

8.3.1 *Benchmark Protocol*

In the benchmark protocol, the source transmits all the hash values for each packet. The total number of hash values required for authenticating the packet is the same as the height of the Merkle tree, l. Let O_k^B denote the overhead for the kth packet of the flow. Then,

The per-packet overhead scales linearly with the tree height.

$$\begin{aligned} O_k^B &= \sum_{i=1}^{I} l|hash(\cdot)| + |e_k| \,, \\ &= I(l|hash(\cdot)| + |e_k|) \,, \end{aligned}$$

where $|e_k|$ denotes the size of the hash value of an encrypted ACK e_k in bytes as used in [64]. Let O_B denote the total overhead of the benchmark protocol. Then,

$$O_B = \sum_{k=1}^{2^l} O_k^B = (2^l I)(l|hash(\cdot)| + |e_k|) \,. \tag{23}$$

8.3.2 *LIDOR Protocol*

The packet contains a nonce which the destination uses to reconstruct the Merkle tree and generate the ACKs.

In LIDOR, the first packet of the flow carries a nonce that is used by the destination node to re-construct the Merkle tree. The intermediary nodes require l hash values of the Merkle tree to verify that the packet belongs to the same flow. Thus, the overhead for the first packet, denoted by O_1^L, is given by

$$O_1^L = Il|hash(\cdot)| + I|n| \,, \tag{24}$$

where $|n|$ is the size of the nonce in bytes. The number of hash values required for authentication may reduce if the recipient of the current packet had previously received one or more packets belonging to the same flow. Since the network is converged, which implies each node sends all the packets to the same node, it can save upon the number of hash values required for transmission, as shown in Program 2.

In LIDOR, we need to transmit $2^l - 1$ tree nodes for a single flow.

Let η denote the number of hash values required to be transmitted when all the packets are sent to the same node, where η is a non-negative integer in the range $[0, l]$. Then,

$$\sum_{k=1}^{2^l} \eta = 2^l - 1 \,. \tag{25}$$

This implies that the total number of hash values to be transmitted when all the packets are sent to the same node is given by $2^l - 1$.

Subsequent packets carry a nonce until the first ACK of the flow is received.

The nonce is retransmitted in future packets if $t_k + t_{RTT} > t_{k+1}$ where t_k is the sending time of the kth packet, and t_{RTT} denotes the RTT. The best case is if the nonce is transmitted only for the first packet. Whereas, the worst case is if the nonce is transmitted for each packet of the flow. Therefore, we assume that the nonce is transmitted for $2 \leqslant k \leqslant \lambda_n$ packets, where $\lambda_n (\leqslant 2^l - 1)$. Thus, the probability of retransmission of nonce, denoted by p_n, is given by

$$p_n = \frac{\lambda_n}{2^l - 1} \,.$$

Then, O_k^L for $2 \leqslant k \leqslant 2^l - 1$ is given by

$$O_k^L = I(\eta|hash(\cdot)| + p_n|n|) \,. \tag{26}$$

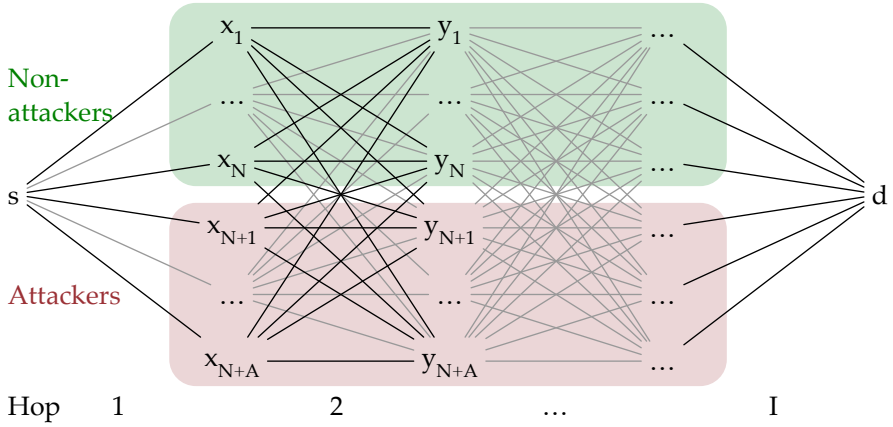


Figure 36: Corridor forwarding model used in our convergence analysis. There are $I-1$ relay layers connecting the source and destination. Each layer consists of N non-attackers and A attackers.

Let O_L denote the overhead of LIDOR protocol for a flow. Then, from Equations (24) to (26), we have

$$\begin{aligned} O_L &= \left[\sum_{i=1}^{2^l} \eta\right] I|hash(\cdot)| + (1 + (2^l - 1)p_n) I|n| \,, \\ &= I((2^l - 1)|hash(\cdot)| + (1 + (2^l - 1)p_n)|n|) \,. \end{aligned} \tag{27}$$

Let ΔO denote the difference of the benchmark and LIDOR protocol. From Equations (23) and (27), ΔO is given by

$$\begin{aligned} \Delta O = I|hash(\cdot)| \left(2^l(l-1)+1\right) + \\ \left(2^l(|e_k|) - \left(1 + (2^l - 1)p_n\right)|n|\right) \,. \end{aligned}$$

8.4 CONVERGENCE ANALYSIS

In this section, we present the lower and upper bounds on the number of packets required to achieve convergence in the benchmark and LIDOR protocol under a greyhole attack.[2] We assume reliable wireless transmissions, i. e., there is no loss on the channel. The network consists of attacker and non-attacker nodes. Unlike a non-attacker node, the attacker node drops the packet if the packet has been unicast to it. However, both non-attacker and attacker nodes forward the packet and relay the ACK in case of a broadcast.

For our convergence analysis, we assume a lossless wireless channel.

Let s and d be the source and destination nodes, respectively. Let μ_s^H denote the maximum reliability value among all the one-hop neighbors of s for the flow H. We say that a path has converged when (1) the reliability of a non-attacker node is maximum, i. e., μ_s^H corresponds to a non-attacker node, (2) μ_s^H does not decrease, and (3) $\mu_s^H \geqslant 1 - \epsilon$,

In a converged path, nodes will unicast only to non-attacker relays.

2 Note that LIDOR and the benchmark are resilient to blackhole attacks. An attacker that drops all packets would effectively remove itself from the network.

where $\epsilon(\approx 0)$ is a suitable threshold on the reliability of the network. Let B_x^M denote the successive broadcast of M packets by the node x, and let B^M denote the successive broadcast of M packets by all the nodes. Let U_x^y denote the unicast of a packet from the node x to the node y in the system.

In the following analysis, we use a generic corridor [111] forwarding network model as depicted in Figure 36.

8.4.1 *Non-Convergence of Benchmark Protocol*

In this section, we present the convergence analysis for the benchmark protocol. In the benchmark protocol, each node computes a reliability estimator for a flow H for the jth neighbor. Let $\mu_{x,j}^H$ denote the reliability estimator computed by the node x for its jth neighbor for the flow H. Then,

$$\mu_x^H = \max_j \mu_{x,j}^H \, .$$

The benchmark protocol maintains two reliability estimators to give preference to paths with a low RTT.

Whereas, $\mu_{x,j}^H$ is given by

$$\mu_{x,j}^H = \frac{\mu_{x,j}^{a,H} + \mu_{x,j}^{f,H}}{2} \, , \tag{28}$$

where $\mu_{x,j}^{a,H}$ and $\mu_{x,j}^{f,H}$ denote the reliability estimators that count the reception of all ACKs and the first ACKs, respectively for the jth neighbor of the node x and flow H. $\mu_{x,j}^{a,H}$ is updated for all neighbors that respond with an ACK, while $\mu_{x,j}^{a,H}$ only increases for the neighbor that is the first to respond with an ACK for an individual PKT. Then, $\mu_{x,j}^{a,H}$ is computed as

$$\mu_{x,j}^{a,H} = \frac{\alpha_{x,j}^{a,H}}{\alpha_{x,j}^{a,H} + \beta_{x,j}^{a,H}} \, , \tag{29}$$

where $\alpha_{x,j}^{a,H}$ and $\beta_{x,j}^{a,H}$ count successful and unsuccessful packet delivery attempts,[3] respectively for the jth neighbor of the node x and flow H. The $\alpha_{x,j}^{a,H}$ and $\beta_{x,j}^{a,H}$ update for each unsuccessful transmission as

$$\begin{aligned} \alpha_{x,j}^{a,H} &\leftarrow \delta \alpha_{x,j}^{a,H} \, , \\ \beta_{x,j}^{a,H} &\leftarrow \delta \beta_{x,j}^{a,H} + 1 \, . \end{aligned} \tag{30}$$

Two estimators count the number of successful and unsuccessful deliveries using a decay factor δ.

Whereas, $\alpha_{x,j}^{a,H}$ and $\beta_{x,j}^{a,H}$ update for each successful delivery as

$$\begin{aligned} \alpha_{x,j}^{a,H} &\leftarrow \delta \alpha_{x,j}^{a,H} + 1 \, , \\ \beta_{x,j}^{a,H} &\leftarrow \delta \beta_{x,j}^{a,H} \, . \end{aligned} \tag{31}$$

3 A successful delivery is counted when receiving a valid ACK before T_{ACK} expired. An unsuccessful delivery is counted when no ACK is received or it arrives too late.

The parameter δ is a decay factor and controls the adaptivity to changes with $0 < \delta < 1$. Similarly, $\mu^{f,H}_{x,j}$ can be computed as

$$\mu^{f,H}_{x,j} = \frac{\alpha^{f,H}_{x,j}}{\alpha^{f,H}_{x,j} + \beta^{f,H}_{x,j}} \,. \tag{32}$$

The $\alpha^{f,H}_{x,j}$ and $\beta^{f,H}_{x,j}$ update in the same manner as $\alpha^{a,H}_{x,j}$ and $\beta^{a,H}_{x,j}$ for both successful and unsuccessful transmission. Initially, $\alpha^{a,H}_{x,j} = \alpha^{f,H}_{x,j} = 0\ \forall x, j$ and $\beta^{a,H}_{x,j} = \beta^{a,H}_{x,j} = 1\ \forall x, j$. Let M denote the number of broadcast packets by the source node. If we assume that all the packets are received successfully, the reliability of the neighbors of the source increase for all the packets. Let Δ_M denote the reliability of the neighbors after the transmission of M packets. Then, using Equations (29) and (31), Δ_M is given by

We compute the reliability of the neighbors Δ_M after M successful deliveries.

$$\Delta_M = \left(\frac{\sum_{i=0}^{M-1} \delta^i}{\sum_{i=0}^{M} \delta^i} \right) \,. \tag{33}$$

Let us consider that the nodes x_1 and x_2 connect s and d via two hops. Let x_1 be a non-attacker node and x_2 be an attacker node, i. e., $A = N = 1$ and $I = 2$ in Figure 36. Consider B^1, $U^{x_2}_s$, B^1 as a packet transmission scenario. We assume that x_2 provides the first ACK for both broadcasts. Then, using Equations (28) to (33), we have

$$\begin{aligned} \mu^{H}_{s,x_1} &= \frac{1}{2}\Delta_2 \,, \\ \mu^{H}_{s,x_2} &= \frac{\delta^2 + 1}{\delta^3 + \delta^2 + \delta + 1} = \frac{1}{1+\delta} \,. \end{aligned}$$

Clearly, $\mu^{H}_{s,x_1} < \mu^{H}_{s,x_2}$. Therefore, if s selects to unicast, it will unicast only to x_2. Consider the case when s broadcast after $N-1$ successive B^1, U^y_s transmissions, and x_2 always provides the first ACK. Then, μ^{H}_{s,x_1} and μ^{H}_{s,x_2} after the Nth broadcast,

We show that the benchmark protocol does not converge by a negative example.

$$\begin{aligned} \mu^{H}_{s,x_1} &= \frac{1}{2}\Delta_N \,, \\ \mu^{H}_{s,x_2} &= \frac{\sum_{i=0}^{N-1} \delta^{2i}}{\sum_{i=0}^{2N-1} \delta^i} = \frac{1}{1+\delta} \,. \end{aligned} \tag{34}$$

Let Z denote the difference of μ^{H}_{s,x_1} and μ^{H}_{s,x_2}. Using Equation (34), we have

$$\begin{aligned} Z &= \frac{1}{2}\Delta_N - \frac{1}{1+\delta} \,, \\ &= \frac{1-\delta^N}{2(1-\delta^{N+1})} - \frac{1}{1+\delta} \,, \\ &= \frac{(\delta-1)(1+\delta^N)}{2(1+\delta)(1-\delta^{N+1})} \,. \end{aligned} \tag{35}$$

Since $\delta < 1$, $Z < 0$ for any N, $\mu^H_{s,x_1} < \mu^H_{s,x_2}$ is implied. Thus, there is a possibility that the network gets stuck in the loop of B^1, $U^{x_2}_s$ when x_2 only provides the first ACK. This implies that there exists a possibility that s never converges to the non-attacker neighbor, i. e., x_1. Next, we describe the convergence in the LIDOR protocol wherein each node will converge to a non-attacker neighbor.

8.4.2 *Convergence of LIDOR Protocol*

LIDOR does not differentiate between $\mu^{a,H}_{x,j}$ and $\mu^{f,H}_{x,j}$, i. e., $\mu^H_{x,j} = \mu^{a,H}_{x,j} = \mu^{f,H}_{x,j}$ and updates $\mu^H_{x,j}$ for all received ACKs. However, the procedures of updating $\mu^{a,H}_{x,j}$ and selecting whether to broadcast or unicast a packet are the same as in the benchmark. In the case that two or more neighbors have the same reliability value, an RTT estimation from past transmissions similar to TCP is used to break the tie. In the following, we present the convergence analysis with and without attackers.

We modify the reliability update mechanism to achieve convergence.

NO ATTACKERS In this scenario, we assume that all nodes are non-attackers. Initially, s broadcast the packet to all its neighbors. Since there is no loss, the reliability will increase for all the neighbors of s. After the broadcast of a few packets, s will unicast to the node with least RTT. Let x be the neighbor of s that has the lowest RTT. Thus, once s unicasts to x, the reliability of x becomes more than the reliability of any other neighbor of s. Since the reliability of x can only increase, s will unicast to x with high probability and hence converge to x. This implies the node which connects s and d in the least number of hops and has the lowest RTT will be chosen as a unicast forwarder.

We first show that LIDOR converges in a scenario where no attackers are present.

Considering B^{M-1}_s and U^x_s, we have $\mu^H_s = \Delta_M$. Then, from the definition of convergence, we have

$$\mu^H_s = \Delta_M \geqslant 1 - \epsilon, \tag{36}$$

Substituting Equation (33) into Equation (36), we have

We calculate the minimum number of packets to achieve convergence as a lower bound.

$$\left(\frac{\sum_{i=0}^{M-1} \delta^i}{\sum_{i=0}^{M} \delta^i}\right) \geqslant 1 - \epsilon\,,$$

$$\delta^M \leqslant \epsilon \frac{1 - \delta^{M+1}}{1 - \delta}\,,$$

$$M \geqslant \frac{1}{\ln(\delta)} \left[\ln\left(\frac{\epsilon}{\epsilon\delta + 1 - \delta}\right)\right]\,.$$

Let M_{min} denote the minimum number of packets to attain convergence for the source node in the absence of attackers. Then,

$$M_{min} = \frac{1}{\ln(\delta)} \left[\ln\left(\frac{\epsilon}{\epsilon\delta + 1 - \delta}\right)\right]\,. \tag{37}$$

ATTACKERS WITH ONE RELAY LAYER In this scenario, we consider that s and d have one layer of relay nodes between them, i. e., s and d are connected in two hops via relay nodes. The layer of relay nodes consists of N non-attacker and A attacker nodes (see Figure 36. Let x_i denote the ith non-attacker node for $i \in \{1, 2, \ldots, N\}$. Similarly, let x_i denote the ith attacker node for $i \in \{N+1, N+2, \ldots, N+A\}$. The reliability increases on the unicast for each x_i for $i \in \{1, 2, \ldots, N\}$ whereas the reliability decreases on the unicast for each x_i for $i \in \{N+1, N+2, \ldots, N+A\}$. Therefore, once x_i for $i \in \{N+1, N+2, \ldots, N+A\}$ receives a unicast, its reliability decreases and hence it will not receive any unicast in the future. Intuitively, after dropping any packet, the attacker will no longer be among the most reliable neighbors and, therefore, a unicast candidate.

We first prove convergence for a network with a single relay layer.

We consider that s has transmitted $M(> A)$ packets. Considering all possible combinations of broadcast and unicast, the reliability of the most reliable non-attacker node lies between $[\Delta_{M-A}, \Delta_M]$. The reliability of Δ_M corresponds to the best case path wherein the most reliable node has received the first unicast. It also corresponds to the sequence of M broadcasts, which is less probable. The reliability of Δ_{M-A} corresponds to the worst-case path which contains one unicast to each of the A attacker nodes. Then, the upper bound on the number of packets required for convergence is computed by considering the worst-case reliability of the non-attacker node, i. e., Δ_{M-A}. Therefore, from the definition of convergence, we have

We consider the worst-case scenario, where the source tries to unicast to each attacker.

$$\Delta_{M-A} \geqslant 1 - \epsilon \,. \tag{38}$$

From Equation (33), we have

$$\Delta_{M-A} = \left(\frac{\sum_{i=0}^{M-A-1} \delta^i}{\sum_{i=0}^{M-A} \delta^i} \right) \,. \tag{39}$$

Let M_{max} denote the number of packets required for convergence in the worst case. Substituting Equation (39) into Equation (38), we have

$$\begin{aligned} \frac{\sum_{i=0}^{M-A-1} \delta^i}{\sum_{i=0}^{M-A} \delta^i} &\geqslant 1 - \epsilon \,, \\ \delta^M &\leqslant \frac{\epsilon \delta^A}{\epsilon\delta + (1-\delta)} \,, \\ M_{max} &= \frac{1}{\ln(\delta)} \left[\ln \left(\frac{\epsilon \delta^A}{\epsilon\delta + (1-\delta)} \right) \right] \,. \end{aligned} \tag{40}$$

The lower bound on the number of packets required for convergence is obtained by considering the best case reliability, i. e., Δ_M. Therefore, the lower bound on the number of packets required for convergence,

denoted by M_{min}, is given by Equation (37). From Equations (37) and (40), we have

$$M_{max} - M_{min} = \frac{1}{\ln(\delta)} \left(\ln(\epsilon \delta^A) - \ln(\epsilon) \right) ,$$
$$= \frac{1}{\ln(\delta)} \ln(\delta^A) = A .$$

After at most A unicast transmissions, the source will choose a non-attacker node.

Thus, $M_{max} = M_{min} + A$. Since there are A attacker nodes to which unicast can happen only once, the convergence gets delayed by A packets if s happens to choose the worst-case path, i. e., s unicasts to each attacker node.

ATTACKERS WITH MULTIPLE RELAY LAYERS In this section, we present the analysis for the network containing N non-attacker and A attacker nodes for $I-1$ layers of relay nodes between s and d. Let us consider a network with $I = 3$. Let x_i and y_i, where $i \in \{1, \ldots, N\}$, denote the ith non-attacker node in the first and second layer of relay nodes, respectively. Let x_i and y_i , where $i \in \{N+1, \ldots, N+A\}$, denote the ith attacker node in the first and second layer of relay nodes, respectively. The μ^H_{s,x_i} for any non-attacker node (i. e., $i \in \{1, \ldots, N\}$) decreases if x_i unicast to any attacker node y_j where $j \in \{N+1, \ldots, N+A\}$. Therefore, each non-attacker node can have A unsuccessful unicast attempts. Therefore, the worst-case path for the network with $I = 3$ consists of A iterative cycles of successive broadcast followed by a unicast to each attacker node by the source node and an unsuccessful attempt of each non-attacker node being unicast by the source node. A unicast to each attacker node and a series of successive follows the end of the iterative cycle. Let us consider $N = 2$ and $A = 3$. The worst-case path for this network is $B^{M_{min}}, U_s^{x_3}, U_s^{x_4}, U_s^{x_5}, U_s^{x_1} U_{x_1}^{y_3}, U_s^{x_2} U_{x_2}^{y_3}, B^{G_1}, U_s^{x_3}, U_s^{x_4}, U_s^{x_5}, U_s^{x_1} U_{x_1}^{y_4}, U_s^{x_2} U_{x_2}^{y_4}, B^{G_2}, U_s^{x_3}, U_s^{x_4}, U_s^{x_5}, U_s^{x_1} U_{x_1}^{y_5}, U_s^{x_2} U_{x_2}^{y_5}, B^{G_3}, U_s^{x_3}, U_s^{x_4}, U_s^{x_5}$.

We extend the analysis to network consisting of multiple relay layers each consisting of the same number of attacker and non-attacker nodes.

In the worst case, the number of packets to achieve convergence increases as the number of possible paths.

The μ^H_{s,x_i} $\forall i$ decreases and becomes equal before $B_s^{G_\nu}$, where $\nu = \{1, 2, 3\}$. G_ν represents the minimum number of broadcasts to be performed by all the nodes such that $\mu^H_{s,x_i} \geqslant 1 - \epsilon$ $\forall i$. Then, G_1 is given by

$$G_1 = \frac{1}{\ln(\delta)} \ln \left(\frac{\frac{\epsilon}{1-\delta}}{\delta^{M_{min}+1} \left(1 + \frac{\epsilon\delta}{1-\delta}\right) + 1} \right) . \tag{41}$$

Whereas, G_ν for any $\nu > 1$ is given by

$$\Gamma_\nu = \sum_{m=1}^{\nu-1} \left(\delta^{\left(\sum_{l=\nu-m}^{\nu-1} G_l\right)+m} \right) ,$$
$$\Upsilon_\nu = \delta^{M_{min}+\nu+\left(\sum_{l=1}^{\nu-1} G_l\right)} \left(1 + \frac{\epsilon\delta}{1-\delta}\right) + 1 + \Gamma_\nu ,$$
$$G_\nu = \frac{1}{\ln(\delta)} \ln \left(\frac{\frac{\epsilon}{1-\delta}}{\Upsilon_\nu} \right) . \tag{42}$$

Then, using Equations (41) and (42), the upper bound on the number of packets required for convergence for a network with $I = 3$ hops is given by

$$M_{max} = M_{min} + (A+1)(A) + NA + \left(\sum_{\nu=1}^{A} G_\nu\right) .$$

For a network with I hops, the following holds in general. (1) The number of unsuccessful unicast attempts for each non-attacker nodes is given by the number of unicast packets for the network with $I-1$ hops. (2) The number of unicast transmissions by the source node to each attacker node is one additional the number of unicast transmissions in the network with $I-1$ hops. (3) The number of successive broadcasts is given by the number of unicast for the network with $I-1$ hops.

Let ξ_I denote the number of unicast packets in the network with I hops. Then, $\xi_2 = A$, $\xi_3 = A(A+1) + NA$, and $\xi_I = (N+A)\xi_{I-1} + A$. On solving further, we obtain

$$\begin{aligned}
\xi_I &= (N+A)^{I-1}\xi_2 + \left(\sum_{i=0}^{I-2}(N+A)\right) A \,, \\
&= (N+A)^{I-1}A + \left(\sum_{i=0}^{I-2}(N+A)\right) A \,, \\
&= \left(\sum_{i=0}^{I-1}(N+A)^i\right) A \,, \\
&= \left(\frac{(N+A)^I - 1}{N+A-1}\right) A \,. \qquad (43)
\end{aligned}$$

Then, using Equations (41) to (43), the upper bound on the number of packets required for a network with I hops is given by

$$M_{max} = M_{min} + \xi_I + \left(\sum_{\nu=1}^{\xi_{I-1}} G_\nu\right) . \qquad (44)$$

8.5 IMPLEMENTATION

We choose the Click modular router [106] for LIDOR implementation to allow for a realistic evaluation on both real hardware and simulation. In this section, we discuss suitable candidate functions for LIDOR's cryptographic primitives and devise a practical link-local broadcast authentication scheme based on symmetric cryptography.

We implement LIDOR in the Click modular router.

8.5.1 *Reference Platforms*

We evaluate LIDOR on heterogeneous platforms with different CPU architectures, processing capabilities, memory configurations (256 MB

to 16 GB RAM), and operating systems (OSs) (Debian Linux, Android 6, macOS 10.11). In particular, these are:

Click can be run a different UNIX-like platforms.

- *ALIX* [148] (32-bit single-core 500 MHz AMD Geode LX, 256 MB RAM, Debian Linux),
- *APU* [149] (64-bit dual-core 1 GHz AMD T40E, 4 GB RAM, Debian Linux),
- LG *Nexus* 5 (32-bit quad-core 2.3 GHz Snapdragon 800, 2 GB RAM, rooted Android 6), and
- *MacBook* Pro (early 2015, 64-bit dual-core 2.9 GHz Intel Core i5, 16 GB RAM, macOS 10.11).

8.5.2 *Cryptographic Primitives*

The choice of efficient cryptographic primitives is imperative for any practical communication protocol. In LIDOR, cryptographic operations consume the longest processing time during packet forwarding and constitute the major portion of the communication overhead. Our implementation relies on primitives provided by the lightweight, cross-platform libsodium (v1.0.11) [50]. Table 11 shows a performance comparison between different candidate algorithms on our reference platforms. The table also gives an intuition on why public-key cryptography is unsuitable to be used on a per-PKT basis: the cumulated forwarding delay would be unacceptably large. We select LIDOR's cryptographic primitives as follows.

We choose cryptographic primitives based on computation time and output size.

- $\mathtt{hash}(\cdot)$ is implemented as Blake2b with an output size of 16 bytes. Note that the hash function used to construct the Merkle tree does not need to be collision-resistant but only preimage and second-preimage resistant.[4] Hence, a 128-bit security margin is sufficiently large. Blake2b is optimized for 64-bit architectures and performs better than, for example, SHA-2.
- $\mathtt{prf}(K_{sd}, n)$ is implemented as XSalsa20/20, a stream cipher using 256-bit keys and 192-bit nonces.
- $\mathtt{tag}(K_{sd}, \cdot)$ is implemented as SipHash-2-4 [17], which generates small 8-byte authentication tags for short-input (order of kilobytes) packets using a shared secret K_{sd}. In contrast, general-purpose message authentication codes such as HMAC-SHA2 are computationally more expensive and create larger tags (8 vs. 32 bytes).

The selected cryptographic primitives are provided by libsodium.

4 *Collision resistance*: given $\mathtt{hash}(\cdot)$, it is hard to find x and x' such that $\mathtt{hash}(x) = \mathtt{hash}(x')$. *Preimage resistance*: given y, it is hard to find x such that $\mathtt{hash}(x) = y$. *Second-preimage resistance*: given x, it is hard to find $x' \neq x$ such that $\mathtt{hash}(x) = \mathtt{hash}(x')$.

FUNCTION	ALGORITHM	ALIX	APU	NEXUS	MACBOOK
$\mathtt{hash}(\cdot)$	SHA-256	184	36	18	6
	Blake2b	167	8	29	3
$\mathtt{prf}(\cdot,\cdot)$	XSalsa20/20	97	12	12	5
$\mathtt{tag}(\cdot,\cdot)$	SipHash-2-4	66	4	8	1
	HMAC-SHA-512	417	35	92	6
	Ed25519 (verify)	8761	1479	815	168

Table 11: Computation time in µs of several cryptographic algorithms on various platforms for 1024-byte strings averaged over 10000 runs. Ed25519 is a state-of-the-art elliptic-curve signature scheme and included as a reference.

8.5.3 *Practical One-Hop Broadcast Authentication*

When broadcasting, PKTs still need to be authenticated.

LIDOR requires neighbor-to-neighbor communication to be authenticated to prevent blackmailing and Sybil attacks. Cryptographic methods to authenticate broadcast communication are either based on digital signatures or on TESLA [151], which is based on symmetric-key cryptography and delayed key disclosure to achieve asymmetry. We deem both approaches impractical. Digital signatures are computationally expensive, and TESLA requires time synchronization between all nodes and introduces authentication delay, which would impede LIDOR's reactiveness to path changes. The small output size of SipHash enables us to implement a one-hop broadcast authentication scheme based on symmetric-key cryptography without TESLA's deficiencies: a forwarding node computes authentication tags for each neighbor $h \in \mathbb{F}$ (excluding the sender) and appends all of them to the PKT. A receiving node then tries to authenticate every tag. If any of them succeeds, the PKT is processed, and otherwise discarded. The expected number of SipHash calculations at a receiving node is $|\mathbb{F}|/2$, and the communication overhead is $|\mathbb{F}| \times |\mathtt{tag}(\cdot,\cdot)|$. We argue that this scheme is practical since (1) the number of neighbors is typically low compared to the total number of nodes in the network (which is what TESLA was designed for), so the communication and computational overhead for transmitting and verifying all tags remains low on average; and (2) broadcasts are used for route exploration, which rarely occurs in converged communication flows.

Instead of relying on digital signatures, we append a list of short and efficient symmetric-key authentication tags.

8.6 EXPERIMENTAL EVALUATION

Previous works [64, 160] have already shown that LIDOR's approach successfully thwarts several blackhole and greyhole attack variants. Therefore, we focus on two specific variants that have not been ad-

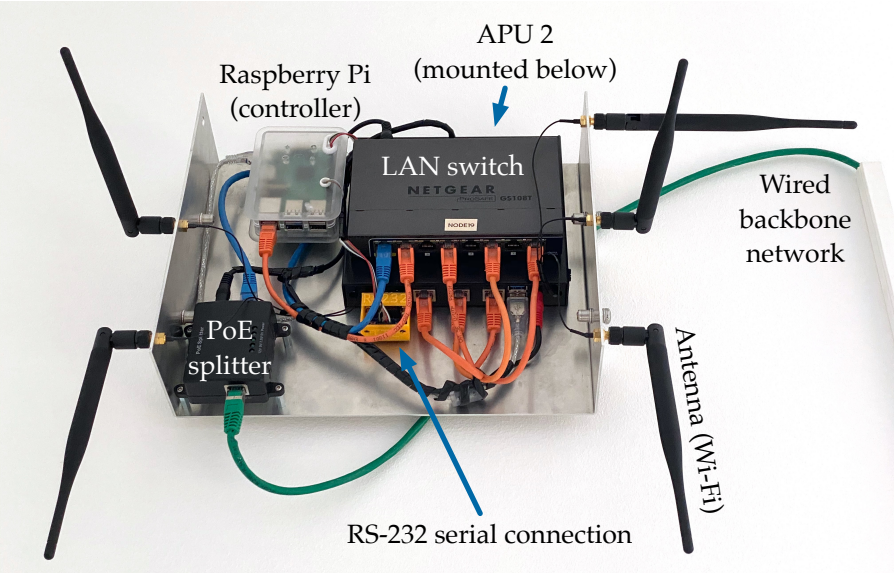


Figure 37: Picture of an wall-mounted APU node in an office environment. A Raspberry Pi acts as a controller and has a serial connection to the APU that can be used for power-cycling. The entire node is powered over Ethernet (PoE) via the wired backbone network, which is also used to orchestrate the experiments.

We focus on attacks that the benchmark has not addressed.

dressed so far. In this section, we first describe our experiment and testbed setup and then evaluate the impact of a replay-supported and a wormhole-supported greyhole attack.

8.6.1 *Test Setup*

We use a 10-node testbed with Wi-Fi ad hoc for neighbor communication.

Our testbed consists of 10 APU-based nodes [149] that are distributed in an office environment. Figure 37 depicts the architecture of a testbed node, and Figure 44 shows the network layout. For each of the following experiments, we use Wi-Fi ad hoc on channel 14 in the 2.4 GHz band to minimize interference with production networks. Before each experiment, we synchronize all nodes to a local NTP server via the nodes' Ethernet interfaces and bind the synchronization error to 0.1 ms resulting in a maximum error in the end-to-end delay measurements of 0.2 ms. In addition, each node filters its neighbors by received signal strength indication (RSSI) with a threshold of -70 dBm to avoid spurious links. We select source and destination nodes to be at a maximum distance such that the shortest path between them consists of five hops. In all experiments, the source injects 128-byte packets at a rate of 10 packets per second for an entire flow of 256 packets. We repeat each experiment 100 times. For the *wormhole* scenario, the attacker nodes use their wired Ethernet interface as a direct connection to tunnel traffic between the nodes. We use the *TPy* framework [166] to orchestrate our experiments.

We select a single source-destination pair that has the maximum distance.

8.6.2 *Summary*

We present a summary of our experiment results comparing the performance of LIDOR to the benchmark in three scenarios: (1) without attackers present, (2) with two attackers mounting a replay-supported greyhole attack, and (3) with two attackers mounting a wormhole–supported greyhole attack where the wormhole endpoints are direct neighbors to the source and destination, respectively. We show the packet delivery rate (PDR) in Figure 38, the end-to-end delay in Figure 39, and the overhead Figure 40. The figures show the mean and standard deviation over the different runs.

We evaluate delivery rate, delay, and overhead in three different scenarios.

In short, we see that LIDOR and the benchmark both achieve perfect PDRs in the benign case (Figure 38). When under attack, LIDOR's reliability reduces *only by 3.5 %* for both attacks, while the benchmark breaks down to 5.2 % and 64.1 %, respectively. Thereby, LIDOR achieves improvements over the benchmark of 91 % and 32 %, respectively. In Figure 39, we see that the end-to-end delay is similar for the benchmark (1.3 ms) and LIDOR (1.2 ms) under no attack, which is to be expected since they are both based on the same implementation. Under attack, the end-to-end delay increases. The reason is that the attacker nodes are placed in a favorable position and would allow faster delivery if they would be used as a next hop. Furthermore, Figure 40 shows the network-wide overhead of a single packet. We see that LIDOR reduces this network overhead by 35 % compared to the benchmark, which is in line with our overhead analysis in Section 8.3. In addition, LIDOR's mean overhead does not increase under attack. For the benchmark, the mean overhead decreases under attack as the packets are dropped early and do not traverse the entire path from source to destination. In the following sections, we investigate the results of the two attack scenarios in more detail.

Compared to the benchmark, LIDOR achieves a slightly better end-to-end delay but greatly improves attack resiliency.

The reduction in overhead matches our analytical result.

8.6.3 *Replay-Supported Greyhole Attack*

In this section, we expose nodes to greyhole attackers that concurrently replay expired PKTs and ACKs to reinforce their appearance as reliable forwarders. We first sketch the attacker's behavior, which is disrupting the communication between s and d. First, the attacker overhears and records any valid PKT–ACK pair of some flow H between s and d. Then, the attacker replays (i. e., broadcasts) PKT and ACK shortly after one another at an interval of T_{rep} (after an initial delay of T_{rep}). The attacker chooses T_{rep} such that it is larger than the ACK timeout, i. e., $T_{rep} > T_{ACK}$. To ensure this, the attacker conservatively sets T_{rep} to 200 ms for each pair. We limit the rate of replayed pairs to 10 per second to avoid DoS by flooding.

A replay attacker records valid PKT–ACK pairs and sends them out again when the ACK timeout has expired.

Figure 38 shows the severe impact of missing replay protection: the benchmark's reliability drops to about 5.2 %, which renders the

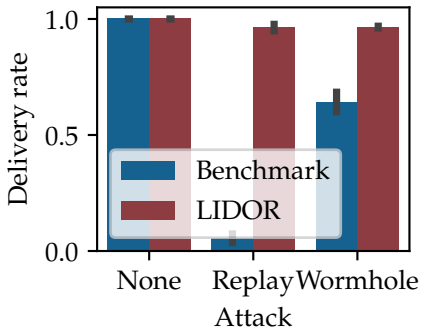


Figure 38: Average packet delivery rate in different scenarios.

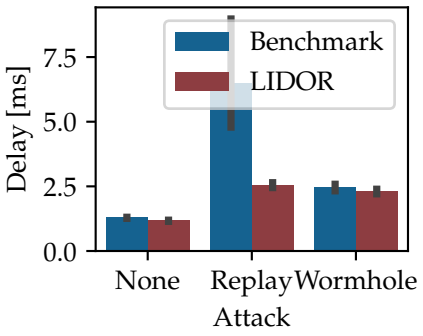


Figure 39: Average end-to-end delay in different scenarios.

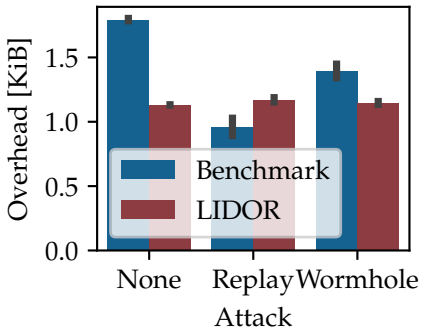


Figure 40: Average per-packet overhead in different scenarios.

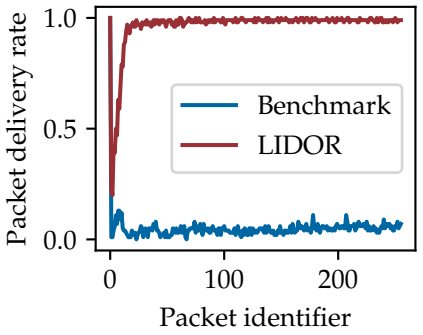


Figure 41: Packet delivery rate per packet ID under a *replay*-supported greyhole attack.

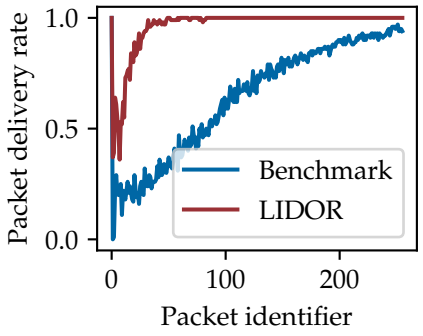


Figure 42: Packet delivery rate per packet ID under a *wormhole*-supported greyhole attack.

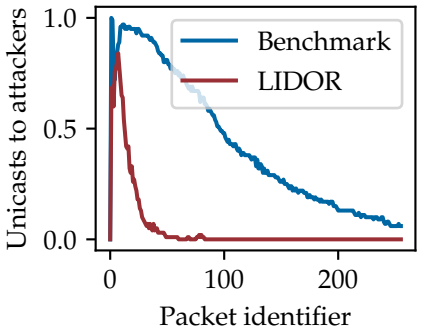
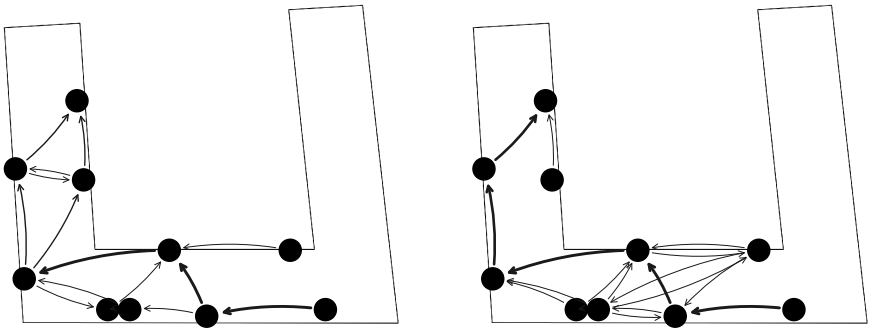


Figure 43: Attacker selection per packet ID under a *wormhole*-supported greyhole attack.



(a) Benchmark, $k \in [0, 256[$ (b) LIDOR, $k \in [0, 256[$

Figure 44: Path convergence without attack. See Figure 45 for description.

protocol unusable. On the other hand, LIDOR performs exceptionally well. There is a small drop in the mean reliability that is due to LIDOR having to route around the greyhole attackers. Once the protocol has found a reliable path, it keeps using it. This can be seen in Figure 42, where only the first few packets of each flow are less likely to be delivered.

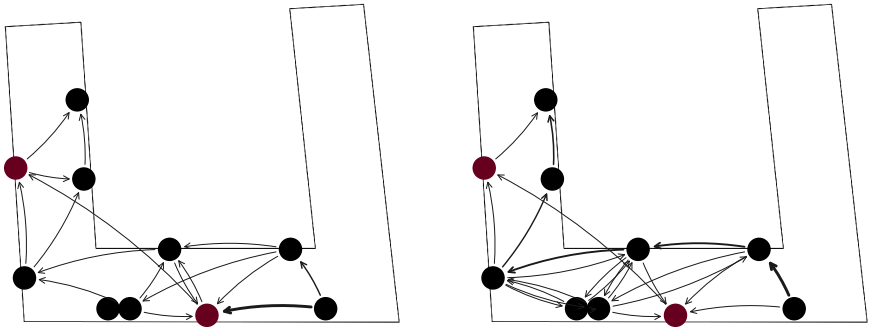
LIDOR ignores replayed PKTs and ACKs and, thus, does not suffer from reliability degradation.

8.6.4 *Wormhole-Supported Greyhole Attack*

We investigate the impact of a wormhole-supported greyhole attack on both protocols. In Figure 38, we have already seen that the PDR of LIDOR slightly drops. In fact, only the first packets of each flow are dropped as LIDOR first needs to find a valid path; i. e., it needs to *converge*. The PDR per packet ID is shown in Figure 42, where we see that after about 100 packets, the loss rate becomes zero. To verify that this is, in fact, due to the attackers being selected, we depict the relative frequency that an attacker was selected as the sole forwarder (unicast) for a given packet ID in Figure 43. The figure confirms that attackers are no longer selected as forwarders after about 100 packets. The benchmark protocol does not perform as well. We see that while attacker selection decreases and, thus, PDR increases within a flow (Figures 42 and 43), the benchmark does not completely reject the wormhole attacker as a viable forwarder. For an even more detailed analysis, we show the network graph, including forwarding decisions and the resulting path selection for different portions of a flow in Figure 45. The figure shows the average of all 100 experiment runs. For comparison, Figure 44 shows the path selection without an attack. Figures 45a and 45b show that both protocols are "fooled" by the fast link that the wormhole offers for the first packets, i. e., a path including the wormhole has the lowest RTT. While the benchmark prefers a non-adversarial path, it still uses the wormhole in a significant number of cases (Figure 45e). In contrast, LIDOR completely avoids the attackers for packets in the second half of the flow (Figure 45f).

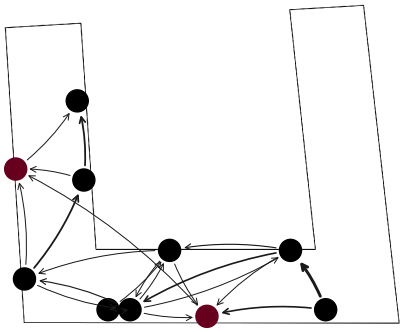
The first packets of each flow are "lost" as the protocol first has be find a viable path.

LIDOR manages to completely avoid attacker nodes after sending 100 packets.

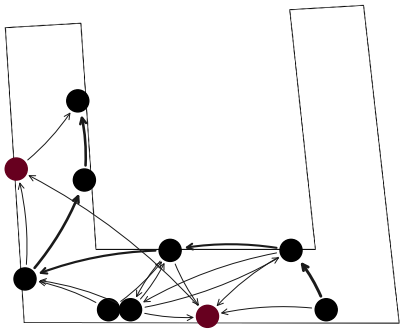


(a) Benchmark, $k \in [0, 32[$

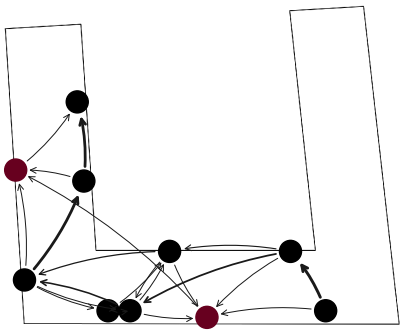
(b) LIDOR, $k \in [0, 32[$



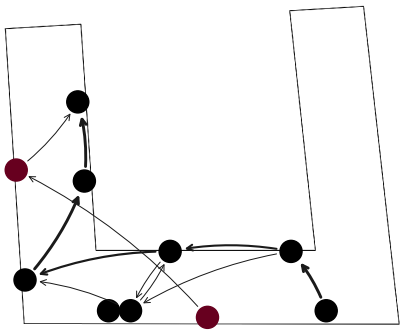
(c) Benchmark, $k \in [32, 128[$



(d) LIDOR, $k \in [32, 128[$



(e) Benchmark, $k \in [128, 256[$



(f) LIDOR, $k \in [128, 256[$

Figure 45: Path convergence under *wormhole*-supported greyhole attack. Showing unicast transmissions. Flow from bottom right to top left. Red nodes are attackers. Edge thickness indicates link usage frequency.

8.7 DISCUSSION AND SUMMARY

In this section, we elaborate on our analytical and experimental results, conjecture the applicability in large-scale IoT deployments, discuss the possibility for 100 % reliable communication, and highlight possible other application domains.

8.7.1 *Convergence: Analysis vs. Experiments*

We validate our convergence analysis with experimental results.

Our analysis in this work shows that LIDOR converges under attack while the benchmark protocol does not. Our experiments confirm that LIDOR indeed converges. However, they do not necessarily show that the benchmark does not converge either. In fact, the benchmark seems to be able to slowly approach a converged state (see Figure 42). Note that the non-convergent cases for the benchmark are statistically rare, which explains the effect. However rare in our experiment setup, these occurrences might be more prevalent in other scenarios.

8.7.2 *Feasibility for Large-Scale IoT Deployments*

LIDOR combines algorithmic and implementation optimizations to become practically efficient, yet, is still constrained by the theoretical scalability limits of wireless networks.

While LIDOR is not able to exceed the theoretical limits of scalability in wireless multihop networks [77], we employ measures to keep the network overhead as low as possible. In particular, we show that LIDOR's overhead is generally lower than the benchmark, which is due to our in-network Merkle tree compression mechanism. Also, LIDOR's overhead does not increase under attack, meaning that attacks do not impede scalability. Finally, we show the feasibility of a comprehensively DoS-resilient communication protocol in the IoT context by implementing LIDOR in a computationally efficient manner. Despite per-packet cryptographic operations, we achieve end-to-end delays in the order of 1 ms (slightly increased to 2.5 ms during an attack) in our 5-hop testbed, which confirms that the computational overhead is negligible.

8.7.3 *Towards 100 % Reliability*

LIDOR does not prevent all packet loss but could be extended with compensation mechanisms to create a reliable transport.

While LIDOR already performs exceptionally under attack, we are aware that we still encounter a certain amount of packet loss. By design, LIDOR tries to be *resilient* to all causes of packet loss but does not employ measures to compensate for loss once it occurred. We are aware that some applications might require a 100 %-reliable transport. We could increase reliability by introducing redundancy in the form of packet transmissions reactively. Thanks to the end-to-end feedback, the source knows if the destination received a specific packet and could issue a retransmission (using a new packet ID) to the destination. Such a mechanism could be implemented as a LIDOR-aware transport

overlay that receives feedback from the network layer and takes care of end-to-end retransmissions.

8.7.4 *Further Application Domains*

Assessing LIDOR's performance in new contexts requires new experiments and analytical models.

While we focus on a static IoT in this chapter, LIDOR's adaptivity to any kind of packet dropping allows for applications in more dynamic scenarios, including public safety [160] or highly-dynamic UAV-based networks [24]. However, an experimental evaluation for these types of networks is still missing. Also, our analytical results will not hold for dynamic scenarios.

8.7.5 *Summary*

LIDOR significantly improves resiliency against two specific attacks compared to the state-of-the-art.

The provisioning of robust and secure communication is crucial for safety-critical IoT applications. In this chapter, we have proposed LIDOR, a multihop communication protocol with an efficient end-to-end acknowledgment-based feedback mechanism tailored to IoT scenarios. To the best of our knowledge, LIDOR is the first protocol of its kind with proven convergence in the presence of greyhole DoS attacks. We have performed extensive experiments on our premises. These experiments have confirmed the resiliency of LIDOR against replay and wormhole attacks. Specifically, LIDOR outperforms the benchmark protocol by 91 % under replay attack and 32 % under wormhole attack in terms of PDR and reduces overhead by 35 % in the benign scenario and *does not increase significantly* under attack. The current proof-of-convergence is valid for a particular packet dropping strategy, i. e., the attacker always drops unicast packets to cause maximum harm, which is, to the best of our knowledge, the most sophisticated strategy in place [64, 160, 161]. Our experiments indicated that LIDOR converges even in a wormhole-supported greyhole attack. For reproducibility, we make the source code of our implementation publicly available (Appendix C.4).

We make our prototype available as open-source software.

9 DOS-RESILIENT ARCHIPELAGO COMMUNICATION

During disasters, existing telecommunication infrastructures are often congested or even destroyed. Local networked islands (see Chapter 8) can be formed to partially restore communication. In order to restore communication *across* islands and thereby close the gap between those partitions, disruption-tolerant networking (DTN) principles can be used to leverage node mobility, i. e., nodes that move from one island to another, to create space-time paths. Unfortunately, such distributed and resource-constrained networks are particularly susceptible to a wide range of denial-of-service (DoS) attacks. In this chapter, we present RESCUE, a resilient and secure archipelago communication framework for emergency scenarios that provides comprehensive protection against common attacks. RESCUE features a minimalistic DTN protocol that, by design, is secure against attacks such as spoofing, dropping, or blackholing. To further protect against message flooding and Sybil attacks, we present a twofold mitigation technique. First, a mobile and distributed certificate infrastructure particularly tailored to the emergency use case hinders the adversarial use of multiple identities. Second, a message buffer management scheme significantly increases resiliency against flooding attacks, even if they originate from multiple identities, without introducing additional overhead. Finally, we demonstrate the effectiveness of RESCUE via large-scale simulations in a synthetic as well as a real natural disaster scenario. Our simulation results show that RESCUE achieves competitive message delivery rates, even under flooding and Sybil attacks. In the following, we first give an overview of the system and then present its components in detail. Finally, we evaluate the protocol via simulation and discuss the results.

RESCUE is resilient to flooding attacks by Sybil adversaries.

9.1 OVERVIEW

We present RESCUE, a resilient and secure device-to-device communication framework for emergency scenarios which provides comprehensive attack protection. RESCUE's minimalistic communication protocol (Section 9.2) relies on epidemic routing, authenticated and immutable messages, and an effective acknowledgment processing. This way, common attacks, such as message or routing manipulation, blackholing, or impersonation, are already prevented. Yet, as in today's Internet infrastructure [204], the key challenge is to defend against DoS attacks originating from *individuals* as well as *multiple identities* (Sybil attack) that flood the network. For this purpose, RESCUE pur-

A minimalistic communication protocol greatly reduces the attack surface.

sues a twofold mitigation technique. First, certificates are used to cryptographically bind users to network identifiers, which hinders the adversary from assuming multiple identities. Since traditional static certificate infrastructures may be unavailable in the disaster area, we propose a novel decentralized approach that enables new users to obtain certificates from mobile authorities during the disaster (Section 9.3). Second, RESCUE applies a novel buffer management scheme (Section 9.4) that substantially increases message delivery rates, i. e., availability, in the presence of flooding attacks, both from individuals and multiple identities (Section 9.5). As our solution relies on node-local decisions rather than a (complex) distributed protocol, it provides a *minimal surface to attacks* and adds *no network overhead* by design. In addition, instead of identifying and excluding misbehaving users from the network, our scheme provides a fair allocation of available resources to all users. Hence, RESCUE does not suffer from false positives, where a valid user is mistakenly excluded from the emergency communication system.

Mobile authorities enable users to join the network post-disaster.

9.1.1 *System Model*

COMMUNICATION MODEL We support a wide range of communication models that are reasonable during emergencies, including *one-to-one* (contact with friends or family), *many-to-many* (within task forces or departments), or *one-to-many* (emergency notification broadcasts). Due to the inherent delay of DTN-based communication and our focus on emergency communication, we consider mainly small messages, such as text and distress messages (including additional information, such as GPS location of the sender), serving a similar purpose as the classic *112* or *911* emergency call. Compared to rich media (images, voice, video), the information in text messages is more compact and, therefore, more suitable for DTN communication.

We propose a framework that accommodates different communication models.

ADVERSARY MODEL We consider an adversary who can mount network attacks and compromise network entities. Specifically, they can eavesdrop, manipulate, forge, or drop messages. Furthermore, the adversary can assume a limited number of entities, either by compromising or stealing devices or by registering multiple times in our system. Unlike the classic Dolev–Yao adversary model, our adversary controls only a part of the communication channel and a fraction of all network entities. Moreover, they cannot break cryptographic primitives or tamper with the *root authority* (see Section 9.3.1). In Table 12, we summarize well-known attacks (see Section 2.2) that the adversary can mount, and list RESCUE's countermeasures to prevent them.

NODE CAPABILITIES We require that a RESCUE node has (1) a neighbor communication interface, e. g., Wi-Fi or Bluetooth, (2) has

ATTACK	COUNTERMEASURE	SECTION
Route spoofing Message dropping Blackholing	Epidemic routing	9.2.1
Message spoofing Impersonation	Authentic immutable messages	9.2.2
ACK flooding	ACKs only for known messages	9.2.3
Sybil attack	User registration	9.3.3
	Priority sets	9.5.3
Message flooding	Source-based elastic buckets	9.4.2
TTL spoofing	Source-based elastic buckets	9.4.2

Table 12: Attack resiliency of RESCUE.

a buffer to store messages, (3) can create and validate digital signatures, and (4) has access to a monotonic clock, i. e., a clock that never decreases (otherwise our prioritization scheme in Section 9.4.3 might prefer an old message over a new one from the same node).

9.2 MINIMALISTIC COMMUNICATION PROTOCOL

This section describes RESCUE's communication protocol, i. e., its routing protocol, message format, and acknowledgment processing. By employing a simple routing mechanism and a minimalistic frame format, RESCUE is immune to a large set of common attacks on DTN protocols (see Table 12).

9.2.1 *Epidemic Routing*

Epidemic routing is immune to route spoofing attacks.

Instead of relying on infrastructure, DTN-enabled devices exchange messages directly using Wi-Fi or Bluetooth. DTNs exploit user mobility to increase coverage. To this end, devices act as "data mules" that store their messages as well as messages from other users, carry them, and finally forward them to the destination upon contact. We use *epidemic* routing [203] because there are no routing control messages, thus, mitigating all types of routing manipulation attacks. In addition, message dropping attacks have no effect, since messages are replicated to all available neighbors. Carried messages are stored in the node's *buffer* that we protect against flooding and Sybil attacks, as detailed in Sections 9.4 and 9.5, respectively.

Prioritization is important to use short contact times effectively.

When two devices discover each other via Bluetooth or Wi-Fi beacon frames, they initiate an authenticated handshake. As part of the handshake, both devices first exchange metadata about carried messages and then start transferring messages that the other device is missing.

However, due to limited buffer capacity and short contact times (e. g., two cars passing each other), not all messages might be exchanged. A message prioritization scheme (Section 9.4.3) determines which messages are exchanged first upon contact.

9.2.2 *Authentic Immutable Messages*

A node's public key also acts as its network identifier.

Each user possesses a unique *signature key pair* generated during initialization. The public signature key serves as a unique addressable *network identifier*. When emitting a message, the source signs the message content with its private signature key and appends the signature σ_s, source network identifier s, and, if available, the identity certificate $\mathcal{C}$ to the message. Identity certificates can be cached and only transmitted on demand to reduce overhead. Devices verify messages at each hop by checking the message signature and, if available, the source's identity certificate. Devices discard messages if a check fails so that corrupted messages do not propagate in the network. In addition, a message (MSG) contains the destination network identifier d, the creation timestamp t, the message *lifetime* Δt, and an optionally encrypted *payload* $\mathcal{P}$, resulting in the tuple:

$$\text{MSG} = (s, d, t, \Delta t, \mathcal{P}, \sigma_s, \mathcal{C})\,. \tag{45}$$

Having only immutable fields prevents all types of message spoofing.

We further define the message ID m as a hash over all message fields: $m = \mathtt{hash}(s, d, t, \Delta t, \mathcal{P})$. The signature σ_s is then calculated on m. We note that all header fields are *immutable*, i. e., they are not changed in transit, which would be required for time-to-live (TTL) fields. Immutable fields allow the signature to protect the entire message and, thus, they prevent all types of spoofing such as message modification and impersonation attacks. Assuming that the clocks of all valid nodes are roughly synchronized, the TTL of MSG can be locally computed by each node with:

$$\text{TTL} = t + \Delta t - t_{now}, \tag{46}$$

where t_{now} is the current time. Nodes regularly remove *expired* messages (negative TTL) from their buffers.

9.2.3 *Authentic Acknowledgments*

RESCUE uses acknowledgments (ACKs) for *one-to-one* communication. Previous work [203] has shown that epidemic routing greatly benefits from ACKs since they free up buffer capacity for other messages. Upon receiving a message, the destination creates an ACK as a reply and forwards it with the same mechanism used for relaying regular messages. The ACK contains only the message ID m and a signature from the destination σ_d:

$$\text{ACK} = (m, \sigma_d)\,. \tag{47}$$

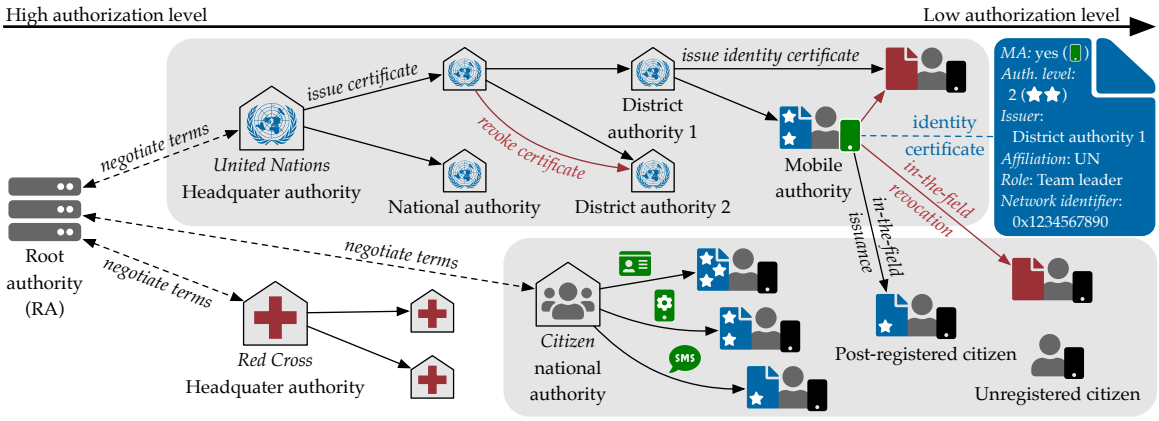


Figure 46: Illustration of our mobile distributed certificate infrastructure. The authorization level decreases from left to right. Registered users have a higher authorization level than unregistered users. Citizens are depicted in a box with a separate authorization level.

Upon receiving and verifying an ACK, intermediate nodes can safely remove the acknowledged message payload from their buffers. The ACK is stored until the corresponding message has expired. Attackers cannot forge ACKs, since they are cryptographically signed and, hence, cannot purge valid undelivered messages from the network. Since ACKs are small, they present a potential attack vector: by creating a large number of bogus ACKs, an attacker can exhaust the computational resources of the receiving nodes, because they have to verify each signature, leading to DoS. To solve this problem, nodes in RESCUE only accept and process ACKs for messages they currently carry. This stops the spreading of bogus ACKs at the first valid node.

Nodes accept ACKs only for known messages.

9.3 IN-THE-FIELD USER REGISTRATION

To establish trust relationships, we deploy so-called *identity certificates* that bind important properties in the emergency context (e. g., user role or affiliation) to the network identifier of users. In this section, we first describe our backbone certificate infrastructure. Next, we extend the backbone infrastructure by *mobile authorities*, enabling their operation during disasters in the field, where the backbone infrastructure might be unavailable. Finally, we propose multiple user identity verification methods that hamper fake registrations with the certificate infrastructure. This way, an adversary is prevented from obtaining multiple identities, i. e., certified network identifiers, that could be used to perform distributed DoS attacks. Figure 46 illustrates our certificate infrastructure.

Our certificate infrastructure enables both pre-disaster and in-the-field user registrations.

9.3.1 *Static Authorities*

Each organization can have its own CA.

The backbone certificate infrastructure consists of multiple hierarchically organized static certificate authorities (CAs). The root of these CAs constitutes a dedicated authority named *root authority (RA)* that serves as a trust anchor, and its certificate is pre-deployed on all RESCUE-enabled devices. Before the actual disaster, the RA establishes relationships with organizations or governments that want to participate as *authorities* in the certificate infrastructure. All authorities initially undergo a rigorous audit, since their authenticity and trustworthiness are crucial to the overall security, and negotiate *user roles* as well as pre-configured *user groups* that the authority introduces to the network. For instance, in Figure 46, the United Nations (UN) added the user roles *team leader* and *official* and arranged a pre-configured user group *United Nations*, so users can specifically address all UN members when sending a message. Organizations manage their own certificate infrastructure and, therefore, maintain one or multiple, potentially hierarchically organized, CAs. On the lowest hierarchical CA level, CAs issue identity certificates to staff members. Furthermore, the overall infrastructure contains at least one authority, e. g., from the national government, that issues identity certificates to regular users, i. e., citizens.

Identity certificates bind the public signing keys of users, which function as their unique network identifiers (see Section 9.2), to user properties. Important properties in the emergency setting are the *affiliation* (e. g., UN, red cross), *user role* (e. g., citizen, physician), and *authorization level*, which indicate a user's permission level and trustworthiness. Figure 46 exemplifies the identity certificate of a UN team leader, and depicts the authorization level of entities by their position on the x-axis as well as stars in the certificate.

A revoked certificate deprives the owner of any role and privileges.

We further consider certificate revocation, since an adversary may obtain identity certificates, compromise user devices, or even infiltrate authorities. RESCUE implements certificate revocation via certificate revocation lists (CRLs) that authorities broadcast in the network. We distinguish between two different entities: authorities and users. An authority $\mathcal{A}$ can revoke an entity $\mathcal{E}$ if $\mathcal{A}$ has a higher authorization level than $\mathcal{E}$, and there is a certificate chain (i. e., a chain of trust) between $\mathcal{A}$ and $\mathcal{E}$. Upon the revocation of an authority, all certificates issued by the authority are regarded as invalid, depriving it of its power. In case a user identity certificate is revoked, the certificate is considered invalid, and the respective user loses his or her role, authorization level, and any message transmission privileges (see Section 9.5).

9.3.2 *Mobile Authorities*

In a disaster area, infrastructure-based communication is mostly unavailable and, thus, users rarely have a connection to the static authorities in the backbone infrastructure. This is not an issue when established user properties are retrieved since identity certificates can be verified and transmitted between users on demand (see Section 9.2). Nevertheless, operations that inevitably involve CAs, such as issuing new certificates or revoking existing certificates, cannot be performed when static authorities are unavailable.

Therefore, we propose that specially privileged users employ their mobile devices to serve as *mobile authorities (MAs)* during a disaster. Since it is easier for an adversary to compromise MAs than static authorities, MAs have restricted capabilities. In detail, they can only issue identity certificates for citizens, but not for specific user groups like red cross staff members. We argue that this is not a restriction since professional emergency workers typically set up their systems before the disaster or outside the disaster area, where a connection to static authorities is available. Additionally, MAs are allowed to revoke identity certificates. An MA $\mathcal{M}$ can revoke a user $\mathcal{U}$, if $\mathcal{U}$ is a citizen, or if $\mathcal{U}$ has a lower authorization level than $\mathcal{M}$ and both belong to the same affiliation (Figure 46).

Mobile authorities can issue and revoke certificates in the field.

Since malicious MAs can seriously harm the network, only privileged and trustworthy users with devices that satisfy certain security requirements are permitted to become MAs. In the initialization phase, each organization negotiates the maximum permitted number of MAs they introduce to the network, and then carefully selects those users qualifying to become MAs. MA users have a high social trust level and protect their devices using security mechanisms like a trusted execution environment, full disk encryption, and strong passwords. These mechanisms have shown to significantly increase the effort for physical attacks [165], giving MA users enough time to report and revoke stolen MA devices.

MA devices should be resistant to physical attacks.

9.3.3 *Secure Identity Verification Methods*

In the following, we present methods that enable static and mobile authorities to identify registering entities based on hard-to-forge *identification tokens*. The methods hinder individuals from registering repeatedly and obtaining multiple identity certificates. Our goal is to increase the cost for fake registrations, such that bypassing our subsequently presented flooding and Sybil mitigations (see Sections 9.4 and 9.5) becomes uneconomical for an adversary. We assume that organizations and governments are already able to supply each staff member with exactly one identity certificate, e. g., by handing out pre-configured devices. Therefore, we focus on fake registrations of

Secure identity verification methods make it harder for an attacker to register under multiple identities.

users with the authorization level *citizen*. Citizens that employ stronger authentication methods during registration with CAs are considered more trustworthy, indicated by a higher authorization level in their identity certificate. Messages from users with high authorization levels are transmitted preferentially (see Section 9.4) to encourage citizens to (1) obtain identity certificates and (2) use strong identification methods during registration. Typically, as summarized in Table 13, the stronger the identity proof is, the more restrictive (i. e., less applicable) and time-consuming the verification process gets, resulting in limited practical usability. Authorities maintain a shared database that stores all registered users and their identification token to prevent an individual from registering at multiple authorities. Since MAs are located in the disaster area and thus cannot access this database, MAs inform each other about users that registered during the disaster by using the proposed communication system. Since communication during a disaster is typically delayed, an adversary might be able to register with the same identification token at multiple MAs, namely those that have not yet exchanged information about registered users. When, however, the affected MAs eventually communicate and find out that the same identification token is used multiple times, the corresponding certificate is immediately revoked by them.

The strength of the identity verification method can be used to derive the authorization level.

IDENTIFICATION VIA PHYSICAL PRESENCE This method constitutes a fallback solution only used by MAs during a disaster if all other verification methods are inapplicable. It does not rely on an identification token but instead requires a user to physically approach an authority during registration and thus expend physical effort. MAs assert the physical proximity of users by employing short-range communication channels (e. g., QR codes, near-field communication (NFC), or Bluetooth) to transmit identity certificates. Furthermore, MAs manually confirm the issuance of each identity certificate to prevent an adversary from obtaining multiple certificates at once. The method has a weak identification strength since an attacker can simply approach different MAs or, with some delay, the same MA repeatedly. Also, usability is poor, since users cannot register remotely.

Registration via physical presence is a fallback solution and provides only poor identity verification.

IDENTIFICATION VIA SIM The SIM card is used as an identification token by requiring the user to enter a nonce that is sent via a call or SMS to the user's device during registration. Authorities prevent multi-registrations by allowing each phone number to belong only to one network identifier. The approach provides excellent usability and applicability since it requires the user to take a minimum effort, and SIM cards are available in many mobile devices. Nevertheless, an adversary can create fake users by using anonymous prepaid SIM cards. Also, since the approach requires a functioning cellular network, it is unsuitable for registering new users during a disaster.

SIM-based registration is only possible if infrastructure is (still) available.

METHOD	STRENGTH	USABILITY	APPLICABILITY	AVAILABLE IN CRISIS
Physical Presence	•	••	•••••	✓
SIM	•••	•••••	••••	✗
Remote Attestation	••••	•••••	•••	(✓)
eID	•••••	•••	••	(✓)

Table 13: Comparison of identity verification methods. Ratings scale from poor (•) to excellent (•••••). "(✓)" indicates the lack of an off-the-shelf implementation for evaluation.

IDENTIFICATION VIA REMOTE ATTESTATION This method employs the mobile device itself as an identification token, as authorities perform a remote attestation [4] with devices of registering users. This way, authorities obtain an attestation report that is signed with a device-unique secret attestation key and a certificate that testifies the validity of the attestation key. Authorities prevent multi-registrations by storing the public attestation key, requiring the adversary to possess one device per fake user. To date, applicability is good, as recent Samsung [157], Windows [131], and Android [4] devices provide remote attestation capabilities. Additionally, remote attestation will become increasingly widespread with upcoming technologies [178, 196], and MAs could act as verifiers and thus identify new users during a disaster. In practice, though, remote attestation without backbone access is not yet implemented.

IDENTIFICATION VIA EIDS This method uses national electronic ID cards, which often provide identification capabilities, as identification tokens. As an example, the eIDAS regulation defines electronic identification services in the entire European Union (EU) [58]. eIDAS specifies the restricted identification (RI) protocol, e. g., implemented in the German identity card since 2010. RI allows a service provider (SP) terminal to recognize an eID chip based on a chip-unique pseudonym. By using mobile devices as local terminals [191], authorities can act as SP terminals and securely identify eID cards of registering users. The approach provides a strong proof of identity as it is hard to forge eID cards or to obtain multiple valid eID cards, including their PIN. Since MAs can, in principle, act as SP terminals, the approach is applicable during a disaster. As a downside, users must initially activate their eID cards and have them at hand.

National electronic ID cards provide a strong proof of identity.

9.4 LOCAL BUFFER MANAGEMENT

Buffer management decides which messages to keep and which ones to drop.

Within the *buffer*, a node stores unacknowledged and unexpired messages. If there are many such messages, a node might not have the resources to store them all: at some point, it needs to decide which messages to keep and which ones to drop. Given a set of messages and a buffer with a node-defined *capacity* C, the *buffer management* has to decide which messages to store in the buffer without exceeding its capacity to enforce:

$$\sum_{MSG} |MSG| \leqslant C. \tag{48}$$

We optimize buffer management for DoS resiliency.

Besides this hard constraint, buffer management can have multiple optimization goals, e. g., throughput maximization, delay minimization, or delivery reliability. In this work, we are concerned with security, in particular, resiliency against DoS attacks. To this end, we first motivate the need for security mechanisms in the buffer management, define security requirements, and then present a novel secure buffer management strategy which achieves protection against flooding attacks.

9.4.1 *Security Requirements and Design*

FIFO is does not perform well as a buffer management strategy.

Poor buffer management schemes can expose a network to flooding attacks. For example, malicious nodes can exploit trivial schemes such as first-in first-out (FIFO) queues to replace valid messages with bogus ones [116]. Our buffer management scheme has the following goals: (1) be fair, i. e., ensure that a single attacker can occupy at most $1/n$ of available buffer space; (2) maximize buffer utilization to increase message redundancy and, thus, the delivery rate; and (3) prevent that a single MA compromise can compromise the entire network (such Sybil attacks are discussed in Section 9.5). To reach these goals, we apply a *locality* principle [64] to allow nodes to decide locally and independently which messages to store. Hence, nodes do not need to trust and verify third-party information, which keeps the attack surface small. Furthermore, bandwidth efficiency increases, since control messages need not be exchanged. The locality principle requires us to reverse the role of making the replication decision. Classically, the carrying node decides whether or not to replicate a message to another node based on some utility function, e.ġ., [22]. However, since nodes should make all decisions themselves, they should also decide which messages to accept or drop. Therefore, upon contact, the receiving node selects the messages that it wants to receive.

9.4.2 *Source-Based Elastic Buckets*

We now present our novel buffer management strategy *source-based elastic buckets (SEB)* that, by design, prevents valid messages from being purged from the network during flooding attacks. The basic idea is that all messages from a source s are placed in an isolated bucket $B(s)$. Since RESCUE ensures message authenticity through digital signatures, an adversary cannot forge messages in a way that they occupy buckets of valid users. SEB is fair in the sense that each bucket $B(s)$ has a *guaranteed capacity* of $C_n = C/n$, with n being the number of currently allocated buckets (number of source nodes that a respective node currently carries messages from). The *occupancy* of a single source bucket $O(s)$ is a non-negative number and is subject to

SEB isolates messages from different sources.

$$\sum_s O(s) \leqslant \sum_s C_n = C. \tag{49}$$

We further define the surplus $S(s)$ as the (possibly negative) difference between the occupancy and the guaranteed capacity:

$$S(s) = C_n - O(s). \tag{50}$$

If s does not exhaust its guaranteed capacity ($S(s) > 0$), because it has not sent "enough" messages, $S(s)$ is provided to other buckets requiring it, which means that their surplus can become negative. However, when s sends more messages at a later point, overdrawn buckets ($S(s) < 0$) are emptied first. These *elastic quotas* allow full exploitation of the local buffer capacity while maintaining strict message separation of different sources. Program 3 shows SEB's message insertion procedure: the underlying idea is that SEB inserts a new message in the corresponding source bucket and then drops messages from the bucket with the smallest surplus[1] until the total occupancy meets C (see Equation (48)). When a bucket becomes empty, SEB deallocates it and shares the freed capacity among the remaining buckets.

With elastic quotas, a node can get more than its guaranteed capacity to prevent resource underutilization.

SEB's robustness relies on the fact that messages are source-authenticated, and on the high costs of registering multiple identities in RESCUE. Without the latter costs, an attacker could assume multiple identities, flood the network with messages and, thus, hijack a disproportional amount of buffer capacity. In addition, SEB mitigates TTL spoofing attacks where an attacker sets excessively high values for Δt to maximize the lifetime of its messages: by separating messages of different sources, an attacker would only be able to replace own messages.

Isolated buckets effectively mitigate TTL spoofing attacks.

1 Ties are broken at random. In order to assert that the buffer converges to a stable state, tie-breaking needs to be consistent, i. e., the same tie needs to be broken consistently at a single node. We implement this by comparing the salted hashes of node IDs, while the salt is drawn at random once by each node. A tie is then broken by the smaller hash value.

```
Input: MSG, buckets, C
s ← source of MSG
if not buckets contains B(s) then
    insert empty B(s) in buckets;
end if
B(s) ← bucket from buckets for s;
insert MSG in B(s);
while ∑_i O(i) > C do
    B̂ ← bucket from buckets for arg min_{s'} S(s');
    M̂SG ← message with the lowest rank in B̂;
    drop M̂SG;
    if B̂ is empty then
        drop B̂ from buckets;
    end if
end while
```

Program 3: Message insertion with SEB.

9.4.3 *Prioritization and Convergence*

Message rank generates a deterministic and reproducible message ordering.

Within each bucket, SEB uses *message rank (MR)* for prioritization. MR prioritizes (1) acknowledgments, (2) messages with the largest TTL, and (3) messages with the smallest payload size. Carrying messages with a large TTL increases the probability that they will be delivered before expiration (we confirm this in Section 9.6), while small messages take less time for transmission, and help to prevent buffer fragmentation. Upon device contact, messages exchanged first have a higher chance of actually being transmitted to the next hop and eventually reaching their destination. A sending node transfers messages in its buffer to a receiving node d in the following order: (1) messages destined for d, (2) own messages, (3) messages from other registered users, (4) all other messages. *MR* sorts messages in each category. MR only relies on fields in the message header. Since they are immutable, MR calculates the same ordering independent of the order in which messages were received, which means that the buffers of two nodes will converge to a stable state if the contact duration is long enough. This is a problem that has not been properly addressed by the research community, which is reflected by the fact that the most popular network simulator for DTN research, *ONE* [104], only implements non-converging random and FIFO-based dropping strategies.

9.5 LOCAL PRIORITY SETS

Until now, we have assumed that MAs behave correctly and cannot be compromised by an adversary. We recognize that this is a strong assumption, since MA devices may be stolen or infected with mal-

LEVEL	CONTAINED NODES	PURPOSE	SECTION
0	Local (own)	DD lower bound	9.5.1
1	Social network/by MA	Sybil protection	9.5.3
2	Other registered	Flooding protection	9.4.2
3	Unregistered	Best effort	9.5.4

Table 14: All priority sets used in RESCUE.

We extend SEB to provide protection against a Sybil attacker that has compromised an MA.

ware. However, if we lift our assumption on secure MAs, our buffer management presented in Section 9.4 is vulnerable to Sybil attacks. This is because an adversary that gets hold of an MA can generate as many certificates as it wants. Since SEB allocates buffer resources fairly among all nodes, a *Sybil* attacker would receive an unfair amount of buffer space. In this section, we secure RESCUE even against such Sybil attackers by leveraging the concept of *secure message copies* using *priority sets* where nodes prioritize messages originating from a certain set of other nodes in the network. In the following, we first introduce the concept of *secure copies*, explain the workings of *priority sets*, and discuss Sybil-secure fill strategies. Finally, we explain how priority sets also help to securely support unregistered users.

9.5.1 *Secure Copies*

We leverage the fact that direct delivery forwarding is inherently immune to Sybil attacks.

In direct delivery (DD) forwarding, nodes do not carry messages for others, but only deliver them when they encounter the destination. In other words, the hop count of each delivered message is precisely one. This diminishes the advantage of having "data mules." However, DD has a desirable security property: *it is inherently immune to flooding attacks even from Sybil attackers* since it simply does not carry messages for other nodes. In other words, DD ensures that there is always one copy of every message in the network, namely in the buffer of the source node. We call this a *secure copy*, i. e., a message copy that an attacker cannot remove or replace. In the case of DD, the number of secure copies per message is exactly one. Though this is a straightforward strategy, other buffer management schemes fail to achieve this guarantee. For example, when using FIFO, own messages might be replaced by more recently received messages of other nodes. In the following, we increase the number of secure copies to improve delivery reliability.

9.5.2 *Priority Sets Overview*

We adopt the concept of *secure copies* to solve the problem of Sybil attacks while still accepting messages from other nodes using *priority sets (PSs)*. PS includes the identities of other nodes that a node is

We extend SEB with priority sets that group buckets of a particular set of source nodes.

willing to prioritize messages for. How a node determines the content of a PS is crucial for the security of the system. Making individual local decisions makes it hard for an attacker to appear in all PS, thus preventing that their messages fill the buffers of all other nodes. By using PS, each node essentially gains several *secure relays* that prioritize messages for it, effectively increasing the number of *secure copies*. Next, we show how PS integrates with SEB (Section 9.4), and discuss how PS can be used to protect against Sybil attacks in Section 9.5.3. To integrate PS in SEB, we need to ensure that buckets of nodes in a PS are emptied last. We generalize this approach by allowing an arbitrary number of PS *levels* l. For the remainder of this work, we denote $\mathcal{S}_l$ as the priority set at level l. The set with the numerically *lowest level* has the *highest priority*. We adapt line 8 in Program 3 to first select the PS with the largest l that is not empty. We then choose the bucket with the smallest surplus from only this set:

8a: $\hat{l} \leftarrow \arg\max_l \{\mathcal{S}_l, |\mathcal{S}_l| > 0\}$;

8b: $\hat{B} \leftarrow B\left(\arg\min_{s \in \mathcal{S}_{\hat{l}}} S(s)\right)$;

To achieve the performance of DD (at least one secure copy per message), $\mathcal{S}_0$ only includes the local node, such that messages of the local node are removed last. We explain more PS levels in the following sections, and we summarize all PS levels used in RESCUE in Table 14.

9.5.3 *A Sybil-Secure Priority Set*

We discuss options for a Sybil-secure priority set.

How to select the nodes that are to be put in $\mathcal{S}_1$ is key to achieving protection against Sybil attacks. The selection strategy needs to ensure that the nodes residing within each set are (preferably) different for each node, and it is hard for the Sybil attacker to become a member of many of those sets. In the following, we discuss two PS fill strategies that are secure against Sybil attacks and can be practically used in emergency scenarios. The members of this set are selected either (1) by exploiting social relationships between the nodes, e. g., by leveraging phone numbers in the address book of the users' smartphones or (2) by letting the MA assign the set during registration. We now present both approaches in detail.

PRE-REGISTRATION: SOCIAL NETWORKS Social networks have been used to detect Sybil identities effectively [198]. While the particular method [198] is unpractical in DTNs (it needs to perform online verifications), we borrow the idea of using social networks as a defense against Sybil attacks. We exploit the user's social network to prioritize messages from direct neighbors in the user's social graph, e. g., those nodes that are in the local node's address book. Since the attacker has no control over uncompromised devices, they cannot forcibly add themselves to others' address books. Thus, they will not be able to

We rely on the user's address book to fill $\mathcal{S}_1$.

appear in $\mathcal{S}_1$ of legitimate nodes. This option is only available for users that have registered with an authority before the disaster, e. g., using their SIM card (Table 13), and have resolved the phone number to the RESCUE public key via a central server similar to secure messaging applications (e. g., Signal). For all users that registered during a disaster with an MA, a different method is required.

POST-REGISTRATION: MA-ASSIGNED We assume that most users will only register post-disaster and, thus, cannot use social contacts to fill their PS. However, we can leverage the trustworthiness of MAs by letting MAs suggest identities for the priority set during registration. We propose that the suggested identities are those that recently registered with the MA. Since an attacker is not able to manipulate the PS of nodes that registered with the MA *before* being compromised, only nodes that registered *after* an MA compromise may be affected by PS manipulations.[2] The latter may experience decreased service quality since their identities will not appear in the PS of other nodes. This effectively reduces the number of secure copies for their messages to one, which is the same for other unregistered users. Other than that, they are not affected negatively by registering with a compromised MA.

The MA suggests identities for $\mathcal{S}_1$ during registration.

For both strategies, we need to determine the size of $\mathcal{S}_1$ (the number of contacts in the users' address books or the size of the MA-suggested list) that will effectively thwart Sybil attackers. In Section 9.6, we empirically show that already a small size is sufficient to withstand Sybil attacks.

9.5.4 *Supporting Unregistered Users*

Apart from Sybil attacks, PS enables us to solve another remaining problem: supporting unregistered nodes without compromising the security of registered users. To this end, we introduce another PS with a priority level higher than the one used in Section 9.5.3. This essentially assigns all remaining buffer capacity to unregistered nodes. These nodes will consequently receive the lowest quality-of-service level since their messages will be dropped first. However, we show later that in case the network is not fully congested (i. e., not during a flooding attack), unregistered users receive a quality-of-service level similar to that of registered users. Even when the network is under a flooding attack, the performance never drops below the DD lower bound.

All unregistered nodes are assigned to the lowest priority set.

2 Using this method, an adversary may be present in $\mathcal{S}_1$ of legitimate nodes by registering at an MA during disasters. However, the adversary may only register a few identities this way, since it is hard to register multiple times at an MA (see Section 9.3.3).

Scenario	*Synthetic*	*2013 Typhoon Haiyan*
Mobility	Map RWP (*Helsinki*)	*Natural Disaster*
Speed	90 % ped., 10 % car	100 % pedestrian
Duration	12 h (+5 h cooldown)	168 h = 7 days
Dimensions	4500 × 3400 m²	5000 × 7000 m²
Total Nodes	1000	500 (7 roles)
Unregistered	—	40 (injured citizens)
Post-registered	800	200 (healthy citizens)
Pre-registered	200	260 (all others)
Message Rate R_n	0.1 s^{-1} (10 s interval)	0.1 s^{-1} (10 s interval)
Message Size	25 KB	25 KB
Message Lifetime	5 h	12 h
Buffer Capacity	5 MB	20 MB
Buffer Management	*FIFO*, *MR*, *SEB*, *PS*	
Routing	*Epidemic*, *DD*	
Radio Link	Bluetooth (2 Mbit/s at 10 m range)	

Table 15: Simulation settings for the ONE.

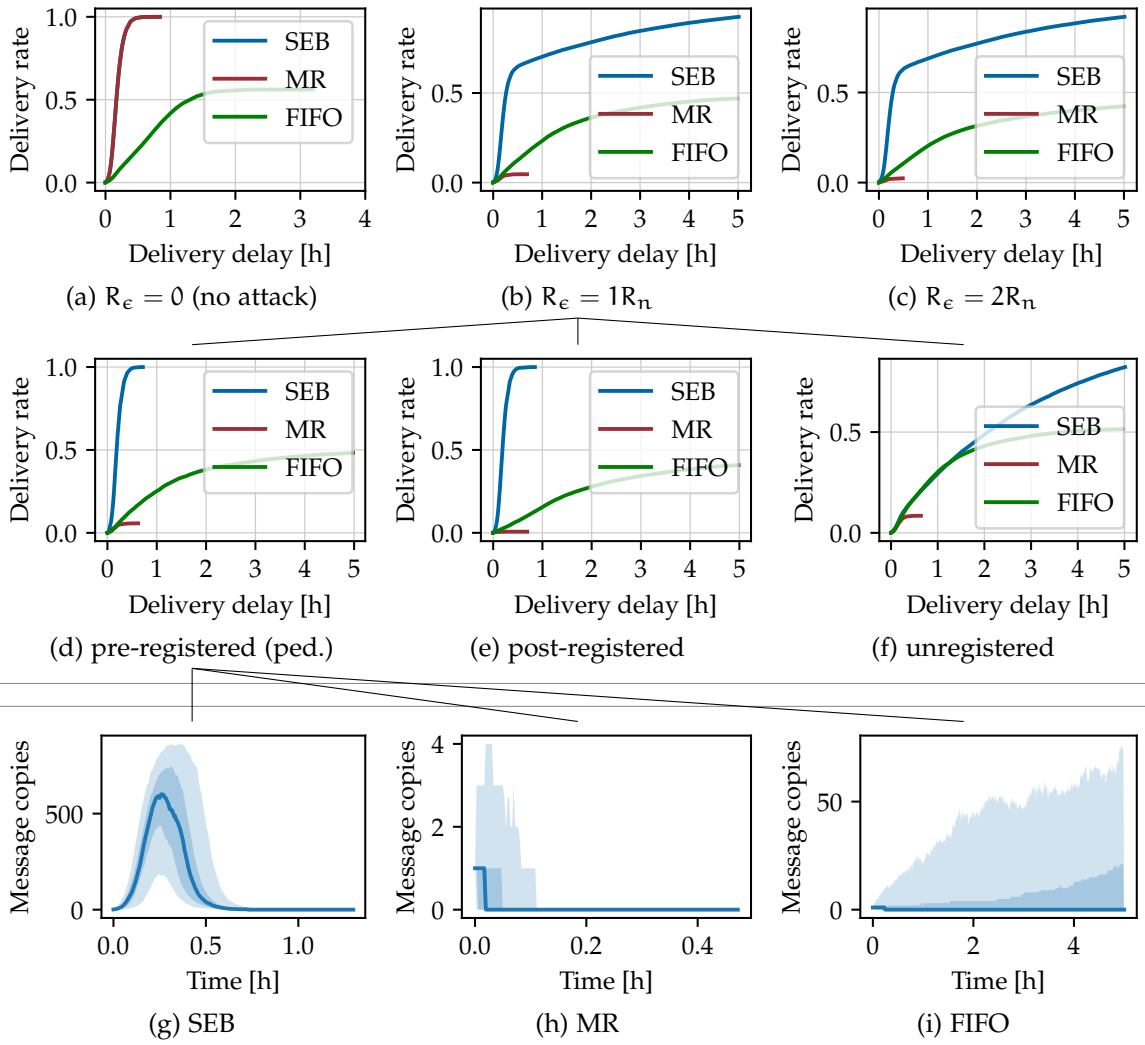
9.6 EXPERIMENTAL EVALUATION

In this section, we evaluate the behavior of our security mechanisms in large networks using simulation. We first describe our evaluation scenario and present the performance results of RESCUE under flooding and Sybil attacks. Finally, we repeat the experiments under an accurate non-synthetic mobility model for large-scale natural disasters.

9.6.1 *Test Setup*

We use a synthetic and a realistic disaster scenario in our evaluation.

We consider two scenarios as detailed in Table 15: (1) A *synthetic* scenario to isolate the effect of two distinct attacks on the network (Sections 9.6.2 and 9.6.3) and (2) the *Typhoon Haiyan* scenario to assess performance under more realistic conditions (Section 9.6.4). In both scenarios, we consider three different node classes: *pre-registered*, *post-registered*, and *unregistered*. All *post-registered* nodes are unregistered at the start of the simulation and become registered until the end of the simulation linearly over time. All nodes listed as *unregistered* in Table 15 remain unregistered. For simplicity, we only evaluate two *authorization levels* (Section 9.3): registered and unregistered. In the *Typhoon Haiyan* scenario, we have additional *roles* such as injured and healthy citizens, urban search and rescue team (USRT), and UN officials, which all have distinct mobility patterns (see Chapter 3). We



Delivery rate over the **delivery delay** for different attacker injection rates R_ϵ (*first row*) and individual user groups at $R_\epsilon = 1R_n$ (*second row*).

Message copies over time of the *pre-registered (pedestrian)* node group at $R_\epsilon = 1R_n$ for simulation run 1. The center line shows the median, the shaded areas the 10th, 30th, 70th, and 90th percentiles.

Figure 47: Performance during a flooding attack showing delivered messages, delivery delay, and message copies over time.

use epidemic routing and compare six different buffer management strategies:

We compare six different buffer management strategies.

- *FIFO* which uses a first-in-first-out queue for prioritization and as a drop strategy,
- *MR* which uses message rank instead of a FIFO queue,
- *SEB* which employs our source-based elastic buckets and uses all priority set levels except for $\mathcal{S}_1$,
- *PS* which uses *all* priority set levels,
- *PSo* which only uses the priority sets $\mathcal{S}_0$ and $\mathcal{S}_1$, and
- *DD* which uses direct delivery and effectively only uses $\mathcal{S}_0$.

We evaluate the latter two only for the Sybil attack scenarios. We choose a small buffer capacity to exaggerate the effect of the different buffer management strategies. The message size is fixed to avoid fragmentation effects in the buffers. We chose the simulation parameters in the *Typhoon Haiyan* scenario following [175] for comparison reasons. We use the *ONE* simulator v1.6.0 [104] for our experiments and, unless stated otherwise, show the average over ten runs with different seeds.

9.6.2 *Flooding Attack*

We express the message injection rate as an aggregate over all attackers.

We first evaluate the resiliency of different buffer management strategies against a small number (5 %) of flooding attackers injecting bogus messages at a high rate in the *synthetic* scenario. The attackers maximize the message lifetime of their messages. They address their messages to nonexistent destinations so that acknowledgments are never returned, and they set the message lifetime to a value that is larger than the simulation time to keep their messages persistent unless the buffer management drops them. Valid users choose the destination randomly among all other valid users. Figure 47 shows the overall delivery rate and delay for different attacker injection rates R_ϵ as a function of the valid users' injection rate R_n. Note that R_ϵ and R_n are aggregate rates, e. g., $R_\epsilon = 1R_n$ means that all attackers inject as many messages as all valid users combined. To better understand the results, Figure 47 differentiates between registered and unregistered users, and includes the network-wide copies per message during the attack. We make multiple observations.

ACKs are important to remove stale messages from the buffers.

IMPORTANCE OF ACKNOWLEDGMENTS In a benign setting, ACKs help to keep buffers clean (SEB and MR). Once a message is delivered (Figure 47a), the number of copies in the network reduces about as quickly as they increased (Figure 47g) yielding perfect (i. e., 100 %) delivery rates.

FIFO VS. MR Using FIFO as a buffer management strategy does not yield satisfactory results even in a benign scenario because buffer

states do not converge: FIFO always accepts an incoming message even if it has previously been dropped. The performance further decreases as the attackers' injection rate increases (Figure 47c). On the other hand, prioritizing messages by deadline is a very effective metric to achieve 100 % message delivery in less than 1 hour (MR in Figure 47a). However, at the same time, this is tremendously susceptible to flooding attacks, as it uses the TTL for prioritization that the attacker manipulates by setting an arbitrarily large message lifetime, thus, reducing the message delivery rate to about 5 % (Figure 47c). This occurs since the attacker removes virtually all valid message copies in the network quickly (Figure 47h).

Our message prioritization mechanism alone improves delivery rates in a benign scenario.

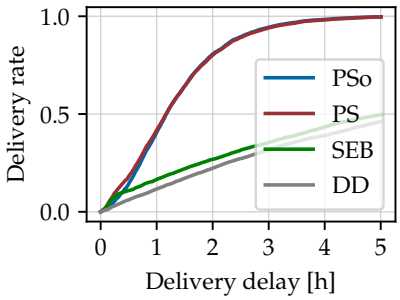
SEB MITIGATES FLOODING ATTACKS FOR ALL REGISTERED USERS SEB uses MR as a secondary metric within each bucket. Therefore, SEB achieves the same delivery rate as standalone MR under no attack (plots in Figure 47a overlap), but maintains a high delivery rate of more than 90 % even under the flooding attack (Figure 47c). In fact, as we vary the flooding injection rate, performance only decreases for unregistered users (Figure 47f), while all other groups (Figures 47d and 47e) *remain unaffected*. Since SEB's delivery rate did not change from $R_\epsilon = 1R_n$ to $2R_n$, we abstain from studying results when further increasing R_ϵ.

9.6.3 *Sybil Attack*

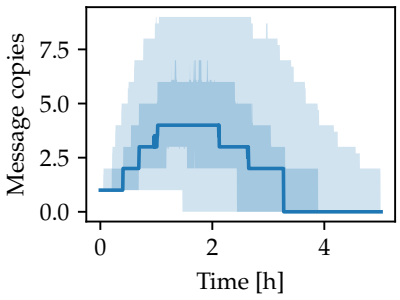
In our next experiment, we evaluate the impact of a Sybil attacker on RESCUE, again in the *synthetic* scenario. The Sybil attacker is a single node (physical position) that can generate an unlimited number of registered identities, e. g., by compromising an MA. To cause maximum harm, the attacker executes a flooding attack as in Section 9.6.2 but uses a new identity for each injected message. We compare the results of *DD*, *SEB*, and *PS*. To isolate the effect of our Sybil-secure priority set, we include an additional strategy *PSo* which only uses $\mathcal{S}_0$ and $\mathcal{S}_1$ (i. e., this strategy does not relay messages for unregistered nodes and registered nodes that are not in the Sybil-secure priority set). We omit the results for *FIFO* and *MR* since they already performed poorly under a non-Sybil attack. We set the Sybil-secure priority set size $|\mathcal{S}_1|$ to 10. For the pre-registered nodes, the sets are generated from a fixed-degree random graph, while the sets of the post-registered nodes are determined by the $|\mathcal{S}_1|$ most recently post-registered nodes. For space reasons, Figure 48 shows only the results for the pre-registered and unregistered group.

With the PSo *buffer management strategy, we isolate the effect of the Sybil-secure priority set $\mathcal{S}_1$.*

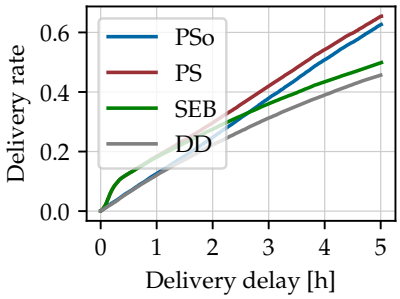
SYBIL ATTACKER REPLACES ALL NON-SECURE COPIES In Figure 48a, we can see that the delivery rate for *SEB* significantly decreases under a Sybil attack compared to Figure 47d. With *PS*, the



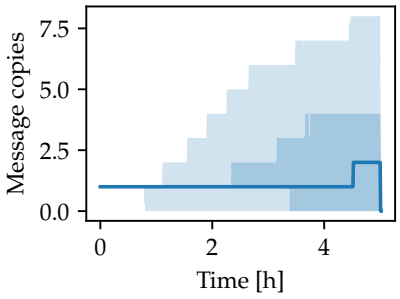
(a) Delivered messages of the *pre-registered pedestrian* group.



(b) Message copies for *PS* of the *pre-registered pedestrian* group.

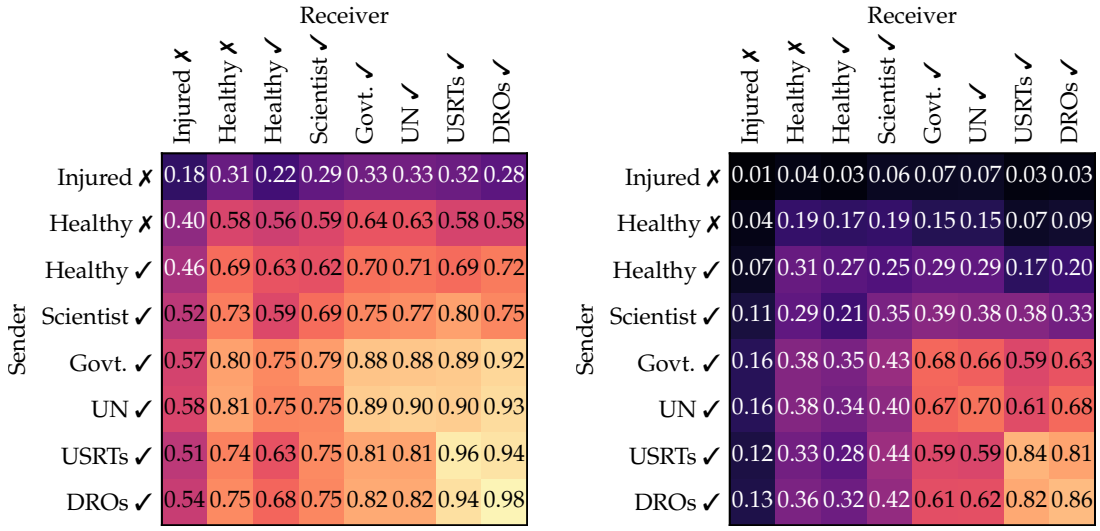


(c) Delivered messages of the *unregistered* group.



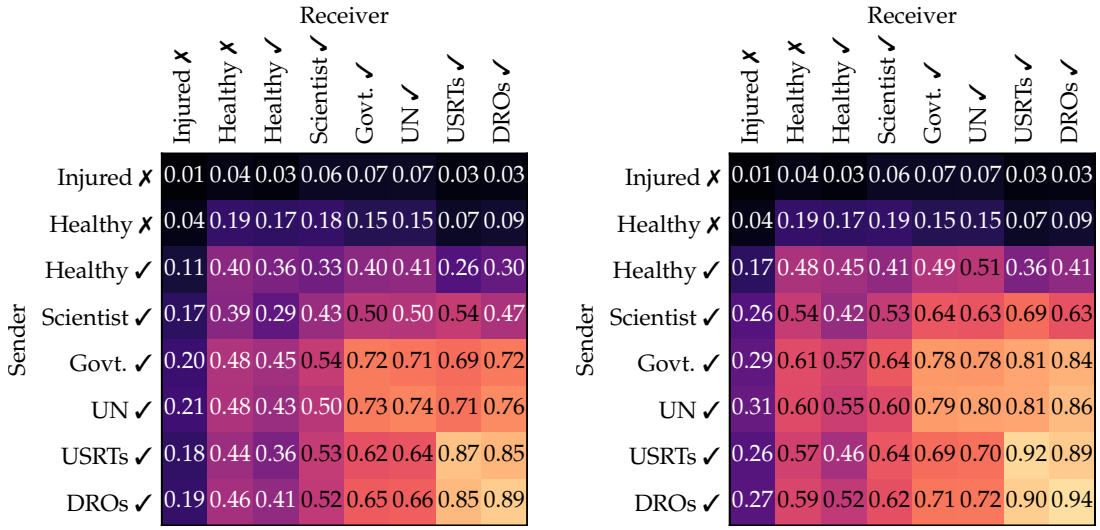
(d) Message copies for *PS* of the *unregistered* group.

Figure 48: Impact of Sybil attack ($R_\epsilon = 1R_n$) on the pre-registered and unregistered groups. Plots show the delivered messages over the delivery delay and message lifetime.



(a) No attack.

(b) Under attack and $|\mathcal{S}_1| = 10$.



(c) Under attack and $|\mathcal{S}_1| = 20$.

(d) Under attack and $|\mathcal{S}_1| = 50$.

Figure 49: Typhoon Haiyan scenario without ($R_\epsilon = 0$) and with Sybil attack ($R_\epsilon = 1R_n$). Matrices show the average message delivery rates between roles using PS.

situation improves enormously. The *secure copies* in $\mathcal{S}_1$ work as intended, so that there is a small number ($\leqslant |\mathcal{S}_l|$) of message copies that propagate through the network. Conversely, this experiment also shows that all $|\mathcal{S}_l|$ with $l > 1$ are completely vulnerable to a Sybil attack: the attacker effectively replaces all non-*secure copies*, which is confirmed by the almost identical performance of *PSo* and *PS* (overlap in Figure 48a). This behavior demonstrates the *resiliency* aspect of RESCUE: even under a strong attack, RESCUE continues to operate considerably well but with reduced performance.

Only $\mathcal{S}_0$ and $\mathcal{S}_1$ are secure against Sybil attacks.

DIRECT DELIVERY DEFINES THE LOWER BOUND PERFORMANCE In Figure 48c, we show the performance that unregistered users experience. We see that the performance is significantly lower than for registered users. However, all of our buffer management strategies perform better than *DD*, asserting our claim of achieving a lower bound performance. *SEB* performs slightly better as it can take advantage of the epidemic flooding at the beginning of the simulation when the attacker's messages have not yet fully saturated the network. *PS* performs better because some of the unregistered nodes become registered while their messages traverse the network and, thus, are prioritized by other nodes. Figure 48d shows the average message copies that increase towards the end of the message lifetime.

We verify our claim of achieving a lower bound performance.

9.6.4 *2013 Typhoon Haiyan Scenario*

To evaluate how RESCUE would perform in a disaster scenario, we repeat our experiments under an accurate human mobility model for large-scale natural disasters (Chapter 3). This model features seven different node roles (e. g., citizens and professional disaster response teams (DRTs)), various points-of-interest (e. g., base camps), and time-of-day dependent activities. Depending on the group, we consider that users are registered (✓) or unregistered (✗). We evaluate RESCUE in the *Typhoon Haiyan* scenario (city of Tacloban) with a total of 500 nodes. We show the inter-group message delivery rates for *PS* in Figure 49. For space reasons, we only show the results for no attacker (Figure 49a) and for the Sybil attack with a flooding rate of $R_e = 1R_n$ but with different values for $|\mathcal{S}_1|$ (Figures 49b to 49d). The Sybil attack is the same as in Section 9.6.3, with the difference that the attacker node is chosen randomly from the disaster response organization (DRO) group since this group is the best-connected one due to high contact rates.

IMPACT OF MOBILITY MODEL In comparison to our *synthetic* scenario, the most striking difference is that we do not achieve perfect delivery rates when no attackers are present (compare Figure 47a and Figure 49a). The underlying mobility causes these differences, e. g.,

We do not achieve perfect delivery rates in a realistic scenario.

some nodes (injured citizens) do not move at all. Also, the *synthetic* scenario features fast-moving cars and a higher node density. Since cars might not be usable due to blocked roads, we focus on pedestrians only. Still, PS achieves significantly better results than FIFO. This suggests that PS can even apply to benign settings.

EFFECTIVELY NO SERVICE FOR UNREGISTERED CITIZENS Unregistered users (✗) are almost completely denied service under attack and experience a delivery rate of 19 % at best (Figures 49b to 49d). Those citizens already receive reduced service in a benign scenario (Figure 49a). The direct-delivery performance that is achieved under attack does not suffice to maintain a reasonable delivery rate. This discrepancy emphasizes the importance of user registration, especially under a Sybil attack.

Direct delivery does not achieve usable delivery rates in a realistic scenario.

IMPACT OF THE SYBIL-SECURE PRIORITY SET SIZE Under attack, delivery rates drop significantly for all groups (Figure 49b). While well-connected registered groups can still achieve inter-group delivery rates above 90 %, unregistered groups are effectively cut off from communication. The situation for registered nodes improves when we increase $|\mathcal{S}_1|$ from 10 up to 50 (Figures 49c and 49d): at $|\mathcal{S}_1| = 50$, the delivery rate is only 3–12 % lower than in a benign setting. However, as expected, increasing $|\mathcal{S}_1|$ does not affect unregistered users.

Well-connected groups achieve decent delivery rates even only with $\mathcal{S}_0$ and $\mathcal{S}_1$.

DELIVERY ASYMMETRIES We detect asymmetries in the delivery rates between certain group pairs in Figure 49a, such as the registered DROs and the unregistered injured citizens (0.54 vs. 0.28). Our PS levels can explain these asymmetries: registered users can use more *secure relays* than unregistered ones. These asymmetries might influence the design of applications building on RESCUE.

9.7 DISCUSSION AND SUMMARY

We presented RESCUE, a communication framework for resilient and secure disruption-tolerant emergency communication on mobile devices. RESCUE is the first secure emergency communication solution that allows users to join the network during disasters when infrastructure networks are unavailable by deploying mobile authorities in the field. In addition, we are the first to present a buffer management approach to mitigate flooding attacks even in the presence of Sybil attackers. In particular, our solution uses a minimalistic communication protocol and implements flooding mitigation via *source-based elastic buckets* that prevent attackers from purging valid messages. We also leverage the concept of *secure copies* to implement *priority sets* to offer protection against Sybil attacks. We have evaluated our solution in a synthetic scenario and have shown that, under flooding attacks,

In comparison to related proposals, RESCUE allows users to join the network post-disaster.

Registered users are well-protected against Sybil attacks, while unregistered ones are not.

RESCUE maintains a delivery rate of 100 % for all registered users, while unregistered users experience a drop of up to 20 %. In the presence of a Sybil attacker, RESCUE maintains a delivery rate close to 100 % for all registered users, while unregistered users can still deliver more than 60 % of their messages. Finally, we confirmed that RESCUE performed reasonably well in a realistic natural disaster scenario. Under a Sybil attack, we showed that the priority set size could be increased to improve the delivery rates, which means that registered users such as professional responder teams can continue to operate. Unregistered users' experiences will, however, suffer drastically under such an attack.

Part IV

CONCLUSIONS

10 CONCLUSIONS

In this thesis, we have investigated how to build distributed wireless networks (DWNs) to be resilient against denial-of-service (DoS) attacks. We have used two different perspectives: analyzing and improving existing protocols in Chapters 4 to 7 and designing novel ones in Chapters 8 and 9.

Today's mobile devices to not offer a high-throughput cross-platform neighbor-scope communication technology.

While designing the two new protocols, we found that the lack of an openly accessible neighbor scope communication technology impeded practical evaluation of, e. g., emergency communication systems on smartphones, which typically use rooted Android devices with Wi-Fi ad hoc [3]. For this reason, we decided to investigate the best-performing protocol, which was a proprietary protocol called Apple Wireless Direct Link (AWDL). We wanted to understand how it worked and, therefore, started reverse-engineering the protocol. During this lengthy process, we learned a lot about the internals of macOS and complex interactions of its various components. Our newly gained insights put us in a unique position of being able to conduct a partial security analysis of a large proprietary ecosystem. In addition, our efforts resulted in several code artifacts that we shared with the community: an AWDL Wireshark dissector and open AWDL as well as AirDrop implementations (Appendix C).

We started reverse-engineering AWDL initially to enable cross-platform neighbor-scope communication.

AWDL is currently available on more than 1.4 billion end-user devices.

When we started this project, AWDL was the only widely available (> 1.4 billion devices) high-throughput neighbor communication technology. Meanwhile, the Wi-Fi Alliance published the specification of Neighbor Awareness Networking (NAN), AWDL's unofficial successor. While NAN is still not widely available on end-user devices (only on selected Android devices), the results of our security analysis on AWDL are at least partially transferable. As the core design concepts of both protocols are identical, our desynchronization attack on AWDL will likely affect NAN as well. Interestingly, we found that the Wi-Fi Alliance discusses the possibility of desynchronization [61, Section 7.5]. However, they do not propose countermeasures arguing that a jamming attack would have the same effect. In our security analysis, however, we were able to show that a protocol-level DoS attack such as desynchronization makes it much easier to launch more sophisticated attacks on higher layers, such as the machine-in-the-middle (MitM) on AirDrop. Consequently, applications relying on open wireless interfaces (AWDL or NAN) might be at risk if developers do not consider the possibility of DoS-supported MitM attacks. Even in the academic security community, the possibility of a MitM attack on a wireless link seems to be underrated [20]. Yet, we are happy that our findings made

DoS attacks can enable MitM attacks on a wireless link.

a real-world impact as Apple has fixed all vulnerabilities (including several privacy leaks) that we reported to them (Appendix B).

Such neighbor communication is the basic building block for multihop communication protocols. In the second part of the thesis, we investigated the latter and proposed two novel DoS-resilient protocols for island and archipelago scope applications.

State-of-the-art IoT networks rely on simple flooding, which does not scale.

Current networking stacks for the Internet of things (IoT), such as Bluetooth Mesh [31], implement flooding-based protocols. While such protocols incur very little complexity and can provide high resiliency against DoS attacks, plain flooding with simple duplicate detection does not scale well as each packet will be forwarded by every node in the network once. This appears especially problematic in light of an estimated IoT device count in the order of ten billion within the next years that are supposed to interconnect in the island scope. For such island networks, we showed that integrated routing protocols could achieve similar resiliency against attacks while being much more bandwidth-efficient. In particular, we proposed a protocol that draws only on efficient symmetric-key cryptographic primitives and uses a secure end-to-end acknowledgment scheme to route around packet dropping attackers. We implemented an open-source prototype of the protocol, which we used to conduct real-world experiments. The results support our analytical proof that LIDOR is, in fact, resilient to a strong wormhole-supported greyhole attack.

Our emergency communication framework is the first of its kind to withstand flooding by a Sybil attacker.

While our lead application scenario throughout this thesis is an emergency scenario, our protocols and proposals are mostly application-agnostic, except for our archipelago communication framework RESCUE, which employs some components that are specific to the emergency scenario. With RESCUE, we have conceived a framework for the emergency context, which is the first of its kind that withstands flooding by a Sybil adversary. To achieve this, we proposed to use a novel buffer management scheme that—based on node-local decisions—prevents an attacker from replacing valid messages with their own. We used a mobility model based on actual past disasters to show that RESCUE remains resilient in practice.

We leave a composite system using the individual protocols presented in this thesis as future work.

We evaluated all protocols via real-world experiments or using simulations with realistic mobility. With promising results on the individual scopes, we think that it is possible to conceive a novel composite system that combines the investigated and proposed protocols in this work for safety-critical applications. For example, we can imagine a DoS-resilient smartphone-based emergency communication system that uses AWDL for neighbor communication and seemlessly switches between island and archipelago communication protocols depending on the context.

Part V

APPENDIX

A

PRIVACY ISSUES IN AWDL

During our security analysis of AWDL, we discovered several privacy issues that easily allow for user tracking. We report these results in this chapter as they are relevant for our security analysis in Chapter 7. First, we discuss protocol fields that enable tracking. Then, we perform an experimental vulnerability assessment at different locations and compare the results with a user study spanning 500 participants. We propose possible mitigations and, finally, summarize related work on user tracking.

AWDL is vulnerable to user tracking.

A.1 PROTOCOL FIELDS WITH SENSITIVE INFORMATION

AWDL implements medium access control (MAC) randomization for the IEEE 802.11 header. Yet, AWDL-specific fields contain long-term device identifiers that disclose sensitive information about the user, undermining MAC randomization. In particular, AWDL includes the following sensitive fields in the action frames (AFs), which devices broadcast *in the clear* multiple times per second when the AWDL interface is active. We indicate the name and type of the tag, which includes this field in parentheses (see Table 7).

The device hostname is transmitted in the clear and often includes the user's name.

- The *hostname* may include parts of the user's *name*, e. g., "Janes-iPhone," which is the default when setting up a new device. (Arpa tag, type 16.)
- The *real MAC address* of the device and the basic service set identifier (BSSID) of the Wi-Fi network that the device is currently connected to. (Data path state tag, type 12.)
- The *device class* differentiates between devices running macOS, iOS/watchOS, and tvOS. In combination with the *protocol version*, this can be used to infer the operating system (OS) version, e. g., AWDL v2 is used in macOS 10.12 while AWDL v3 is used in macOS 10.13 and 10.14. The attacker could exploit the OS information during reconnaissance to mount attacks on vulnerable driver implementations. (Version tag, type 21.)

Targets need to broadcast AFs to make these vulnerabilities exploitable, which an attacker can practically enforce by mounting the attack presented in Section 7.3.

A.2 THE POTENTIAL OF APPLE DEVICE USER TRACKING

We run an iOS and macOS user study to assess the the risk of device and user tracking.

The hostname set by default during Apple iOS and macOS device installation includes the user's name [8]. Due to its frame structure, the AWDL protocol aids an adversary in mapping a hostname with the MAC address of the device. This enables them to track users even if users change this hostname on their device. The combination also enables more sophisticated threats. For example, a person's name can be combined with information from public databases (e. g., US census [183]) to infer their home and work locations, while the MAC address can be used to track them in real time. To assess what percentage of device hostnames contain parts of the owner's name, we conducted a survey of 500 Apple device users on Amazon Mechanical Turk. This survey contained questions[1] relevant to the attacks demonstrated in this thesis, and we report the statistics in the relevant sections. In particular, in the context of tracking, we asked the surveyors if it was easy for other users to find their device because their hostname contained parts of their real name. We report the results of this question, along with the results of an experimental evaluation in the next section.

We asked 500 users to participate in our study via Amazon Mechanical Turk.

A.3 EXPERIMENTAL VULNERABILITY ANALYSIS

To demonstrate the feasibility of user tracking using AWDL, we collect the number of discovered devices and check whether that device's hostname includes a person's name in four different locations within the US. We selected the locations to reflect static as well as dynamic environments. In particular, we recorded at a departure gate of an airport, in the reading section of a public library, in a moving metro train, and in the food court of a university.

We collect data in in an airport, a library, a metro train, and a food court.

DETERMINING WHETHER A HOSTNAME CONTAINS A PERSON'S NAME We use two databases to determine whether a hostname contains a person's name: the 2010 US Census [180] containing 162 253 family names, and the 1918–2017 baby names from the US Social Security Administration [181] containing 96 743 given names. When detecting a new AWDL node, we check string segments separated by hyphens against these two databases.[2] Note that when one segment matches the given name database, it is not matched again as a family name because it is more likely that an Apple device will include a person's given name [8].

We use two US databases containing given and family names to assess whether discovered devices contain a person's name.

1 The survey questionnaire is available at https://goo.gl/forms/0okC4UphTQBnQ0FB3

2 If a segment ends with the letter "s," we also check the segment without a trailing "s." Also, we ignore segments containing common device names such as "iPhone" and "Mac." For example, for the hostname "Johns-iPhone," we try to match the strings "Johns" and "John" to our name databases.

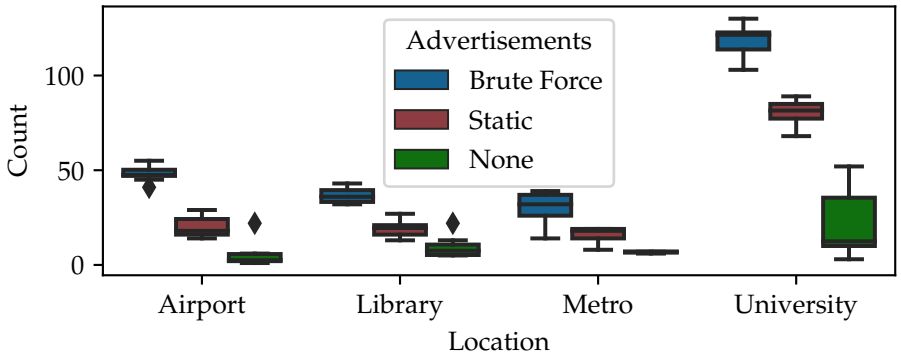


Figure 50: Discovered AWDL devices at one location during one minute.

ETHICAL STATEMENT To preserve user privacy and to prevent storage of any sensitive user information, we fully automated the name matching procedure. In particular, we only stored salted hashes of the discovered hostnames (to differentiate between devices) together with two bits indicating whether the hostname contained a given or a family name. The salt was generated randomly, kept in memory only, and discarded after the completion of each experiment.

We do not store identifiable information about discovered devices.

SETUP We do the measurements (1) without an attack (passive), (2) with static Bluetooth Low Energy (BLE) advertisements containing only the "zero" contact hash, and (3) with our BLE brute force approach. With (2), only devices in the *everyone* mode should respond, with (3) we also capture those that are in *contacts-only* mode. We run each setting for 60 seconds and repeat it ten times per location. To avoid statistical bias, we cycle through the (1) to (3) settings back to back in each iteration and use a cooldown time of 40 seconds between them. The cooldown ensures that all devices in proximity have turned off their AWDL interfaces again.

We also count the number of devices that are in everyone and contacts-only only mode.

EXPERIMENTAL RESULTS Figure 50 shows the number of discovered AWDL devices in the different locations. By using the brute force approach, we can discover about twice as many devices compared to sending only regular advertisements. This means that in our experiments, approximately 50 % of the Apple devices are in AirDrop's *everyone* mode. Our survey complements our experimental results by indicating that 20 % of Apple device users have AirDrop turned off and, thus, are not trackable via AWDL. It is interesting to note that we pick up AWDL devices even when not sending any advertisements. This can happen if a device (not controlled by us) sends out advertisements itself, for example, when a user opens the AirDrop sharing pane, which presumably occurred regularly at the university location. Finally, we found that among all discovered devices, more than 75 % contain a person's name in the hostname. Most devices contain only a given name, which is the default for freshly set up Apple devices [8],

The BLE brute force is effective in enabling contacts-only devices.

More than 75 % of the discovered device names contain a person's name.

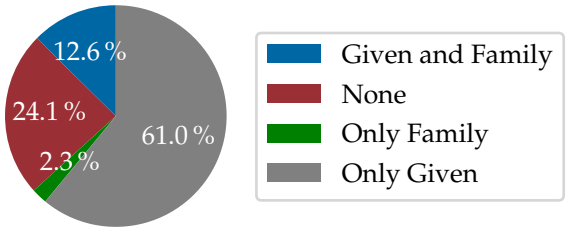


Figure 51: Occurrences of persons' names in device hostnames.

some contain a combination of a given and family name, and very few contain only a family name. While we cannot verify that the discovered names are the users' real names, our survey confirms our experimental results as 68 % answered that it was "easy" or "very easy" for others to recognize their device because it contained their name.

A large scale tracking attack would be possible if the attacker deployed several low-cost nodes in a target area.

POTENTIAL FOR LARGE-SCALE ATTACK In this analysis, we show what kind of information a motivated attacker would be able to collect. We used a single fixed physical location for each experiment and did not attempt to track any user movement. However, given that we can receive unique identifiers of Apple devices (Wi-Fi MAC address and hostname), mounting a large-scale tracking attack should be trivial for an adversary that can deploy multiple low-cost Wi-Fi and BLE nodes throughout an area.

A.4 MITIGATION

We present a short-term solution and then propose two mitigation techniques that remove stable device identifiers to prevent user tracking via AWDL.

Disabling AWDL triggers helps to protect user privacy.

DISABLE AIRDROP Before Apple addressed the problem, the only way to thwart user tracking was to disable AirDrop completely. Disabling AirDrop prevents BLE advertisements from activating the AWDL interface remotely. Apple has meanwhile implemented a rate-limiting mechanism (Appendix B.7) that prohibits our brute force attack and makes activating nearby AWDL devices much harder.

Apple has removed the real MAC address from AWDL AFs.

HIDE REAL MAC ADDRESS WHEN NOT CONNECTED TO AN ACCESS POINT When a device connects to an access point (AP), it uses its real MAC address for communication, in which case AWDL does not disclose new information. However, the MAC address is occasionally included in AFs even when the device is not connected to an AP. This appears to be unintended behavior and should be fixed via a

software update, which Apple has done in response to our report (Appendices B.3 and B.5).

RANDOMIZE HOSTNAME FOR AWDL Apple devices transmit their hostname in AWDL AFs as well as the multicast DNS (mDNS) responses during service discovery that are used to find AirDrop instances (see Chapter 6). As a countermeasure, we propose to use a randomized hostname with AWDL similar to MAC address randomization. If an application such as AirDrop needs the real hostname for identification, it should only be transmitted via an encrypted and authenticated channel such as TLS. AirDrop already transmits the device name in the HTTPS handshake and uses this name in the UI, ignoring the hostname from mDNS responses. Therefore, hostname randomization would not require any changes to the AirDrop implementation, which would retain backward compatibility.

We propse hostname randomization in analogy to MAC address randomization which Apple has implemented in response to our report.

Apple has picked up our idea and replaced the hostname in both AWDL AFs and mDNS responses with a random universally unique identifier (UUID) that rotates together with the MAC address (Appendix B.6).

A.5 RELATED WORK ON USER TRACKING

Several related works have studied the topic of user tracking from mobile devices. Some common attack vectors include using the GPS sensor [65, 110, 126, 179, 192], cellular [14, 86, 112, 154], Wi-Fi [30, 79, 159, 201, 202], radio interface fingerprinting [93], and motion sensors [51, 78, 134, 138, 140]. We believe that the above works are orthogonal to our approach, and could be used in conjunction with our approach to improve tracking performance.

Some device-specific identifiers have also been used for tracking, e. g., IMEI [68, 190], MAC addresses [46, 63, 124, 137], BLE addresses [48, 59, 108], and custom BLE advertisements [129]. While IMEI-based tracking can be easily mitigated by protecting access to this information, BLE is a dominant standard for fitness trackers and smartphone communication, and their addresses must be exposed. Tracking using BLE identifiers has been demonstrated to be easy. However, our approach has the added benefit for an attacker that the hostname is exposed. This allows inferring additional user information such as home and work locations, family members, or movement patterns, which are useful for more targeted tracking [66, 182]. Like BLE addresses, MAC addresses are also essential as they form the backbone of layer 2 network communication and must be exposed for networking (e. g., Wi-Fi probe requests).

Device identifiers such as MAC addresses have been previously used to track mobile users.

MAC address randomization has been proposed to prevent device tracking through Wi-Fi probe requests [23, 52]. Today, both Apple and Google implement MAC address randomization in their mobile

MAC address randomization is used in industry to preserve Wi-Fi user privacy but has been shown to still be vulnerable to tracking attacks.

OSs. Randomization does improve user privacy; however, some works have demonstrated that devices are still trackable. For example, a study [186] uses probe request fingerprinting that has a 50 percent success rate for tracking users for 20 minutes. Another work [130] demonstrated that MAC randomization could be defeated through timing attacks, where a signature based on inter-frame arrival times of probe requests can be used to group frames coming from the same device with distinct MAC addresses. Their framework could group random MAC addresses of the same device up to 75% of cases for about 500 devices. Our work advances the scalability, tracking time, and accuracy of the prior works. We show that, owing to design and implementation issues in the AWDL protocol, an adversary could track millions of Apple device owners globally with 100 % accuracy until Apple fixed the vulnerabilities in recent OS updates (Appendices B.3 and B.5 to B.7).

B

VULNERABILITY DISCLOSURES

We list all vulnerabilities that we discovered during the course of this thesis. We give a discription and the disclosure date for each vulnerability and, if available, provide information about the update releases. The web links to the advisories (`HT...`) can be accessed by prefixing them with `https://support.apple.com/en-us/`.

B.1 CVE-2018-4368

DESCRIPTION	An attacker may wirelessly crash devices in proximity by sending malformed AWDL frames.
DISCLOSURE	August 27, 2018
MITIGATION	The DoS attack was addressed with improved validation.
RELEASES	October 30, 2018 for iOS 12.1 (HT209192) October 30, 2018 for macOS 10.14.1 (HT209193) October 30, 2018 for tvOS 12.1 (HT209194) October 30, 2018 for watchOS 5.1 (HT209195)
CREDITS	Milan Stute and Alex Mariotto

B.2 NO-CVE-2018-1

DESCRIPTION	An attacker may wirelessly crash devices in proximity by sending malformed AWDL frames.
DISCLOSURE	August 14, 2018
AFFECTED	macOS 10.12
COMMENT	Apple declined to patch old OS version.

B.3 CVE-2019-8567

DESCRIPTION	An attacker may passively track a device by the broadcast Wi-Fi MAC address.
DISCLOSURE	December 17, 2018
MITIGATION	The privacy issue was addressed by removing the BSSID MAC address from AWDL frames.
RELEASES	March 25, 2019 for iOS 12.2 (HT209599) March 25, 2019 for macOS 10.14.4 (HT209600)
CREDITS	David Kreitschmann and Milan Stute

B.4 CVE-2019-8612

DESCRIPTION An attacker may perform a DoS attack by desynchronizing two AWDL targets.

DISCLOSURE December 17, 2018

MITIGATION A logic issue was addressed with improved state management.

RELEASES May 13, 2019 for iOS 12.3 (HT210118)
May 13, 2019 for macOS 10.14.5 (HT210119)
May 13, 2019 for tvOS 12.3 (HT210120)
May 13, 2019 for watchOS 5.2.1 (HT210122)

CREDITS Milan Stute

B.5 CVE-2019-8620

DESCRIPTION An attacker may passively track a device by the broadcast Wi-Fi MAC address.

DISCLOSURE December 17, 2018

MITIGATION The privacy issue was addressed by removing the Wi-Fi chip's MAC address from AWDL frames.

RELEASES May 13, 2019 for iOS 12.3 (HT210118)
May 13, 2019 for macOS 10.14.5 (HT210119)
May 13, 2019 for tvOS 12.3 (HT210120)
May 13, 2019 for watchOS 5.2.1 (HT210122)

CREDITS David Kreitschmann and Milan Stute

B.6 CVE-2019-8799

DESCRIPTION An attacker in physical proximity may be able to passively observe device names in AWDL communications.

DISCLOSURE December 17, 2018

MITIGATION This issue was resolved by replacing device names with a random identifier.

RELEASES September 19, 2019 for iOS/iPadOS 13.1 (HT210603)
September 19, 2019 for watchOS 13 (HT210607)
September 24, 2019 for tvOS 13 (HT210604)
October 7, 2019 for macOS 10.15 (HT210634)

CREDITS David Kreitschmann and Milan Stute

B.7 NO-CVE-2019-1

DESCRIPTION	An attacker may wirelessly activate AWDL interfaces in proximity by sending BLE advertisements.
DISCLOSURE	December 17, 2018
MITIGATION	The brute force attack was addressed via rate-limiting
RELEASES	March 25, 2019 for iOS 12.2 (HT209599) March 25, 2019 for macOS 10.14.4 (HT209600) March 25, 2019 for tvOS 12.2 (HT209601) March 27, 2019 for watchOS 5.2 (HT209602)
CREDITS	Milan Stute
COMMENT	No CVE but mentioned in "additional recognition."

B.8 NO-CVE-2019-2

DESCRIPTION	An AirDrop user may be tricked into sending files to an attacker.
DISCLOSURE	December 17, 2018
MITIGATION	The sharing pane UI now differentiates between authenticated and unauthenticated devices.
RELEASE	September 19, 2019 for iOS/iPadOS 13.1 (HT210603)
CREDITS	Milan Stute
COMMENT	No CVE but mentioned in "additional recognition."

B.9 CVE-2017-13886 (ASSOCIATED)

DESCRIPTION	An unprivileged user may change Wi-Fi system parameters leading to denial of service.
DISCLOSURE	July 19, 2017
MITIGATION	Introduce gated `ioctl` commands using special entitlement.
RELEASE	December 6, 2017 for macOS 10.13.2 (HT208331)
CREDITS	David Kreitschmann and Matthias Schulz
COMMENT	Discovered by David during his master thesis under my supervision. Part of our co-authored paper [174].

C

SOFTWARE RELEASES

This section lists all open-source software artifacts that have been developed and published as part of this thesis. Instructions on how to use the particular programs are included in the `README` files of the respective source code repositories. All Apple-related projects can also be found on the website of the Open Wireless Link project https://owlink.org.

C.1 AWDL PROTOCOL DISSECTOR FOR WIRESHARK

We implemented a Wireshark dissector for Apple Wireless Direct Link (AWDL). The dissector has been merged with the official Wireshark project and is shipped since version 3.0. The original source code repository is no longer maintained but contains David Kreitschmann's CoreCapture dissector, which is not part of the official Wireshark project.

RELEASE NOTES https://www.wireshark.org/docs/relnotes/wireshark-3.0.0.html

SOURCE CODE https://github.com/seemoo-lab/wireshark-awdl

AUTHORS David Kreitschmann and Milan Stute

C.2 OPEN WIRELESS LINK

Open Wireless Link (OWL) is a an open implementation of the AWDL protocol written in C. It currently runs on Linux and macOS and requires a Wi-Fi card with support for monitor mode and frame injection. OWL runs in userspace and makes use of Linux's Netlink API for Wi-Fi specific operations such as channel switching and to integrate itself in the Linux networking stack by providing a virtual network interface such that existing IPv6-capable programs can use AWDL without modification.

SOURCE CODE https://github.com/seemoo-lab/owl

AUTHOR Milan Stute

C.3 OPENDROP

OpenDrop is a command-line tool written in Python that allows sharing files between devices directly over Wi-Fi. It is protocol-compatible with Apple AirDrop and allows to share files with Apple devices running iOS and macOS. To support communication with Apple devices, OpenDrop needs to run over an AWDL-compatible link and, thus, supports macOS and any platform that supports OWL.

BINARY RELEASE https://pypi.org/project/opendrop
SOURCE CODE https://github.com/seemoo-lab/opendrop
AUTHORS Alexander Heinrich and Milan Stute

C.4 LIDOR COMMUNICATION PROTOCOL

We implemented the LIDOR communication protocol for the Click Modular Router [106]. The end-to-end communication protocol runs on layer 2, is based on Castor [64] (hence the original project name), and features advanced security properties such as resiliency against advanced attacks, such as blackhole, greyhole, wormhole, and replay attacks.

SOURCE CODE https://github.com/seemoo-lab/click-castor
AUTHOR Milan Stute

C.5 NATURAL DISASTER MOBILITY MODEL AND SCENARIOS

We developed and implemented a natural disaster mobility model as a module for the ONE simulator [104]. The repository contains a mobility model for natural disasters as well as two specific disaster scenarios: 2013 Typhoon Haiyan (for the city of Tacloban) and 2010 Haiti earthquake (for the surroundings of Port au Prince).

SOURCE CODE https://github.com/seemoo-lab/natural-disaster-mobility
EXPERIMENT DATA doi:10.5281/zenodo.836815
AUTHORS Tom Schons and Milan Stute

BIBLIOGRAPHY

[1] Cédric Adjih, Daniele Raffo, and Paul Mühlethaler. "Attacks Against OLSR: Distributed Key Management for Security." In: *2nd OLSR Interop/Workshop*. Aug. 2005. URL: https://perso.crans.org/~raffo/papers/attacks-olsr-dkm.pdf (retrieved Dec. 9, 2019) (cit. on p. 15).

[2] Lars Almon, Flor Álvarez, Laurenz Kamp, and Matthias Hollick. "The King is Dead Long Live the King! Towards Systematic Performance Evaluation of Heterogeneous Bluetooth Mesh Networks in Real World Environments." In: *IEEE Conference on Local Computer Networks (LCN)*. Oct. 2019 (cit. on p. 19).

[3] Flor Álvarez, Lars Almon, Patrick Lieser, Tobias Meuser, Yannick Dylla, Björn Richerzhagen, Matthias Hollick, and Ralf Steinmetz. "Conducting a Large-scale Field Test of a Smartphone-based Communication Network for Emergency Response." In: *ACM Workshop on Challenged Networks (CHANTS)*. Oct. 2018. DOI: 10.1145/3264844.3264845 (cit. on p. 147).

[4] Android Developers. *SafetyNet Attestation API*. URL: https://developer.android.com/training/safetynet/attestation (retrieved Dec. 9, 2019) (cit. on p. 129).

[5] Android Developers. *Wi-Fi Aware*. URL: https://source.android.com/devices/tech/connect/wifi-aware (retrieved Dec. 9, 2019) (cit. on p. 19).

[6] Apple Inc. *System Integrity Protection Guide*. Sept. 16, 2015. URL: https://developer.apple.com/library/archive/documentation/Security/Conceptual/System_Integrity_Protection_Guide/Introduction/Introduction.html (retrieved Dec. 9, 2019) (cit. on p. 36).

[7] Apple Inc. *About Entitlements*. Mar. 27, 2017. URL: https://developer.apple.com/library/archive/documentation/Miscellaneous/Reference/EntitlementKeyReference/Chapters/AboutEntitlements.html (retrieved Dec. 9, 2019) (cit. on p. 36).

[8] Apple Inc. *Change the Name of Your iPhone, iPad, or iPod Using Your Computer*. Oct. 7, 2019. URL: https://support.apple.com/en-us/HT201997 (retrieved Dec. 9, 2019) (cit. on pp. 152, 153).

[9] Apple Inc. *How to Unlock Your Mac with Your Apple Watch*. Oct. 2019. URL: https://support.apple.com/en-us/HT206995 (retrieved Dec. 9, 2019) (cit. on p. 3).

[10] Apple Inc. *How to use AirDrop on your iPhone, iPad, or iPod touch*. Oct. 2019. URL: https://support.apple.com/en-us/HT204144 (retrieved Dec. 9, 2019) (cit. on pp. 3, 4, 67).

[11] Apple Inc. "Introducing iPad Apps for Mac." In: *Apple Worldwide Developers Conference (WWDC)*. June 2019. URL: https://developer.apple.com/videos/play/wwdc2019/205/ (retrieved Dec. 9, 2019) (cit. on pp. 32, 39).

[12] Apple Inc. *iOS Security – iOS 12.3*. May 2019. (Retrieved Dec. 9, 2019) (cit. on pp. 32, 39, 43).

[13] Apple Inc. *NSNetService Class Documentation*. URL: https://developer.apple.com/documentation/foundation/nsnetservice (retrieved Dec. 9, 2019) (cit. on pp. 43, 48, 61).

[14] Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, and Mark Dermot Ryan. "Analysis of Privacy in Mobile Telephony Systems." In: *International Journal of Information Security* (Oct. 2017). DOI: 10.1007/s10207-016-0338-9 (cit. on p. 155).

[15] Nitay Artenstein. "Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets." In: *Exodus Intelligence* (July 26, 2017). URL: https://blog.exodusintel.com/2017/07/26/broadpwn/ (retrieved Dec. 9, 2019) (cit. on p. 89).

[16] Nils Aschenbruck and Matthias Schwamborn. "Synthetic Map-Based Mobility Traces for the Performance Evaluation in Opportunistic Networks." In: *Proceedings of the Second International Workshop on Mobile Opportunistic Networking (MobiOpp)*. Feb. 2010. DOI: 10.1145/1755743.1755769 (cit. on p. 27).

[17] Jean Philippe Aumasson and Daniel J. Bernstein. "SipHash: A Fast Short-Input PRF." In: *Progress in Cryptology (INDOCRYPT)*. Springer Berlin Heidelberg, Dec. 2012. DOI: 10.1007/978-3-642-34931-7_28 (cit. on p. 112).

[18] Algirdas Avižienis, Jean Claude Laprie, Brian Randell, and Carl Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing." In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 1.1 (Jan. 2004), pp. 11–33. DOI: 10.1109/TDSC.2004.2 (cit. on p. 14).

[19] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. "ODSBR: An On-Demand Secure Byzantine Resilient Routing Protocol for Wireless Ad Hoc Networks." In: *ACM Transactions on Information and System Security* 10.4 (Jan. 2008), 6:1–6:35. DOI: 10.1145/1284680.1341892 (cit. on pp. 20, 21).

[20] Xiaolong Bai, Luyi Xing, Nan Zhang, Xiaofeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. "Staying Secure and Unprepared: Understanding and Mitigating the Security Risks of Apple ZeroConf." In: *IEEE Symposium on Security and Privacy (S&P)*. May 2016. DOI: 10.1109/SP.2016.45 (cit. on pp. 83, 147).

[21] Kashyap Balakrishnan, Jing Deng, and Pramod K. Varshney. "TWOAK: Preventing Selfishness in Mobile Ad Hoc Networks." In: *IEEE Wireless Communications and Networking Conference*. Mar. 2005. DOI: 10.1109/WCNC.2005.1424848 (cit. on p. 20).

[22] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. "DTN Routing as a Resource Allocation Problem." In: *SIGCOMM Computer Communication Review* 37.4 (Aug. 2007), pp. 373–384. DOI: 10.1145/1282427.1282422 (cit. on p. 130).

[23] Marco V. Barbera, Alessandro Epasto, Alessandro Mei, Vasile C. Perta, and Julinda Stefa. "Signals from the Crowd: Uncovering Social Relationships through Smartphone Probes." In: *ACM Internet Measurement Conference (IMC)*. Oct. 2013. DOI: 10.1145/2504730.2504742 (cit. on p. 155).

[24] Lars Baumgärtner, Stefan Kohlbrecher, Juliane Euler, Tobias Ritter, Milan Stute, Christian Meurisch, Max Muhlhäuser, Matthias Hollick, Oskar von Stryk, and Bernd Freisleben. "Emergency Communication in Challenged Environments via Unmanned Ground and Aerial Vehicles." In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Oct. 2017. DOI: 10.1109/GHTC.2017.8239244 (cit. on p. 120).

[25] Emrah Bayraktaroglu, Christopher King, Xin Liu, Guevara Noubir, Rajmohan Rajaraman, and Bishal Thapa. "Performance of IEEE 802.11 Under Jamming." In: *Mobile Networks and Applications* 18.5 (Oct. 2013), pp. 678–696. DOI: 10.1007/s11036-011-0340-4 (cit. on p. 15).

[26] Sarah Bell. "Police Investigate 'First Cyber-Flashing' Case." In: *BBC News* (Aug. 13, 2015). URL: https://www.bbc.com/news/technology-33889225 (retrieved Dec. 9, 2019) (cit. on p. 83).

[27] Steve M. Bellovin. "Security as a Systems Property." In: *IEEE Security & Privacy* 7.5 (Sept. 2009), pp. 88–88. DOI: 10.1109/MSP.2009.134 (cit. on p. 13).

[28] Gal Beniamini. "Over the Air: Exploiting Broadcom's Wi-Fi Stack (Part 2)." In: *Google Project Zero* (Apr. 11, 2017). URL: https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html (retrieved Dec. 9, 2019) (cit. on p. 89).

[29] Christian Bettstetter, Giovanni Resta, and Paolo Santi. "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks." In: *IEEE Transactions on Mobile Computing (TMC)* 2.3 (July 2003), pp. 257–269. DOI: 10.1109/TMC.2003.1233531 (cit. on p. 27).

[30] Laurent Bindschaedler, Murtuza Jadliwala, Igor Bilogrevic, Imad Aad, Philip Ginzboorg, Valtteri Niemi, and Jean-Pierre Hubaux. "Track Me If You Can: On the Effectiveness of Context-based Identifier Changes in Deployed Mobile Networks." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2012. URL: https://www.ndss-symposium.org/ndss2012/track-me-if-you-can-effectiveness-context-based-identifier-changes-deployed-mobile-networks (retrieved Dec. 9, 2019) (cit. on p. 155).

[31] Bluetooth SIG. *Mesh Model Bluetooth Specification v1.0*. July 2017. URL: https://www.bluetooth.com/specifications/mesh-specifications/ (retrieved Dec. 9, 2019) (cit. on pp. 4, 19, 148).

[32] Bluetooth SIG. *Bluetooth Core Specification v5.1*. Tech. rep. Jan. 2019. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification/ (retrieved Dec. 9, 2019) (cit. on pp. 18, 19, 84–87).

[33] Bluetooth SIG. *Bluetooth Market Update 2019*. May 2019. URL: https://www.bluetooth.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf (cit. on p. 3).

[34] Paulo Borges. *BLESSED*. URL: https://github.com/paulobordes/blessed (retrieved Dec. 9, 2019) (cit. on p. 87).

[35] Briar Project. *Website*. URL: https://briarproject.org (retrieved Dec. 9, 2019) (cit. on p. 21).

[36] Broadcom. *Leaked BCM4360 Driver Code*. No longer available. 2015. URL: https://github.com/kyuhsim/khsim_repository/tree/72708c6709/FutureSys/FutureProj_20141222/hg_clone/D700_wl (cit. on p. 34).

[37] Broadcom. *Leaked Asus RT-AC86U Driver Code*. URL: https://github.com/blackfuel/asuswrt-rt-ac86u/tree/master/release/src-rt-5.02hnd (retrieved Dec. 9, 2019) (cit. on p. 52).

[38] John Burgess, George Dean Bissias, Mark D. Corner, Brian Neil Levine, and Brian Neil Levine. "Surviving Attacks on Disruption-tolerant Networks Without Authentication." In: *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. Sept. 2007. DOI: 10.1145/1288107.1288116 (cit. on pp. 15, 21).

[39] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. "Device-to-Device Communications with Wi-Fi Direct: Overview and Experimentation." In: *IEEE Wireless Communications* 20.3 (July 2013), pp. 96–104. DOI: 10.1109/MWC.2013.6549288 (cit. on p. 18).

[40] Daniel Camps-Mur, Eduard Garcia Villegas, Elena López-Aguilera, Paulo Loureiro, Paul Lambert, and Ali Raissinia. "Enabling Always On Service Discovery: WiFi Neighbor Awareness Networking." In: *IEEE Wireless Communications* 22.2 (Apr. 2015), pp. 118–125. DOI: 10.1109/MWC.2015.7096294 (cit. on p. 19).

[41] Glenn Carl, George Kesidis, Richard R. Brooks, and Suresh Rai. "Denial-of-Service Attack-Detection Techniques." In: *IEEE Internet Computing* 10.1 (Jan. 2006), pp. 82–89. DOI: 10.1109/MIC.2006.5 (cit. on p. 15).

[42] Stuart D. Cheshire. "Proximity Wi-Fi." In: *U.S. Patent Application* US 2018/0083858 A1 (Mar. 2018). URL: https://patents.google.com/patent/US20180083858A1 (cit. on pp. 8, 17).

[43] Thomas Heide Clausen and Philippe Jacquet. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626. IETF, Oct. 2003. DOI: 10.17487/RFC3626 (cit. on pp. 6, 99).

[44] Cloudflare. *What is DDoS Mitigation?* URL: https://www.cloudflare.com/learning/ddos/ddos-mitigation/ (retrieved Dec. 9, 2019) (cit. on p. 4).

[45] Aldo Cortesi, Maximilian Hils, and Thomas Kriechbaumer. *mitmproxy: a Free and Open Source Interactive HTTPS Proxy*. URL: https://mitmproxy.org (retrieved Dec. 9, 2019) (cit. on p. 38).

[46] Mathieu Cunche. "I Know Your MAC Address: Targeted Tracking of Individual Using Wi-Fi." In: *Journal of Computer Virology and Hacking Techniques* 10.4 (Nov. 2013), pp. 219–227. DOI: 10.1007/s11416-013-0196-1 (cit. on p. 155).

[47] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. "A Reputation-based Approach for Choosing Reliable Resources in Peer-to-peer Networks." In: *ACM Conference on Computer and Communications Security (CCS)*. Nov. 2002. DOI: 10.1145/586110.586138 (cit. on p. 16).

[48] Aveek K. Das, Parth H. Pathak, Chen-Nee Chuah, and Prasant Mohapatra. "Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers." In: *ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*. Feb. 2016. DOI: 10.1145/2873587.2873594 (cit. on p. 155).

[49] Deloitte. *Building Resilience to Denial-of-Service Attacks*. 2018. URL: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/risk/us-building-resilience-to-denial-of-service-attacks.pdf (retrieved Dec. 9, 2019) (cit. on p. 3).

[50] Frank Denis. *Sodium*. URL: https://libsodium.org (retrieved Dec. 9, 2019) (cit. on p. 112).

[51] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2014. DOI: 10.14722/ndss.2014.23059 (cit. on p. 155).

[52] Adriano Di Luzio, Alessandro Mei, and Julinda Stefa. "Mind Your Probes: De-Anonymization of Large Crowds Through Smartphone WiFi Probe Requests." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2016. DOI: 10.1109/INFOCOM.2016.7524459 (cit. on p. 155).

[53] Craig Dooley and Duy Phan. "What's New in Core Bluetooth." In: *Apple Worldwide Developers Conference (WWDC)*. June 2017. URL: https://developer.apple.com/videos/play/wwdc2017/712/ (retrieved Dec. 9, 2019) (cit. on p. 19).

[54] John R. Douceur. "The Sybil Attack." In: *Peer-to-Peer Systems*. Springer Berlin Heidelberg, 2002, pp. 251–260. DOI: 10.1007/3-540-45748-8_24 (cit. on p. 16).

[55] Mark Dowd. "MalwAirDrop: Compromising iDevices via AirDrop." In: *Ruxcon*. Oct. 2015. URL: https://2015.ruxcon.org.au/assets/2015/slides/ruxcon-2016-dowd.pptx (retrieved Dec. 9, 2019) (cit. on p. 84).

[56] Ericsson. *Mobility Report*. June 2019. URL: https://www.ericsson.com/en/mobility-report/reports/june-2019 (retrieved Dec. 9, 2019) (cit. on p. 3).

[57] Jakob Eriksson, Michalis Faloutsos, and Srikanth V. Krishnamurthy. "Routing Amid Colluding Attackers." In: *IEEE International Conference on Network Protocols (ICNP)*. Oct. 2007. DOI: 10.1109/ICNP.2007.4375849 (cit. on pp. 20, 21).

[58] European Parliament and Council of the European Union. *Regulation (EU) No 910/2014*. July 23, 2014. URL: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32014R0910 (retrieved Dec. 9, 2019) (cit. on p. 129).

[59] Kassem Fawaz, Kyu-Han Kim, and Kang G. Shin. "Protecting Privacy of BLE Device Users." In: *USENIX Security Symposium*. Aug. 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fawaz (retrieved Dec. 9, 2019) (cit. on p. 155).

[60] Wi-Fi Alliance. *Wi-Fi Peer-to-Peer (P2P) Technical Specification*. Tech. rep. July 2016. URL: https://www.wi-fi.org/discover-wi-fi/specifications (retrieved Dec. 9, 2019) (cit. on p. 18).

[61] Wi-Fi Alliance. *Neighbor Awareness Networking Technical Specification Version 2.0*. Tech. rep. Oct. 2017. URL: https://www.wi-fi.org/discover-wi-fi/specifications (retrieved Dec. 9, 2019) (cit. on pp. 18, 19, 147).

[62] Howard Frank and Ivan T. Frisch. "Analysis and Design of Survivable Networks." In: *IEEE Transactions on Communication Technology* 18.5 (Oct. 1970), pp. 501–519. DOI: 10.1109/TCOM.1970.1090419 (cit. on p. 13).

[63] Julien Freudiger. "How Talkative is Your Mobile Device? An Experimental Study of Wi-Fi Probe Requests." In: *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. June 2015. DOI: 10.1145/2766498.2766517 (cit. on p. 155).

[64] Wojciech Galuba, Panagiotis Papadimitratos, Marcin Poturalski, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. "Castor: Scalable Secure Routing for Ad Hoc Networks." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Mar. 2010. DOI: 10.1109/INFCOM.2010.5462115 (cit. on pp. 10, 20, 21, 93–95, 98–101, 104, 113, 120, 130, 162).

[65] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez Del Prado Cortez. "De-anonymization Attack on Geolocated Data." In: *Journal of Computer and System Sciences* 80 (Dec. 2014). DOI: 10.1016/j.jcss.2014.04.024 (cit. on p. 155).

[66] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "Show Me How You Move and I Will Tell You Who You Are." In: *ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. Nov. 2010. DOI: 10.1145/1868470.1868479 (cit. on p. 155).

[67] Gartner. "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent from 2016." In: (Feb. 7, 2017). URL: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016 (retrieved Dec. 9, 2019) (cit. on p. 3).

[68] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. "AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale." In: *International Conference on Trust and Trustworthy Computing*. Springer Berlin Heidelberg, June 2012. DOI: 10.1007/978-3-642-30921-2_17 (cit. on p. 155).

[69] Marc Girault, Robert Cohen, and Mireille Campana. "A Generalized Birthday Attack." In: *Advances in Cryptology (EUROCRYPT)*. Springer Berlin Heidelberg, 1988, pp. 129–156. ISBN: 978-3-540-45961-3 (cit. on p. 98).

[70] Harry Goldstein. "Engineers Race to Restore Communications after Haiti Quake." In: *IEEE Spectrum* (Jan. 19, 2010). URL: https://spectrum.ieee.org/tech-talk/telecom/internet/engineers-race-to-restore-communications-after-haiti-quake (retrieved Dec. 9, 2019) (cit. on p. 23).

[71] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. "They Can Hear Your Heartbeats: Non-Invasive Security for Implantable Medical Devices." In: 41.4 (Oct. 2011), pp. 2–13. DOI: 10.1145/2043164.2018438 (cit. on pp. 15, 78).

[72] Google Developers. *Nearby: A Platform for Discovering and Communicating with Nearby Devices*. URL: https://developers.google.com/nearby (retrieved Dec. 9, 2019) (cit. on p. 3).

[73] Matthew Green. "How Does Apple (Privately) Find Your Offline Devices?" In: (June 5, 2019). URL: https://blog.cryptographyengineering.com/2019/06/05/how-does-apple-privately-find-your-offline-devices/ (retrieved Dec. 9, 2019) (cit. on p. 3).

[74] Matthew Green. "Looking Back at the Snowden Revelations." In: (Sept. 24, 2019). URL: https://blog.cryptographyengineering.com/2019/09/24/looking-back-at-the-snowden-revelations/ (retrieved Dec. 9, 2019) (cit. on p. 3).

[75] Debarati Guha-Sapir. *EM-DAT: The Emergency Events Database*. Université Catholique de Louvain, CRED. URL: https://www.emdat.be (retrieved Dec. 9, 2019) (cit. on p. 24).

[76] Arnt Gulbrandsen, Paul Vixie, and Levon Esibov. *A DNS RR for Specifying the Location of Services (DNS SRV)*. RFC 2782. IETF, Feb. 2000. DOI: 10.17487/RFC2782 (cit. on p. 53).

[77] Piyush Gupta and P. R. Kumar. "The Capacity of Wireless Networks." In: *IEEE Transactions on Information Theory* 46.2 (Mar. 2000), pp. 388–404. DOI: 10.1109/18.825799 (cit. on p. 119).

[78] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. "ACComplice: Location Inference Using Accelerometers on Smartphones." In: *IEEE International Conference on Communication Systems and Networks (COMSNETS)*. Jan. 2012. DOI: 10.1109/COMSNETS.2012.6151305 (cit. on p. 155).

[79] Xiuping Han, Zhi Wang, and Dan Pei. "Preventing Wi-Fi Privacy Leakage: A User Behavioral Similarity Approach." In: *IEEE International Conference on Communications (ICC)*. May 2018. DOI: 10.1109/ICC.2018.8422764 (cit. on p. 155).

[80] Harry Harris. "Oakland-Maui Flight: Pepper Spray Emergency Follows Disturbing Photo." In: *East Bay Times* (Sept. 1, 2018). URL: https://www.eastbaytimes.com/2018/09/01/oakland-maui-pepper-spray-disturbing-photo-delay/ (retrieved Dec. 9, 2019) (cit. on p. 83).

[81] Richard Harris. "Why Typhoon Haiyan Caused So Much Damage." In: *NPR* (Nov. 11, 2013). URL: https://www.npr.org/2013/11/11/244572227/why-typhoon-haiyan-caused-so-much-damage?t=1575975713507 (retrieved Dec. 9, 2019) (cit. on p. 23).

[82] Robert M. Hinden and Stephen E. Deering. *IP Version 6 Addressing Architecture*. RFC 4291. IETF, Feb. 2006. DOI: 10.17487/RFC4291 (cit. on p. 47).

[83] Matthias Hollick, Cristina Nita-Rotaru, Panagiotis Papadimitratos, Adrian Perrig, and Stefan Schmid. "Toward a Taxonomy and Attacker Model for Secure Routing Protocols." In: *SIGCOMM Computer Communication Review* 47.1 (Jan. 2017), pp. 43–48. DOI: 10.1145/3041027.3041033 (cit. on p. 17).

[84] Matthias Hollick, Jens Schmitt, Christian Seipl, and Ralf Steinmetz. "On the Effect of Node Misbehavior in Ad Hoc Networks." In: *IEEE International Conference on Communications (ICC)*. June 2004. DOI: 10.1109/ICC.2004.1313244 (cit. on p. 17).

[85] Erik Hollnagel, David D. Woods, and Nancy Leveson. *Resilience Engineering: Concepts and Precepts*. Ashgate Publishing, Ltd., 2006. ISBN: 0-7546-4641-6 (cit. on p. 14).

[86] Byeongdo Hong, Sangwook Bae, and Yongdae Kim. "GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2018. DOI: 10.14722/ndss.2018.23349 (cit. on p. 155).

[87] Theus Hossmann, Paolo Carta, Dominik Schatzmann, Franck Legendre, Per Gunningberg, and Christian Rohner. "Twitter in Disaster Mode: Security Architecture." In: *ACM Special Workshop on Internet and Disasters*. Dec. 2011. DOI: 10.1145/2079360.2079367 (cit. on p. 21).

[88] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks." In: *Ad Hoc Networks* 1.1 (July 2003), pp. 175–192. DOI: 10.1016/S1570-8705(03)00019-2 (cit. on pp. 20, 95, 101).

[89] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. *Wormhole Detection in Wireless Ad Hoc Networks*. TR 01-384. Department of Computer Science, Rice University, June 2002 (cit. on p. 16).

[90] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Mar. 2003. DOI: 10.1109/INFCOM.2003.1209219 (cit. on p. 17).

[91] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols." In: *ACM Workshop on Wireless Security (WiSe)*. Sept. 2003. DOI: 10.1145/941311.941317 (cit. on p. 17).

[92] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. "Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks." In: *Wireless Networks* 11.1 (Jan. 2005). DOI: 10.1007/s11276-004-4744-y (cit. on pp. 17, 20).

[93] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. "Fingerprinting Wi-Fi Devices Using Software Defined Radios." In: *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. July 2016. DOI: 10.1145/2939918.2939936 (cit. on p. 155).

[94] IEEE Computer Society. *Standard for Local and Metropolitan Area Networks: Overview and Architecture*. IEEE 802. June 2014. DOI: 10.1109/IEEESTD.2014.6847097 (cit. on p. 46).

[95] IEEE Computer Society. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*. IEEE 802.11. Dec. 2016. DOI: 10.1109/IEEESTD.2016.7786995 (cit. on pp. 17, 18, 43, 45, 50, 52, 77).

[96] IEEE Standards Association. *Registration Authority*. URL: https://standards.ieee.org/products-services/regauth/index.html (retrieved Dec. 9, 2019) (cit. on p. 46).

[97] International Charter on Space and Major Disasters. *Typhoon Haiyan in the Philippines*. 2013. URL: https://disasterscharter.org/web/guest/activations/-/article/typhoon-haiyan-in-the-philippin-5 (retrieved Dec. 9, 2019) (cit. on p. 24).

[98] International Data Corporation. *The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast*. June 18, 2019. URL: https://www.idc.com/getdoc.jsp?containerId=prUS45213219 (retrieved Dec. 9, 2019) (cit. on p. 3).

[99] IRIN News. "Life-Saving Radio Begins Broadcasting in Typhoon-Hit Tacloban." In: (Nov. 15, 2013). URL: http://www.irinnews.org/report/99132/life-saving-radio-begins-broadcasting-typhoon-hit-tacloban (retrieved Dec. 9, 2019) (cit. on pp. 3, 23).

[100] Md. Shariful Islam, Md. Abdul Hamid, and Choong Seon Hong. "SHWMP: A Secure Hybrid Wireless Mesh Protocol for IEEE 802.11s Wireless Mesh Networks." In: *Transactions on Computational Science VI* (2009), pp. 95–114. DOI: 10.1007/978-3-642-10649-1_6 (cit. on p. 101).

[101] ISO/IEC JTC 1 Information Technology. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*. ISO/IEC 7498-2. Feb. 1989 (cit. on p. 13).

[102] ISO/IEC JTC 1/SC 27 Information Security, Cybersecurity and Privacy Protection. *Information technology — Security Techniques — Authenticated Encryption*. ISO/IEC 19772. Feb. 2009 (cit. on p. 16).

[103] Bounpadith Kannhavong, Hidehisa Nakayama, Yoshiaki Nemoto, Nei Kato, and Abbas Jamalipour. "A Survey of Routing Attacks in Mobile Ad Hoc Networks." In: *IEEE Wireless Communications* 14.5 (Oct. 2007), pp. 85–91. DOI: 10.1109/MWC.2007.4396947 (cit. on p. 14).

[104] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. "The ONE Simulator for DTN Protocol Evaluation." In: *ICST Conference on Simulation Tools and Techniques (Simutools)*. Mar. 2009. DOI: 10.4108/ICST.SIMUTOOLS2009.5674 (cit. on pp. 27, 132, 138, 162).

[105] Steffen Klee. "Understanding the Apple Auto Unlock Protocol." Bachelor thesis. Technische Universität Darmstadt, Nov. 2017 (cit. on p. 40).

[106] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M Frans Kaashoek. "The Click Modular Router." In: *ACM Transactions on Computer Systems (TOCS)* 18.3 (Aug. 2000), pp. 263–297. DOI: 10.1145/354871.354874 (cit. on pp. 111, 162).

[107] Florian Kohnhäuser, Milan Stute, Lars Baumgärtner, Lars Almon, Stefan Katzenbeisser, Matthias Hollick, and Bernd Freisleben. "SEDCOS: A Secure Device-to-Device Communication System for Disaster Scenarios." In: *IEEE Conference on Local Computer Networks (LCN)*. **Part of this thesis**. Extended in [172]. Oct. 2017. DOI: 10.1109/LCN.2017.47 (cit. on pp. 21, 22, 180).

[108] Constantinos Kolias, Lucas Copi, Fengwei Zhang, and Angelos Stavrou. "Breaking BLE Beacons For Fun But Mostly Profit." In: *ACM European Workshop on Systems Security*. Apr. 2017. DOI: 10.1145/3065913.3065923 (cit. on p. 155).

[109] David Kreitschmann. "User Manual for the Apple CoreCapture Framework." In: *CoRR* abs/1808.07353 (July 2018). arXiv: 1808.07353 (cit. on p. 35).

[110] John Krumm. "Inference Attacks on Location Tracks." In: *International Conference on Pervasive Computing (PERVASIVE)*. Springer Berlin Heidelberg, May 2007, pp. 127–143. ISBN: 978-3-540-72037-9 (cit. on p. 155).

[111] Alexander Kuehne, Anja Klein, Adrian Loch, and Matthias Hollick. "Corridor-based Routing Using Opportunistic Forwarding in OFDMA Multihop Networks." In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. Sept. 2012. DOI: 10.1109/PIMRC.2012.6362553 (cit. on p. 106).

[112] Denis Foo Kune, John Kölndorfer, Nicholas Hopper, and Yongdae Kim. "Location Leaks Over the GSM Air Interface." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2012. URL: https://www.ndss-symposium.org/ndss2012/location-leaks-over-gsm-air-interface (retrieved Dec. 9, 2019) (cit. on p. 155).

[113] Alfredo Mahar Francisco Lagmay et al. "Devastating storm surges of Typhoon Haiyan." In: *International Journal of Disaster Risk Reduction* 11 (Mar. 2015), pp. 1–12. DOI: 10.1016/j.ijdrr.2014.10.006 (cit. on p. 24).

[114] Jean-Claude Laprie. "From Dependability to Resilience." In: *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. June 2008. URL: http://2008.dsn.org/fastabs/dsn08fastabs_laprie.pdf (retrieved Dec. 9, 2019) (cit. on p. 14).

[115] Jennifer Leaning and Debarati Guha-Sapir. "Natural Disasters, Armed Conflict, and Public Health." In: *New England Journal of Medicine* 369.19 (Nov. 2013), pp. 1836–1842. DOI: 10.1056/NEJMra1109877 (cit. on p. 23).

[116] Feng Cheng Lee, Weihan Goh, and Chai Kiat Yeo. "A Queuing Mechanism to Alleviate Flooding Attacks in Probabilistic Delay Tolerant Networks." In: *IEEE Advanced International Conference on Telecommunications (AICT)*. May 2010. DOI: 10.1109/AICT.2010.78 (cit. on p. 130).

[117] Feng Li, Jie Wu, and Avinash Srinivasan. "Thwarting Blackhole Attacks in Disruption-Tolerant Networks Using Encounter Tickets." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2009. DOI: 10.1109/INFCOM.2009.5062170 (cit. on p. 22).

[118] Feng Li, Jie Wu, and Avinash Srinivasan. "Thwarting Blackhole Attacks in Disruption-Tolerant Networks using Encounter Tickets." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2009, pp. 2428–2436. DOI: 10.1109/INFCOM.2009.5062170 (cit. on p. 7).

[119] Qinghua Li and Guohong Cao. "Mitigating Routing Misbehavior in Disruption Tolerant Networks." In: *IEEE Transactions on Information Forensics and Security (TIFS)* 7.2 (Apr. 2012), pp. 664–675. DOI: 10.1109/TIFS.2011.2173195 (cit. on p. 7).

[120] Qinghua Li, Wei Gao, Sencun Zhu, and Guohong Cao. "To Lie or to Comply: Defending against Flood Attacks in Disruption Tolerant Networks." In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 10.3 (May 2013), pp. 168–182. DOI: 10.1109/TDSC.2012.84 (cit. on pp. 7, 21, 22).

[121] Guolong Lin and Guevara Noubir. "On Link Layer Denial of Service in Data Wireless LANs." In: *Wireless Communications and Mobile Computing* 5.3 (May 2005), pp. 273–284. DOI: 10.1002/wcm.221 (cit. on pp. 15, 78).

[122] Kejun Liu, Jing Deng, Pramod K. Varshney, and Kashyap Balakrishnan. "An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in MANETs." In: *IEEE Transactions on Mobile Computing (TMC)* 6.5 (May 2007), pp. 536–550. DOI: 10.1109/TMC.2007.1036 (cit. on p. 20).

[123] Zongqing Lu, Guohong Cao, and Thomas La Porta. "Networking Smartphones for Disaster Recovery." In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Mar. 2016. DOI: 10.1109/PERCOM.2016.7456503 (cit. on p. 3).

[124] Adriano Di Luzio, Alessandro Mei, and Julinda Stefa. "Mind Your Probes: De-anonymization of Large Crowds Through Smartphone WiFi Probe Requests." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2016. DOI: 10.1109/INFOCOM.2016.7524459 (cit. on p. 155).

[125] Aanchal Malhotra, Isaac E. Cohen, Erik Brakke, and Sharon Goldberg. "Attacking the Network Time Protocol." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2016. DOI: 10.14722/ndss.2016.23090 (cit. on p. 98).

[126] Sathiamoorthy Manoharan. "On GPS Tracking of Mobile Devices." In: *IEEE International Conference on Networking and Services (ICNS)*. Apr. 2009. DOI: 10.1109/ICNS.2009.103 (cit. on p. 155).

[127] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. "InternalBlue – Bluetooth Binary Patching and Experimentation Framework." In: *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*. June 2019, pp. 79–90. DOI: 10.1145/3307334.3326089 (cit. on p. 38).

[128] Abraham Martín-Campillo, Jon Crowcroft, Eiko Yoneki, and Ramon Martí. "Evaluating Opportunistic Networks in Disaster Scenarios." In: *Journal of Network and Computer Applications* 36.2 (Mar. 2013), pp. 870–880. DOI: 10.1016/j.jnca.2012.11.001 (cit. on p. 3).

[129] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik Rye, Brandon Sipes, and Sam Teplov. "Handoff All Your Privacy – A Review of Apple's Bluetooth Low Energy Continuity Protocol." In: *Proceedings on Privacy Enhancing Technologies (PoPETS)* 2019.4 (Oct. 2019), pp. 34–53. DOI: 10.2478/popets-2019-0057 (cit. on pp. 38, 155).

[130] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. "Defeating MAC Address Randomization Through Timing Attacks." In: *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. July 2016. DOI: 10.1145/2939918.2939930 (cit. on p. 156).

[131] Alan Meeus. "Windows 10 Mobile Security Guide." In: (Oct. 13, 2017). URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-10-mobile-security-guide (retrieved Dec. 9, 2019) (cit. on p. 129).

[132] Micro:bit Educational Foundation. *Micro:bit website*. URL: https://microbit.org (retrieved Dec. 9, 2019) (cit. on p. 88).

[133] Microsoft. *About the Wireless Ad Hoc API*. May 2018. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ms705973%28v=vs.85%29.aspx (retrieved Dec. 9, 2019) (cit. on p. 18).

[134] Arsalan Mosenia, Xiaoliang Dai, Prateek Mittal, and Niraj K. Jha. "PinMe: Tracking a Smartphone User Around the World." In: *IEEE Transactions on Multi-Scale Computing Systems* 4.3 (July 2017), pp. 420–435. DOI: 10.1109/TMSCS.2017.2751462 (cit. on p. 155).

[135] Munich RE. *TOPICS Geo Natural Catastrophes 2017*. Mar. 2018. URL: https://www.munichre.com/topics-online/en/climate-change-and-natural-disasters/natural-disasters/topics-geo-2017.html (retrieved Dec. 9, 2019) (cit. on p. 3).

[136] Munich RE. *NatCatSERVICE*. URL: https://www.munichre.com/topics-online/en/climate-change-and-natural-disasters/natural-disasters/natural-catastrophies-natcat-service-analysis-tool.html (retrieved Dec. 9, 2019) (cit. on p. 24).

[137] A. B. M. Musa and Jakob Eriksson. "Tracking Unmodified Smartphones Using Wi-fi Monitors." In: *ACM Conference on Embedded Network Sensor Systems (SenSys)*. Nov. 2012. DOI: 10.1145/2426656.2426685 (cit. on p. 155).

[138] Sashank Narain, Triet D. Vo-Huu, Kenneth Block, and Guevara Noubir. "Inferring User Routes and Locations Using Zero-Permission Mobile Sensors." In: *IEEE Symposium on Security and Privacy (S&P)*. May 2016. DOI: 10.1109/SP.2016.31 (cit. on p. 155).

[139] National Disaster Risk Reduction and Management Council. *Final Report – Effects on Typhoon "Yolanda" (Haiyan)*. Nov. 2013. URL: http://www.ndrrmc.gov.ph/attachments/article/1329/FINAL_REPORT_re_Effects_of_Typhoon_YOLANDA_(HAIYAN)_06-09NOV2013.pdf (retrieved Dec. 9, 2019) (cit. on p. 24).

[140] Sarfraz Nawaz and Cecilia Mascolo. "Mining Users' Significant Driving Routes with Low-power Sensors." In: *ACM Conference on Embedded Network Sensor Systems (SenSys)*. Nov. 2014. DOI: 10.1145/2668332.2668348 (cit. on p. 155).

[141] Samuel C. Nelson, Mehedi Bakht, and Robin Kravets. "Encounter-Based Routing in DTNs." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2009. DOI: 10.1109/INFCOM.2009.5061994 (cit. on p. 22).

[142] Nordic Semiconductor. *nRF51822*. URL: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822 (retrieved Dec. 9, 2019) (cit. on p. 87).

[143] Guevara Noubir and Guolong Lin. "Low-power DoS Attacks in Data Wireless LANs and Countermeasures." In: *SIGMOBILE Mobile Computing and Communications Review* 7.3 (July 2003), pp. 29–30. DOI: 10.1145/961268.961277 (cit. on p. 15).

[144] Panagiotis Papadimitratos and Zygmunt J. Haas. "Secure Routing for Mobile Ad Hoc Networks." In: *Communication Networks and Distributed Systems Modeling and Simulation Conference*. SCS, Jan. 2002. URL: https://infoscience.epfl.ch/record/113679 (retrieved Dec. 9, 2019) (cit. on p. 20).

[145] Panagiotis Papadimitratos and Zygmunt J. Haas. "Secure Message Transmission in Mobile Ad Hoc Networks." In: *Ad Hoc Networks* 1.1 (July 2003), pp. 193–209. DOI: 10.1016/S1570-8705(03)00018-0 (cit. on p. 20).

[146] Panagiotis Papadimitratos and Zygmunt J. Haas. "Secure Data Communication in Mobile Ad Hoc Networks." In: *IEEE Journal on Selected Areas in Communications (JSAC)* 24.2 (Feb. 2006). DOI: 10.1109/JSAC.2005.861392 (cit. on p. 100).

[147] Panagiotis Papadimitratos and Aleksandar Jovanovic. "GNSS-based Positioning: Attacks and Countermeasures." In: *IEEE Military Communications Conference (MILCOM)*. Nov. 2008. DOI: 10.1109/MILCOM.2008.4753512 (cit. on p. 98).

[148] PC Engines. *ALIX Platform*. URL: http://www.pcengines.ch/alix.htm (retrieved Dec. 9, 2019) (cit. on p. 112).

[149] PC Engines. *APU2 Platform*. URL: https://www.pcengines.ch/apu2.htm (retrieved Dec. 9, 2019) (cit. on pp. 57, 77, 112, 114).

[150] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. *Ad Hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561. IETF, July 2003. DOI: 10.17487/RFC3561 (cit. on p. 99).

[151] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Song. "Efficient Authentication and Signing of Multicast Streams over Lossy Channels." In: *IEEE Symposium on Security and Privacy (S&P)*. May 2000. DOI: 10.1109/SECPRI.2000.848446 (cit. on p. 113).

[152] Thi Ngoc Diep Pham, Chai Kiat Yeo, Naoto Yanai, and Toru Fujiwara. "Detecting Flooding Attack and Accommodating Burst Traffic in Delay-Tolerant Networks." In: *IEEE Transactions on Vehicular Technology* 67.1 (Jan. 2018), pp. 795–808. DOI: 10.1109/TVT.2017.2748345 (cit. on p. 7).

[153] Yi Ping, Hou Yafei, Zhong Yiping, Zhang Shiyong, and Dai Zhoulin. "Flooding Attack and Defence in Ad Hoc Networks." In: *Journal of Systems Engineering and Electronics* 17.2 (June 2006), pp. 410–416. DOI: 10.1016/S1004-4132(06)60070-4 (cit. on p. 15).

[154] Zhiyun Qian, Zhaoguang Wang, Qiang Xu, Z. Morley Mao, Ming Zhang, and Yi-Min Wang. "You Can Run, but You Can't Hide: Exposing Network Location for Targeted DoS Attacks in Cellular Networks." In: *Network and Distributed System Security Symposium (NDSS)*. The Internet Society, Feb. 2012. URL: https://www.ndss-symposium.org/ndss2012/you-can-run-you-cant-hide-exposing-network-location-targeted-dos-attacks-cellular-networks (retrieved Dec. 9, 2019) (cit. on p. 155).

[155] Daniele Quercia and Stephen Hailes. "Sybil Attacks Against Mobile Users: Friends and Foes to the Rescue." In: *IEEE International Conference on Computer Communications (INFOCOM)*. Mar. 2010. DOI: 10.1109/INFCOM.2010.5462218 (cit. on p. 22).

[156] Vivek Ramachandran and Sukumar Nandi. "Detecting ARP Spoofing: An Active Technique." In: *Information Systems Security*. Springer Berlin Heidelberg, 2005, pp. 239–250. DOI: 10.1007/11593980_18 (cit. on p. 16).

[157] Samsung Electronics Co. *White Paper: An Overview of Samsung KNOX*. June 2013. URL: https://image-us.samsung.com/SamsungUS/samsungbusiness/solutions/topics/iot/081717/Samsung_KNOX_whitepaper_June-0.pdf (retrieved Dec. 9, 2019) (cit. on p. 129).

[158] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. "A Secure Routing Protocol for Ad Hoc Networks." In: *IEEE International Conference on Network Protocols (ICNP)*. Nov. 2002. DOI: 10.1109/ICNP.2002.1181388 (cit. on p. 20).

[159] Piotr Sapiezynski, Arkadiusz Stopczynski, David Kofoed Wind, Jure Leskovec, and Sune Lehmann. "Inferring Person-to-Person Proximity Using WiFi Signals." In: *Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.2 (June 2017). DOI: 10.1145/3090089 (cit. on p. 155).

[160] Milan Schmittner, Arash Asadi, and Matthias Hollick. "SE-MUD: Secure Multi-hop Device-to-Device Communication for 5G Public Safety Networks." In: *IFIP Networking Conference and Workshops*. **Part of this thesis**. Extended in [171]. June 2017. DOI: 10.23919/IFIPNetworking.2017.8264846 (cit. on pp. 20, 21, 94, 95, 113, 120, 180).

[161] Milan Schmittner and Matthias Hollick. "Xcastor: Secure and Scalable Group Communication in Ad hoc Networks." In: *IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2016. DOI: 10.1109/WoWMoM.2016.7523512 (cit. on pp. 94, 95, 102, 120).

[162] Matthias Schulz, Francesco Gringoli, Daniel Steinmetzer, Michael Koch, and Matthias Hollick. "Massive Reactive Smartphone-based Jamming Using Arbitrary Waveforms and Adaptive Power Control." In: *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*. July 2017. DOI: 10.1145/3098243.3098253 (cit. on pp. 15, 78).

[163] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. "The Nexmon Firmware Analysis and Modification Framework: Empowering Researchers to Enhance Wi-Fi Devices." In: *Computer Communications* 129 (Sept. 2018), pp. 269–285. DOI: 10.1016/j.comcom.2018.05.015 (cit. on pp. 52, 57, 78).

[164] Robert W. Shirey. *Internet Security Glossary, Version 2*. RFC 4949. IETF, Aug. 2007. DOI: 10.17487/RFC4949 (cit. on p. 13).

[165] Sergei Skorobogatov. "The Bumpy Road Towards iPhone 5c NAND Mirroring." In: *CoRR* abs/1609.04327 (Sept. 2016). arXiv: 1609.04327 (cit. on p. 127).

[166] Daniel Steinmetzer, Milan Stute, and Matthias Hollick. "TPy: A Lightweight Framework for Agile Distributed Network Experiments." In: *International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH)*. ACM, Nov. 2018. DOI: 10.1145/3267204.3267214 (cit. on p. 114).

[167] Hannah Strange. "Super Typhoon Haiyan smashes into Philippines." In: *The Telegraph* (Nov. 8, 2013). URL: https://www.telegraph.co.uk/news/worldnews/asia/philippines/10434846/Super-Typhoon-Haiyan-smashes-into-Philippines.html (retrieved Dec. 9, 2019) (cit. on p. 24).

[168] Milan Stute. *proxAWDL: simple AWDL–TCP proxy*. 2018. URL: https://github.com/seemoo-lab/proxawdl (retrieved Dec. 9, 2019) (cit. on p. 62).

[169] Milan Stute. *Video of Proof-of-Concept Denial-of-Service Attack Crashing iOS Devices*. Oct. 27, 2018. URL: https://youtu.be/M5D9NeKapUo (retrieved Dec. 9, 2019) (cit. on p. 89).

[170] Milan Stute. *Video of Proof-of-Concept Man-in-the-Middle Attack on AirDrop*. May 15, 2019. URL: https://youtu.be/5T7Qatoh0Vo (retrieved Dec. 9, 2019) (cit. on p. 82).

[171] Milan Stute, Pranay Agarwal, Abhinav Kumar, Arash Asadi, and Matthias Hollick. "LIDOR: A Lightweight DoS-Resilient Communication Protocol for Safety-Critical IoT Systems." In: *IEEE Internet of Things Journal (IoT-J)* (submitted). **Part of this thesis**. Extended from [160] (cit. on pp. 20–22, 179).

[172] Milan Stute, Florian Kohnhäuser, Lars Baumgärtner, Lars Almon, Stefan Katzenbeisser, Matthias Hollick, and Bernd Freisleben. "RESCUE: A Resilient and Secure Device-to-Device Communication Framework for Emergencies." In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* (submitted). **Part of this thesis**. Extended from [107] (cit. on p. 173).

[173] Milan Stute, David Kreitschmann, and Matthias Hollick. "Demo: Linux Goes Apple Picking: Cross-Platform Ad hoc Communication with Apple Wireless Direct Link." In: *ACM Conference on Mobile Computing and Networking (MobiCom)*. Best Demo Award. **Part of this thesis**. Oct. 2018. DOI: 10.1145/3241539.3267716 (cit. on p. 18).

[174] Milan Stute, David Kreitschmann, and Matthias Hollick. "One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol." In: *ACM Conference on Mobile Computing and Networking (MobiCom)*. Best Community Paper Award. **Part of this thesis**. Oct. 2018. DOI: 10.1145/3241539.3241566 (cit. on pp. 18, 159).

[175] Milan Stute, Max Maass, Tom Schons, and Matthias Hollick. "Reverse Engineering Human Mobility in Large-scale Natural Disasters." In: *ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. **Part of this thesis**. Nov. 2017. DOI: 10.1145/3127540.3127542 (cit. on pp. 23, 25, 138).

[176] Telecommunication Standardization Sector of International Telecommunication Union. *Definitions of Terms Related to Quality of Service*. Recommendation ITU-T E.800. Sept. 2008. URL: https://www.itu.int/rec/T-REC-E.800-200809-I (retrieved Dec. 9, 2019) (cit. on p. 13).

[177] Sacha Trifunovic, MacIej Kurant, Karin Anna Hummel, and Franck Legendre. "Preventing Spam in Opportunistic Networks." In: *Computer Communications* 41 (Mar. 2014), pp. 31–42. DOI: 10.1016/j.comcom.2013.12.003 (cit. on p. 22).

[178] Trusted Computing Group. *TPM 2.0 Mobile Reference Architecture Specification*. Dec. 2014. URL: https://www.trustedcomputinggroup.org/tpm-2-0-mobile-reference-architecture-specification/ (cit. on p. 129).

[179] Galini Tsoukaneri, George Theodorakopoulos, Hugh Leather, and Mahesh K. Marina. "On the Inference of User Paths from Anonymized Mobility Data." In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. Mar. 2016. DOI: 10.1109/EuroSP.2016.25 (cit. on p. 155).

[180] United States Census Bureau. *Frequently Occurring Surnames from the 2010 Census*. Dec. 27, 2016. URL: https://www.census.gov/topics/population/genealogy/data/2010_surnames.html (retrieved Dec. 9, 2019) (cit. on p. 152).

[181] United States Social Security Administration. *Popular Baby Names: Beyond the Top 1000 Names*. URL: https://www.ssa.gov/oact/babynames/limits.html (retrieved Dec. 9, 2019) (cit. on p. 152).

[182] Jayakrishnan Unnikrishnan and Farid Movahedi Naini. "De-anonymizing Private Data by Matching Statistics." In: *IEEE Allerton Conference on Communication, Control, and Computing*. Oct. 2013. DOI: 10.1109/Allerton.2013.6736722 (cit. on p. 155).

[183] US Department of Commerce. *US Census Bureau*. URL: https://www.census.gov (retrieved Dec. 9, 2019) (cit. on p. 152).

[184] Pierre B. Vandwalle, Tashbeeb Haque, Andreas Wolf, and Saravanan Balasubramaniyan. "Method and Apparatus for Cooperative Channel Switching." In: *U.S. Patent* 9491593 (Nov. 2016). URL: http://www.google.com/patents/US9491593 (cit. on pp. 32, 44).

[185] Pierre B. Vandwalle, Christiaan A. Hartman, Robert Stacey, Peter N. Heerboth, and Tito Thomas. "Synchronization of Devices in a Peer-to-Peer Network Environment." In: *U.S. Patent* 9473574 (Oct. 2016). URL: http://www.google.com/patents/US9473574 (cit. on pp. 32, 48).

[186] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. "Why MAC Address Randomization Is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms." In: *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. May 2016. DOI: 10.1145/2897845.2897883 (cit. on p. 156).

[187] Verizon. *Data Breach Investigations Report*. 2019. URL: https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf (retrieved Dec. 9, 2019) (cit. on p. 3).

[188] vit9696. *Lilu: Arbitrary Kext and Process Patching on macOS*. URL: https://github.com/acidanthera/Lilu (retrieved Dec. 9, 2019) (cit. on p. 36).

[189] Stefan G. Weber, Yulian Kalev, Sebastian Ries, and Max Mühlhäuser. "MundoMessage: Enabling Trustworthy Ubiquitous Emergency Communication." In: *ACM International Conference on Ubiquitous Information Management and Communication*. Feb. 2011. DOI: 10.1145/1968613.1968649 (cit. on p. 21).

[190] Te-En Wei, Albert B. Jeng, Hahn-Ming Lee, Chih-How Chen, and Chin-Wei Tien. "Android Privacy." In: *IEEE Conference on Machine Learning and Cybernetics*. July 2012. DOI: 10.1109/ICMLC.2012.6359654 (cit. on p. 155).

[191] Alex Wiesmaier, Moritz Horsch, Johannes Braun, Franziskus Kiefer, Detlef Hhnlein, Falko Strenzke, and Johannes Buchmann. "An Efficient Mobile PACE Implementation." In: *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*. Mar. 2011. DOI: 10.1145/1966913.1966936 (cit. on p. 129).

[192] Hao Wu, Weiwei Sun, and Baihua Zheng. "Is Only One GPS Position Sufficient to Locate You to the Road Network Accurately?" In: *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. Sept. 2016. DOI: 10.1145/2971648.2971702 (cit. on p. 155).

[193] Yuan Xue and Klara Nahrstedt. "Providing Fault-Tolerant Ad Hoc Routing Service in Adversarial Environments." In: *Wireless Personal Communications* 29.3 (June 2004), pp. 367–388. DOI: 10.1023/B:WIRE.0000047071.75971.cd (cit. on pp. 20, 21).

[194] Fei Ye, Matthew Adams, and Sumit Roy. "V2V Wireless Communication Protocol for Rear-End Collision Avoidance on Highways." In: *IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2008. DOI: 10.1109/ICCW.2008.77 (cit. on p. 4).

[195] Hu Yih-Chun and Adrian Perrig. "A Survey of Secure Wireless Ad Hoc Routing." In: *IEEE Security & Privacy (S&P)* 2.3 (May 2004), pp. 28–39. DOI: 10.1109/MSP.2004.1 (cit. on pp. 7, 14).

[196] Joseph Yiu. *ARMv8-M Architecture Technical Overview*. No longer available. 2015. URL: https://community.arm.com/docs/DOC-10896 (cit. on p. 129).

[197] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. "SybilLimit: A Near-Optimal Social Network Defense Against Sybil Attacks." In: *IEEE/ACM Transactions on Networking (TON)* 18.3 (June 2010), pp. 885–898. DOI: 10.1109/TNET.2009.2034047 (cit. on p. 22).

[198] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. "SybilGuard: Defending Against Sybil Attacks via Social Networks." In: *SIGCOMM Computer Communication Review* 36.4 (Aug. 2006), pp. 267–278. DOI: 10.1145/1151659.1159945 (cit. on pp. 22, 134).

[199] Manel Guerrero Zapata and Nadarajah Asokan. "Securing Ad Hoc Routing Protocols." In: *ACM Workshop on Wireless Security (WiSe)*. Sept. 2002. DOI: 10.1145/570681.570682 (cit. on p. 20).

[200] Saman Taghavi Zargar, James Joshi, and David Tipper. "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks." In: *IEEE Communications Surveys & Tutorials* 15.4 (Mar. 2013), pp. 2046–2069. DOI: 10.1109/SURV.2013.031413.00127 (cit. on p. 4).

[201] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. "WiWho: Wifi-based Person Identification in Smart Spaces." In: *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. Apr. 2016. DOI: 10.1109/IPSN.2016.7460727 (cit. on p. 155).

[202] Jin Zhang, Bo Wei, Wen Hu, and Salil S. Kanhere. "WiFi-ID: Human Identification Using WiFi Signal." In: *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. May 2016. DOI: 10.1109/DCOSS.2016.30 (cit. on p. 155).

[203] Xiaolan Zhang, Giovanni Neglia, Jim Kurose, and Don Towsley. "Performance Modeling of Epidemic Routing." In: *Computer Networks* 51.10 (July 2007). DOI: 10.1016/j.comnet.2006.11.028 (cit. on pp. 123, 124).

[204] Jing Zheng, Qi Li, Guofei Gu, Jiahao Cao, David KY Yau, and Jianping Wu. "Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis." In: *IEEE Transactions on Information Forensics and Security (TIFS)* 13.7 (July 2018), pp. 1838–1853. DOI: 10.1109/TIFS.2018.2805600 (cit. on pp. 10, 121).

ERKLÄRUNG ZUR DISSERTATIONSSCHRIFT

gemäß § 9 der Allgemeinen Bestimmungen der Promotionsordnung der Technische Universität Darmstadt vom 12. Januar 1990 (ABI. 1990, S. 658) in der Fassung der 8. Novelle vom 1. März 2018

Hiermit versichere ich, Milan Stute, die vorliegende Dissertationsschrift ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Eigenzitate aus vorausgehenden wissenschaftlichen Veröffentlichungen werden in Anlehnung an die Hinweise des Promotionsausschusses Fachbereich Informatik zum Thema „Eigenzitate in wissenschaftlichen Arbeiten" (EZ-2014/10) in Kapitel „*Collaborations and My Contribution*" auf Seiten xxiii bis xxiv gelistet. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. In der abgegebenen Dissertationsschrift stimmen die schriftliche und die elektronische Fassung überein.

Darmstadt, 3. Januar 2020

Milan Stute