# Quantifying and Mitigating Privacy Threats in Wireless Protocols and Services

Jeffrey Anson Pang

CMU-CS-09-145

July 2009

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Srinivasan Seshan, Chair
Adrian Perrig
Peter Steenkiste
David Wetherall, University of Washington

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.*



This research was sponsored by the National Science Foundation under grant numbers CNS-0435382, CNS-0520192, CNS-0721857 and the US Army Research Office under grant number DAAD19-02-1-0389.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

**Keywords:** privacy, security, anonymity, wireless, mobility, tracking, Wi-Fi, 802.11

*For my parents and Ah Mah.*

# Abstract

The ubiquity of mobile wireless devices greatly magnifies the threats of clandestine physical tracking, profiling, and surveillance. This is because these devices often reveal their identities and locations to third parties, either inadvertently to eavesdroppers nearby or in reports to location-based services.

In this dissertation, we address the challenges in building practical wireless protocols and services that protect users from these threats. To understand the nature of the problem, we first quantify how easily eavesdroppers can track devices that use 802.11, the dominant local area wireless protocol for the foreseeable future. Using wireless traffic from hundreds of real devices, we show that eavesdroppers can track 802.11 devices accurately even if explicit identifiers, such as MAC addresses, are changed over time. This is because *implicit identifiers*, or identifying characteristics of 802.11 traffic, can still identify many users with high accuracy. We develop an automated procedure that can identify users even when countermeasures, such as pseudonyms and encryption, are employed.

In response to these shortcomings, we present the design and evaluation of an 802.11-like wireless link layer protocol that obfuscates *all* transmitted bits, rather than select fields, to increase privacy. By obscuring all bits, we greatly increase the difficulty of identifying or profiling users from their transmissions. Our design, called SlyFi, is nearly as efficient as existing schemes for discovery, link setup, and data delivery because transmission requires only symmetric key encryption and reception requires a table lookup followed by symmetric key decryption. Experiments using our implementation on Atheros 802.11 drivers show that SlyFi performs comparably with 802.11 using WPA.

Finally, we demonstrate how to build wireless service directories that can not track users who submit location-aware reports. This problem is increasingly relevant for 802.11 hotspot directories, which may rely on users that submit accurate information about hotspot location and characteristics but want to remain anonymous. We present Wifi-Reports, a location-based service that provides Wi-Fi clients with historical information about AP location, performance, and application support. Wifi-Reports addresses two conflicting goals: preserving the privacy of users' reports and limiting fraudulent reports.

Our contributions demonstrate that future wireless protocols and services need not sacrifice users' privacy in order to be practical.

# Acknowledgments

Useful research can only be done by standing on the shoulders of some and holding the hands of others. This dissertation is no exception and I will be forever indebted to the colleagues, mentors, friends, and family members who provided invaluable technical contributions, advise, and inspiration.

First and foremost, I thank my advisor, Srini Seshan, who both directed my research with a sure hand and let me have the freedom to pursue whatever most pleased me. I know that not every graduate student can say that their advisor provided a working environment that was generally free of stress and politics; I can about Srini. This dissertation would not have been possible without his guidance, inspiration, and criticisms.

Secondly, I thank those that contributed directly to some of the technical content in this dissertation. Ben Greenstein and Damon McCoy were constant collaborators on nearly every part of this thesis and contributed working code, experiments, and domain expertise. Furthermore, I thank Ramki Gummadi, Michael Kaminsky, Yoshi Kohno, Bryan Parno, and David Wetherall for contributing invaluable insights that influenced the technical direction of this dissertation. I also thank my other thesis committee members Adrian Perrig and Peter Steenkiste for giving me valuable feedback. Numerous other people have given me useful feedback, including Dave Andersen, Michael Buettner, Ramón Cáceres, Jason Flinn, Dongsu Han, Jaeyeon Jung, Xi Lu, Pratyusa Manadhata, Sergiu Nedevschi, Samir Sapra, Vyas Sekar, Anmol Sheth, Justin Weisz, and the anonymous reviewers from HotOS 2007, HotNets 2007, Mobicom 2007, Mobisys 2008, and Mobisys 2009.

Thirdly, I want to thank my main collaborators in my other research efforts for helping to shape my own research vision: Aditya Akella, Ashwin Bharambe, John Douceur, James Hendricks, Jay Lorch, Bruce Maggs, Anees Shaikh, and Haifeng Yu. I want to thank Aditya and Ashwin in particular for letting me tag along on their research projects at the start of my research career.

Fourthly, I must acknowledge my fortune in being able to perform my work at Carnegie Mellon University (CMU). There are few other premiere universities where I could have

done this work while being so oblivious to the day-to-day chores performed behind the scenes to ensure a graduate student's survival. For that, I am indebted to Sharon Burks, Deborah Cavlovich, Catherine Copetas, Barbara Grandillo, and the rest of the excellent staff and faculty at CMU. In addition, much of my dissertation work was done while I was employed by Intel Research in Seattle. Therefore, it is incumbent on me to also thank Intel for paying me a generous salary and providing equipment during the months of the year when I was away from CMU.

Finally, I thank my family and friends for helping me get through my six years in Pittsburgh.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Privacy is widely recognized as one of the most important unresolved issues associated with information technologies. For example, a recent CRA report identified "security that users can understand and privacy that they can control" as a grand challenge for Trustworthy Computing [44]. Similarly, the Federal Trade Commission identified privacy as central to its consumer protection mission in light of modern computer and communications technologies [54]. Meanwhile, significant public outcry has followed privacy incidents involving network devices and services (e.g., AOLs release of poorly anonymized Web search queries [15], Intel's introduction of Processor Serial Numbers [6], and Benetton's addition of RFID into their clothing [26]).

In the context of network architecture, the protection of *information privacy* [164] has primarily been achieved by maintaining the confidentiality of messages. That is, no entity other than the sender or intended recipient can access the contents of a message and the sensitive information therein. Much effort has concentrated on developing end-to-end and link-layer protocols that encrypt messages to provide this form of privacy (along with other security properties). The result has been many techniques that make the recovery of the contents of encrypted messages computationally infeasible, even under strong assumptions, such as an attacker with prior knowledge of the encryption algorithm. On the whole, these techniques have been extremely successful (despite occasional problems [30, 34]) and protocols that use these techniques, such as IPSEC, SSL, and WPA, are now in common use.

However, the emergence of ubiquitous computing devices raises new privacy concerns not addressed by these techniques. A sample of these devices includes headsets, game controllers, mobile phones, laptops, cameras, audio and video players, and specialized devices such as blood pressure monitors with wireless communications. These devices are

networked and are rapidly becoming integral parts of our daily lives — people, cars, and homes may now have many of these devices to provide rich and seamless network connectivity. Unfortunately, three properties of this class of devices raise additional threats:

**Mobility of Devices.** Unlike desktop computers, which served as the primary interfaces for network access during most of the previous 30 years, these devices are often carried with their users as they go about their daily lives. Moreover, many, such as smart phones and wireless sensors, are used to connect to the Internet and to each other in an opportunistic manner, often without requiring user input. This mode of network access implies that devices that identify themselves to third parties during communication enable those parties to track them as they move from place to place, even if the remainder of their communication is encrypted.

**Wireless Communication.** These devices typically connect to the Internet and to each other via a wide range of wireless links, such as Bluetooth, 802.11, WiMAX, Zigbee and GSM/UMTS. Wireless links are more exposed than their wired counterparts because transmissions are broadcast and can be received by anyone within radio range (e.g., about 250 meters with standard 802.11b radios [104]). Without sophisticated wiretapping hardware or access to network cables, third parties that are not intended recipients may eavesdrop on conversations using only commodity radios and off-the-shelf software. For example, any nearby observer can intercept unique low-level identifiers that are always sent unencrypted, such as Bluetooth and 802.11 device addresses. Therefore, third entity eavesdroppers can easily detect the presence of their devices and follow them from place to place. Several tracking networks for Bluetooth and 802.11 have already been deployed in the wild [33, 102, 103].

**Location-based Services.** The mobility of networked devices has also given rise to a large number of *location-based services* to which users may explicitly report their locations in order to obtain local information. For example, the iPhone and Android smart phones may send a user's location to Google to obtain driving directions; a number of smart phone applications report users' current locations in order to retrieve information about nearby restaurants and other physical establishments. These services, therefore, also have the ability to physically track clients that use them.

These three properties significantly magnify the threats of clandestine physical tracking and profiling because they give a number of entities the ability to collect information that ties a person to his or her past and current locations. Therefore, our use of ubiquitous computing devices poses threats to our *location privacy* even if the confidentiality of messages is maintained. A growing body of legal and social analyses argue that these threats are significant, as location privacy (as a form of contextual integrity) is an important foun-

dation for freedom, human relationships, and liberal democracy [123]. The erosion of location privacy has, thus, drawn the concern of the popular media [29, 168], the United States government [109, 169], and technical standards bodies [81]. To protect location privacy, users must be able to control when and with whom their location information is shared.

## 1.1 Location Privacy Threats

The primary threat to location privacy comes from the collection of *location traces*. A location trace is a set of location samples known to be from the same device or user. In addition to revealing where a user has been, researchers have shown that outdoor location traces (e.g., from GPS samples) can be used to infer a person's mode of transportation [130] and predict where they will drive [59, 98, 99]. Indoor location traces can be used to infer group gatherings [120], and character traits [112] (e.g., age, work role, smoker, coffee drinker).

In order to collect a location trace about a device, an entity needs only the ability to record the device's *location* and the ability to assign a consistent *identity* to the location measurements that distinguishes them from those recorded for other devices. The precise definitions of location and identity will shape the nature of location privacy threats. In this dissertation we typically define a location to be a spatial point measured with an accuracy of 100 to 250 feet (the local area wireless transmission range). Knowledge of such a location is usually sufficient to answer questions about whether a person visited a physical establishment (e.g., "Was Alice at Bob's house," "Was Alice at the clinic," and "Was Alice at the 4th floor office?"). Since we are primarily concerned with location traces, we define identity to be any identifier, with respect to a set of location measurements, that is consistent and distinguishing over time. For example, the MAC address of wireless traffic collected at a location is an identity because it does not change over time and is globally unique.

It is important to note that the ability to capture location traces is concerning even if these identities are not explicitly tied to user identities (e.g., a person's name). This is because a small amount of location context can be used to implicitly connect device identifiers to user identities [27, 62, 64, 74, 75, 97]. For example, the identity of a device used in a user's home can be inferred to be tied to a person that lives there. Thus, device tracking effectively enables user tracking. Even if a device can not be tied to a unique individual, the ability to track it still poses threats. For example, a thief might want to track all the devices that have been to a high-end retail store, regardless of the identity of their owners.

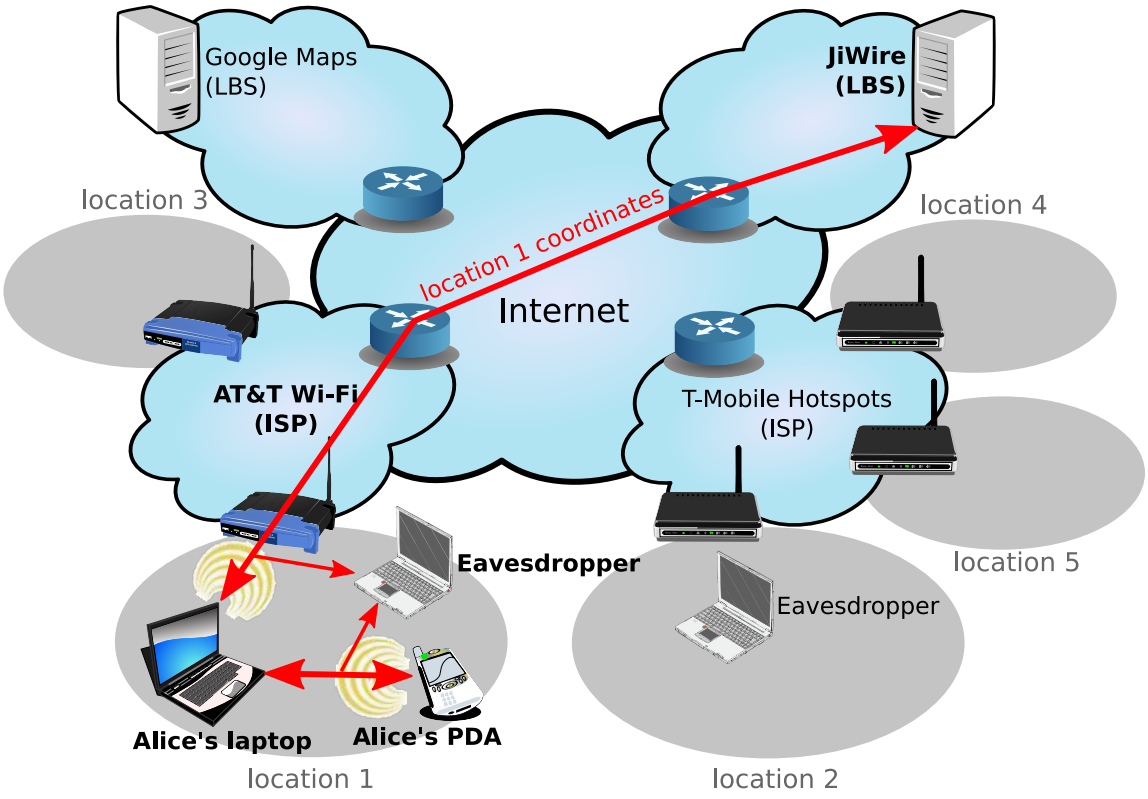


Figure 1.1: Entities that can detect the location and identity of Alice's devices (bold).

### 1.1.1 A Taxonomy of Location Privacy Threats

The three properties of ubiquitous computing devices we discussed above imply that there are a number of entities that may collect location traces about us. In this section, we describe the types of entities that can obtain both location and identity information, how they can do so, and whether they pose significant location privacy threats. We illustrate this taxonomy of entities by considering an example involving a user Alice and her devices (Figure 1.1).

**Internet Service Providers.** When Alice wants to connect her laptop the Internet, the laptop must rendezvous with a wireless access point nearby, such as a Wi-Fi access point (AP) or GSM base station. This access point may be controlled by an individual (e.g., the Wi-Fi AP in Alice's home) or by an access network *Internet Service Provider* (ISP) (e.g., the Wi-Fi APs on a campus network). Since local area wireless communications have a

range of only tens of meters, a Wi-Fi AP can infer that Alice's laptop is nearby, thereby learning its rough location. Since Alice's laptop typically must authenticate itself to access points before obtaining connectivity, it also learns her identity. Access network ISPs, thus, have the ability to collect location traces of their users. The scope of these traces will, of course, be limited to where ISPs have access points and when users connect.

Location information and identity may also be transmitted as part of Alice's communications subsequent to obtaining connectivity. However, end-to-end encryption that protects the confidentiality of messages can prevent ISPs from observing this information. Moreover, as we describe in the next section, economic incentives and legal deterrents often exist that are sufficient to prevent ISPs from revealing our location traces under normal circumstances, so technical countermeasures may not be necessary.

**Ad-hoc devices.** Alice's laptop may also communicate locally with other wireless devices such as her PDA without going through an access network, or to a private access point that itself is mobile [133]. We call this *ad-hoc* communication. Ad-hoc communication is typically used between Bluetooth devices and between some Wi-Fi devices. These devices can obviously detect that Alice's laptop is nearby due to the limited range of local area wireless connectivity. However, Alice will probably only authorize this communication if she owns all such devices or trusts their owners. Moreover, the scope of traces collected by ad-hoc devices are limited to when and where such devices communicate. Ad-hoc communication is typically only used in social settings (e.g., to exchange music files or play games) where device owners already have established social trust relationships. Therefore, technical countermeasures may not be necessary to prevent ad-hoc devices from collecting and abusing location traces.

**Location-based services.** Once a connection is established, Alice's laptop will likely communicate with services on the Internet. These services can obtain the location of Alice's laptop in a number of ways: the IP address that the access ISP assigns to it may reveal coarse geographic location (typically to within 10s of kilometers [92]) or the client may explicitly measure its own location (e.g., using GPS) and send it to the service as part of a query. The later is typically the case with *location-based services* (LBSes), such as a service that tells Alice the restaurants that are nearby. If Alice identifies herself to these services before using them (e.g., by logging in), they then also obtain her identity. Therefore, services that record these two pieces of information also have the ability to collect location traces on their users. The scope of these traces may not be limited to a small number of locations because clients may use these services wherever they have network access. However, an LBS can only collect a location sample when a user makes a query to the service and they can not ensure that all locations in queries are actually visited by

users. Nonetheless, as we describe in the next section, a class of LBSes that are also recommender systems may need to collect location samples more frequently and accurately in order to be effective. Therefore, technical countermeasures to protect location traces in these systems may be needed.

**Wireless eavesdroppers.** Independent of whether her devices communicate through an access point or in an ad-hoc fashion with each other, eavesdroppers nearby can overhear her devices' transmissions with proper radio equipment. These eavesdroppers, like access points and ad-hoc devices, can infer location due to the limited range of wireless transmissions. Furthermore, eavesdroppers can record device identities because they can observe low-level identifiers such as addresses and network names in transmissions.

For example, it is trivial to track an 802.11 device today since each device advertises a globally unique and persistent MAC address with every frame that it transmits. More generally, 802.11 facilitates user tracking and inventorying attacks that are conceptually identical to RFID threats [88], which have prompted much public concern over privacy. The low cost of 802.11 and Bluetooth hardware and ease of access to network monitoring software—all that is required for someone to locate others nearby and eavesdrop on their traffic—enable anyone to track users, from government surveillance networks to ad hoc scanners set up by thieves and curious individuals (e.g., [33, 102, 103]). Although the scope of location traces collected by these entities is limited to the locations where they have monitors deployed, clients may be secretly tracked whenever they use a wireless device if a monitor is nearby. Finally, we note that the collection of location traces by eavesdroppers need not be for malicious purposes. For example, it is often necessary to collect wireless traces for diagnostic purposes. Such traces can effectively be combined to form location traces because device identities are exposed. Since eavesdroppers may have economic incentives to collect location traces (e.g., to profile users), we argue that technical countermeasures are needed to prevent eavesdroppers from collecting them.

### 1.1.2 Legal and Economic Deterrents

ISPs typically do not need to collect location traces on their users in order to operate. Since users enter into service level agreements with network providers and services that they use, ISPs and LBSes have a strong incentive to protect their customer's privacy, lest their customers switch to a competing ISP that protects user privacy better. Moreover, government actors are prevented from arbitrarily procuring this data by law the United States [118, 157] and by the constitution of many European countries. In addition, tight regulation of licensed spectrum used by cellular networks has made it difficult for entities

other than ISPs to obtain the eavesdropping equipment for cellular protocols such as GSM.

Most LBSes only answer location-aware queries submitted by users (e.g., "Find the restaurants near this location"), and, thus, do not have an explicit need to record user identities and locations. We call any service that accepts location-based queries a *query-only LBS*. Since users explicitly decide to report their locations to an LBS, they can control when they allow these services to see their locations and can control the accuracy of the location reported. As with ISP location traces, location privacy threats can be further mitigated if location traces are anonymized or obfuscated (Section 1.3 surveys techniques).

**Limitations of legal and economic deterrents.** Unfortunately, due to the unlicensed nature of the 802.11 and Bluetooth radio spectrum, existing legal deterrents against mobile phone tracking may not apply to the tracking of these devices. Even if this were not the case, it would be very difficult to enforce anti-eavesdropping measures because passive eavesdropping is not easily detectable. No specialized hardware is needed to eavesdrop on 802.11 or Bluetooth, so it is also impractical to regulate the sale of such devices. Furthermore, eavesdroppers do not enter into any contractual agreements with those that they are eavesdropping on, so they do not have any explicit legal incentive to protect users' privacy. Indeed, the purpose of eavesdropping is to collect information about users' traffic, so they have a disincentive to delete or anonymize location traces.

In addition, while ISPs have incentives to protect the privacy of their own customers' privacy, their wireless networks may also act as large scale eavesdropping networks that spy on the communications of other ISPs' customers. Unfortunately, an ISP may not be substantially deterred from protecting the privacy of these users, since they have no contractual agreement with them and, indeed, may find it useful to collect location traces about them to profile competitors and develop ways to try to attract their customers. We categorize ISPs that behave in this manner as eavesdroppers.

Some LBSes are Web 2.0 services that use "crowd-sourcing" (e.g., reports submitted by users) in order to populate their database. We term such LBSes *crowd-sourced LBSes* because they are typically used to recommend services to their users. For example, there are hundreds of Wi-Fi services providers in major metropolitan areas, so AP directories, such as JiWire [85], Hotspotr [78], and WiGLE [167], are often used to locate Wi-Fi service. Some of these directories rely on crowd-sourcing to populate them because manually cataloging hotspots across many geographic areas is too burdensome. Because reports on wireless APs also implicitly reveal locations that users have been, these directories effectively collect location traces. These LBSes can not easily discard or anonymize location traces that they collect because they rely on their users for location-aware information.

Unlike LBSes that just answer location-based queries, these LBSes need to hold users

| entity type | where | when | non-technical deterrents |
|---|---|---|---|
| access network ISPs | access point locations | user uses ISP | legal, economic |
| ad-hoc devices | rendezvous locations | devices communicate | social |
| query-only LBSes | anywhere | user makes query | economic |
| crowd-sourced LBSes | anywhere | user records report | none |
| wireless eavesdroppers | monitor locations | user uses wireless | none |

Table 1.1: Summary of entities that pose location privacy threats, where and when they can track devices, and whether there are substantial non-technical deterrents against location trace collection.

accountable (i.e., by recording their identity) *and* record locations of their reports in order to provide good service. Standard techniques to anonymize or obfuscate reports would severely degrade the utility of an AP directory service, for example, because it would be hard to distinguish fraudulent and inaccurate reports. Therefore, crowd-soured LBSes actually have an incentive to collect identifiable and accurate location traces in order to improve their service. Indeed, these LBSes are most useful when its users submit reports frequently (e.g., for every Wi-Fi AP that they visit).

### 1.1.3 Desirable Technical Countermeasures

Table 1.1 summaries the entities that pose potential threats to location privacy and their properties. This dissertation focuses on technical countermeasures to those threats that are currently not substantially mitigated by legal, economic, and social deterrents, namely, those posed by wireless eavesdroppers and by crowd-sourced LBSes. Technical countermeasures for other types of entities, such as query-only LBSes, may also be desired in some circumstances, but there is already a relatively complete body of work that addresses these measures. We describe this work in the Section 1.3.

## 1.2 Thesis and Approach

The very use of wireless networks by mobile wireless devices poses risks to location privacy even if users trust ISPs that they connect to and we disregard LBSes that are not needed for network communication. In this dissertation, we seek to mitigate these location privacy threats by improving the mechanisms necessary for network communication

by ubiquitous computing devices. We make the following thesis:

> *Existing protocols and techniques that wireless devices use to discover and communicate with each other pose risks to users' location privacy. It is, however, possible to redesign these protocols and techniques to substantially mitigate location privacy threats without degrading their functionality or practicality.*

That is, we show that adversaries need only expend a small amount of resources and effort in order to track and profile devices that use existing wireless protocols. For example, eavesdroppers can track and profile users with commodity hardware and typical LBSes can track all users that login and submit information. The ease and limited cost of these tracking and profiling attacks imply that almost anyone with the motivation can threaten our location privacy. The goal of this dissertation is to develop protocols that significantly raise the bar for adversaries that wish to carry out these attacks so that only a small minority of entities can carry them out in practice. In developing these protocols, however, we also want to maintain the same functions of and achieve comparable efficiency to existing protocols, so that adopters of these protocols do not have to make difficult trade-offs.

In order to illustrate problems with and solutions for wireless protocols in general, this dissertation uses Wi-Fi (or 802.11) as a concrete case study. We use the terms Wi-Fi and 802.11 interchangeably. 802.11 is the most widely used local area wireless protocol today, is used for both infrastructure and ad hoc networks, and is deployed in devices ranging from laptops to cell phones. We examine empirical 802.11 networks and traffic to quantify location privacy threats and develop solutions that maintain the same functionality of existing 802.11 devices and networks. However, the qualitative conclusions we draw apply to the broader class of wireless protocols, networks, and services that we have discussed in this chapter.

Users of 802.11 devices typically take the following steps in order to communicate with each other and the Internet. First, when connecting to the Internet, users must *select* APs to use. Second, whether connecting to the Internet or to a nearby ad-hoc device, users' devices *rendezvous* with relevant APs and devices (i.e., discover and authenticate those nearby). Finally, users' devices *communicate* data to APs and to each other. Through their communication, user devices may *report* about local services and their environment (e.g., about the quality of the AP they are using).

**Select.** Locating and selecting an appropriate AP to use is an important part of the 802.11 usage model. This is typically achieved via out-of-band means, such as by matching the AP's network name to a trusted entity (e.g., a known home network). To facilitate AP

discovery in unknown locations, a number of hotspot directories exist that report their locations (e.g., [85, 78, 167]).

**Rendezvous.** The process of rendezvous with an AP in given area typically occurs by listening for *beacons* transmitted by APs nearby or by broadcasting wild card query *probes* and listening for responses. Once an AP is discovered, the user obtains any necessary credentials to access the AP (e.g., by paying for access), typically through out-of-band means (e.g., by obtaining a password from a web service or the AP's owner). Then, and after subsequent discovery attempts, the user's device can automatically authenticate itself to the AP.

**Communicate.** Once the device finishes setting up the connection with the AP, it can communicate with and through it. One threat to location privacy comes from eavesdroppers that can overhear the messages sent during rendezvous and communication.

**Report.** Many of the hotspot directory services described above rely on reports submitted by users to populate them because manually cataloging hotspots across many geographic areas is too burdensome. Since reports contain the location of APs that users have used in addition to user identities, another potential threat to location privacy comes from these directory services.

## 1.3 Thesis Scope

Section 1.1.3 outlined the three classes of entities that might obtain our location information. In this dissertation, we focus on techniques that mitigate threats from eavesdroppers and by those LBSes used for the discovery of wireless networks, as there are already incentives and mechanisms for ISPs and query-only LBSes to protect their customers' location information. We also address other attacks that eavesdroppers and malicious directory services might carry out, such as user profiling and inventorying.

Table 1.2 shows which entities can observe identity and location during each stage. Identity may be revealed by explicit or implicit identifiers in link-layer or higher-layer (IP or application layer) protocols. Location may be revealed by physical proximity (e.g., knowledge that a device is nearby due to its limited transmission range) or by a device explicitly reporting it.

Most prior work on protecting location privacy has focused on preventing query-only LBSes from obtaining identity, location, or both.

**Explicit identifiers in LBS queries.** The traditional mechanism to protect privacy in location traces is to replace each device identity with a *pseudonym* [132] (i.e., an identifier that can not be traced back to a device). For example, a user may sign into an LBS with a random name rather than with his real name. Unfortunately, a number of research studies have shown that using pseudonyms in location traces is often insufficient to protect location privacy because enough contextual information remains to tie a location trace to a specific individual. These inference attacks have been effective on indoor location traces [27] and outdoor GPS location traces [75, 97]. It has also been shown that a small number of location traces with high-frequency periodic samples can be disambiguated even if no two samples were linked by a pseudonym [64, 74, 166].

**Self-reported location in LBS queries.** As a consequence, there have also been a number of proposals to prevent query-only LBSes from collecting accurate location-traces from their users by adding noise to self-reported location. Location privacy defenses include having users submit queries with lower fidelity [65, 28, 72, 114, 111], with lower frequency [27, 75, 76], with added noise [97, 74, 173], or that are fake [93] to make a collected location trace too inaccurate to be useful. Krum [100] and Duckham and Kulik [53] present more complete surveys of the privacy issues surrounding query-only LBSes and location trace anonymization in general.

Work on protecting location privacy in query-only LBSes is important, but it only addresses one phase of the wireless usage model (locating service). Merely using wireless mobile devices enables undesirable parties such as eavesdroppers to collect location traces because we reveal identity and location to many other entities through our use of mobile devices. Furthermore, unlike query-only LBSes, which users can avoid if they do not provide sufficient privacy guarantees, the processes of network discovery and communication can not be avoided without rendering all network-reliant applications useless. This dissertation focuses on this more fundamental problem:

> *How we can build wireless protocols and discovery services in ways that protect location privacy?*

To mitigate location privacy threats, either identity or location must be concealed from the entities discussed in the previous section that are undeterred from collecting it: eavesdroppers and crowd-sourced LBSes, in particular. Therefore, we focus on the following three problems:

**Explicit identifiers in network protocols.** Wireless transmissions have limited range. Therefore, it is difficult to conceal physical proximity from eavesdroppers because receiving a wireless transmission implies that a device is nearby. To prevent eavesdroppers from

| | identity | | location | |
|---|---|---|---|---|
| | **link-layer** | **higher-layer** | **physical proximity** | **self-reported** |
| **locate service** | | LBS (personalization /accounting) | | LBS (filtering results) |
| **rendezvous** | eavesdroppers, ad-hoc devices, ISPs (device discovery) | ad-hoc devices, ISPs (device discovery) | eavesdroppers, ad-hoc devices, ISPs (limited wireless range) | |
| **communication** | eavesdroppers, ad-hoc devices, ISPs (message filtering) | ad-hoc devices, ISPs (message filtering) | eavesdroppers, ad-hoc devices, ISPs (limited wireless range) | |
| **report on service** | | crowd-sourced LBS (accountability) | | crowd-sourced LBS (accurate results) |

Table 1.2: How identity and location is revealed during different stages of wireless device usage. Each cell lists the entities that can observe such information and, in parenthesis, the primary reason why the information is revealed.

tracking devices, a device's transmissions must not contain any information that identifies the device. Unfortunately, nearly all wireless protocols today, including 802.11, Bluetooth, WiMAX, GSM, and RFID, transmit unique and device addresses in many of the packets transmitted during rendezvous and communication (e.g., when associating in Bluetooth and GSM and in all packets in 802.11).

**Implicit identifiers in network protocols.** Even if explicit identifiers are removed, however, other information that remains exposed in packet transmissions may reveal a device's identity. More generally, a large body of security research has demonstrated that *side-channels* such as packet sizes and inter-packet timing can be used to infer hidden information about those packets, such as their contents, the type of device that sent them, etc. Although, simple counter-measures to these side-channel attacks have been proposed, such as by adding cover-traffic and packet padding, they are heavy-weight and substantially degrade performance.

**Explicit identifiers in crowd-sourced reports.** The mechanisms that facilitate service location (e.g., finding an AP) may pose location privacy risks because they are often crowd-sourced LBSes (e.g., AP directories such as JiWire and Hotspotr). These systems are in the class of LBSes that are also *recommender systems*. That is, they take reports on localized service (e.g., reports on an AP) from user volunteers and summarize those reports to give other users recommendations nearby. The defenses against query-only LBSes discussed above can not typically be applied to location-based recommender systems because they all substantially degrade the accuracy of location information collected, rendering collected reports useless.

## 1.4 Our Contributions

We describe previous proposals that attempt to solve each of these three problems in subsequent chapters. Prior work has two main limitations that this dissertation redresses: First, previous solutions address the parts of the wireless usage model in a piecemeal fashion; therefore, applying one proposal in isolation will not sufficiently address location privacy threats in other stages of the usage model. It is a non-trivial exercise to stitch various solutions together. Secondly, no previous proposal has been shown to be practical when applied as replacements for real wireless protocols such as 802.11. This dissertation studies the practical problems and solutions in building wireless protocols and discovery services by actually implementing them. We make three major contributions:

1. We show that 802.11 eavesdroppers can track users using only logical-layer implicit identifiers. Therefore, even if we conceal obvious device and service addresses, such as MAC addresses, using pseudonyms as much previous work has proposed, eavesdroppers can still carry out practical tracking attacks using commodity hardware [127].

2. We show how to build complete and practical wireless protocols that conceal all explicit identifiers, substantially mitigating eavesdroppers' ability to detect implicit identifiers. In other words, we show that such protocols can have the same functionality and efficiency of 802.11, and also conceal all transmitted bits in messages [63, 129].

3. We show that it is feasible to build recommender LBSes for wireless service discovery that protect the location privacy of users that contribute reports, yet are also resilient to malicious users that submit fraudulent reports. We show that such systems can be practical alternatives to non-private 802.11 hotspot directories today [128].

The following three chapters presents our work for each contribution, respectively, though the analysis of empirical wireless traffic, novel protocol design, and detailed evaluation of prototype implementations.

## 1.5 Dissertation Outline

The remainder of this dissertation is organized as follows:

- Chapter 2 demonstrates that previously proposed counter measures, when applied to existing protocols such as 802.11, are insufficient to prevent eavesdroppers from collecting accurate location traces, thereby necessitating more private protocols.
- Chapter 3 demonstrates how to build complete wireless protocols that protect location privacy from eavesdroppers by presenting the design and evaluation of a practical link-layer protocol that makes it impractical for eavesdroppers from carry out tracking attacks.
- Chapter 4 demonstrates how crowd-sourced LBSes can protect location privacy in a practical manner by presenting the design and evaluation of a privacy-preserving crowd-sourced LBS for the discovery of wireless networks.
- Chapter 5 summarizes the contributions of this dissertation and outlines some open problems.

# Chapter 2

# Quantifying Tracking Threats

The best practices for securing 802.11 networks, embodied in the 802.11i standard [80], provide user authentication, service authentication, data confidentiality, and data integrity. However, they do not provide anonymity, a property essential to prevent location tracking. It is trivial to track an 802.11 device today since each device advertises a globally unique and persistent MAC address with every frame that it transmits. In response, researchers have proposed applying *pseudonyms* [66, 84] (temporary, unlinkable names) to mask this identifier, i.e., by having users periodically change the MAC addresses of their 802.11 devices.

In this chapter, we demonstrate that short-term pseudonyms are insufficient to provide prevent the tracking of 802.11 devices. Even without a unique address, other characteristics of users' 802.11 traffic can identify them implicitly and track them with high accuracy. An example of such an *implicit identifier* is the IP address of a service that a user frequently accesses, such as his or her email server. In a population of several hundred users, this address might be unique to one individual; thus, the mere observation of this IP address would indicate the presence of that user. Of course, in a wireless network that employs link-layer encryption, IP addresses would not be visible to an eavesdropper. However, other implicit identifiers would remain and these identifiers can be used in combination to identify users accurately.

This chapter quantifies how well a passive adversary can track users with four implicit identifiers visible to commodity hardware. We thereby place a *lower bound* on how accurately users can be identified implicitly, as more implicit identifiers and more capable adversaries exist in practice. More specifically, we identify four previously unrecognized implicit identifiers: network destinations, network names advertised in 802.11 probes, differing configurations of 802.11 options, and sizes of broadcast packets that hint at their

contents. We then develop an automated procedure to identify users. This procedure allows us to quantify how much information implicit identifiers, both alone and in combination, reveal about several hundred users in three empirical 802.11 traces.

Our evaluation shows that users emit highly discriminating implicit identifiers, and, thus, even a small sample of network traffic can identify them more than half (56%) of the time in public networks, on average. Moreover, we will almost never mistake them as the source of other network traffic (1% of the time). Since adversaries will obtain multiple traffic samples from a user over time, this high accuracy in traffic classification enables them to track many users with even higher accuracy in common wireless networks. For example, an adversary can identify 64% of users with 90% accuracy when they spend a day at a busy hot spot that serves 25 concurrent users each hour.

**Chapter outline.** In Section 2.1 we illustrate the power of implicit identifiers with several real examples. Section 2.2 discusses prior work on implicit identifiers. Section 2.3 explains our experimental methodology. Section 2.4 describes our empirical 802.11 traces. Section 2.5 analyzes how well 802.11 users can be identified using each implicit identifier individually. Section 2.6 examines how accurately an adversary can track people using these implicit identifiers in public, home, and enterprise networks. We conclude this chapter in Section 2.7.

## 2.1 The Implicit Identifier Problem

How significantly do implicit identifiers erode location privacy? Consider the seemingly innocuous trace of 802.11 traffic collected at the 2004 SIGCOMM conference, now anonymized and archived for public use [47]. Interestingly, hashing real MAC addresses to pseudonyms is also the best practice for anonymizing traces such as this. Unfortunately, implicit identifiers remain and they are sufficient to identify many SIGCOMM attendees. For example:

**Implicit identifiers can identify us uniquely.** One particular attendee's laptop transmitted requests for the network names "MIT," "StataCenter," and "roofnet," identifying him or her as someone probably from Cambridge, MA. This occurred because the default behavior of a Windows laptop is to actively search for the user's preferred networks by name, or Service Set Identifier (SSID). The SSID "therobertmorris" perhaps identifies this person uniquely [137]. A second attendee requested "University of Washington" and "djw." The last SSID is unique in the SIGCOMM trace and suggests that this person may be University of Washington Professor David J. Wetherall, one of our coauthors. More distressingly,

Wigle [167], an online database of 802.11 networks observed around the world, shows that there is only one "djw" network in the entire Seattle area. Wigle happens to locate this network within 192 feet of David Wetherall's home.

**Implicit identifiers remain even when counter measures are employed.** Another SIGCOMM attendee transferred 512MB of data via BitTorrent (this user contacted hosts on the typical BitTorrent port, 6881). A request for the SSID "roofnet" [140] from the same MAC address suggests that this user is from Cambridge, MA. Suppose that this user had been more stealthy and changed his or her MAC address periodically. In this particular case, since the user had not requested the SSID during the time he or she had been downloading, the MAC address used in the SSID request would have been different from the one used in BitTorrent packets. Therefore, we would not be able to use the MAC address to explicitly link "roofnet" to this poor network etiquette. However, the user does access the same SSH and IMAP server nearly every hour and was the only user at SIGCOMM to do so. Thus, this server's address is an implicit identifier, and knowledge of it enables us to link the user's sessions together.

Now suppose that the network employed link-layer encryption scheme, such as WPA, that obscures network addresses. Even then, we could link this user's sessions together by employing the fact that, of the 341 users that sent 802.11 broadcast packets, this was the only one that sent broadcast packets of sizes 239, 245, and 257 bytes and did so repeatedly throughout the entire conference. Furthermore, the identical 802.11 capabilities advertised in each session's management frames improves our confidence of this linkage because these capabilities differentiate different 802.11 cards and drivers. Prior research has shown that peer-to-peer file sharing traffic can be detected through encryption [172]. Thus, even if pseudonyms and link-layer encryption were employed, we could still implicate someone in Cambridge.

In the remainder of this chapter, we examine how the shortcomings of wireless protocols impact the location privacy of a large number of users in different 802.11 networks and demonstrate that the examples described in this section are not isolated anomalies.

## 2.2 Related Work

The body of work that examines inferring information from implicit side-channels falls into three categories: using logical layer side-channels to fingerprint devices, using logical layer side-channels to infer message contents, and using physical layer side-channels to fingerprint devices.

**Logical layer fingerprints.** Seemingly innocuous information such as packet sizes, timing, and header information serve as fingerprints that identify individual devices or classes of devices. We call such fingerprints *implicit identifiers* because they can implicitly identify the device that transmitted a sequence of messages even when no explicit identifiers, such as MAC addresses, are exposed. For example, Kohno *et al.* [95] showed that devices could be fingerprinted using the clock skew exposed by TCP timestamps. Padmanabhan and Yang [126] explored fingerprinting users with "clickprints," or the paths that users take through a website. Security tools like `nmap` [61] and `p0f` [125] leverage differences in network stack behaviors to determine a device's operating system.

Although this work demonstrates serious privacy risks with the release of IP or application-level network traces, the information these fingerprinting techniques rely on is concealed by link-layer encryption. Thus, they are less of a concern for properly secured wireless communications. There are also practical limitations that limit tracking attacks using these implicit identifiers as well. Kohno *et al.* note that one limitation of their work is that an adversary can not passively obtain timestamps from devices running the most prevalent operating system, Windows XP. For example, we find that only 15%-32% of users in send TCP timestamps in wireless traces that we analyze. Padmanabhan and Yang's techniques rely on data from many user sessions collected at actual web servers.

In contrast to these IP and application layer implicit identifiers, Franklin *et al.* [56] showed that it is possible to fingerprint device drivers using the timing of 802.11 probes. However, it is not yet clear how well individual devices can be distinguished using this implicit identifier.

By addressing these limitations in this chapter, we show that tracking wireless devices using implicit identifiers exposed in wireless protocols is practical. These three research efforts compliment our work, since the procedure we develop for identifying users enables an adversary to use these implicit identifiers in combination with ours, yielding even more accurate user fingerprints. None of these previous efforts offer a formal method to combine multiple pieces of evidence. Moreover, to our knowledge, we are the first to evaluate the how well users are identified by implicit identifiers observed in empirical wireless data.

**Logical layer message inference.** Implicit identifiers also reveal sensitive information other than device identity. Key-stroke dynamics have been shown to accurately identify users [115, 147]. The timing and sizes of Web transfers often uniquely identify websites, even when transmitted over encrypted channels [31, 150]. Finally, there has been a large body of research in identifying applications from implicit identifiers in encrypted traffic [90, 91, 116, 172, 177].

It is important to note that many of these side-channel attacks are partly enabled by

the presence of short-term connection (i.e., session) identifiers. This is because they rely on analyzing sequences of encrypted messages in individual connections. Thus, if an eavesdropper could not distinguish the messages sent in different connections, these side-channels would be much noisier and harder to extract accurate information from. We exploit this fact in our solution presented in Chapter 3.

**Physical layer fingerprints.** Physical layer information may also sometimes act as side channels that link messages together, but they are typically less accurate than logical layer fingerprints or require expensive, non-commodity hardware to detect. For example, Tao *et al.* [153] and Bahl and Padmanabhan [12] show that signal strength measurements from multiple locations can be used to distinguish the rough locations where they originated. Therefore, when devices are stationary, they can approximately distinguish the messages sent by each one. Nonetheless, Bauer *et al.* [18, 19] find that, while using multiple signal strength measurements to distinguish messages sources can be moderately accurate, the error rate is sufficiently high that certain side-channel attacks (e.g., [105]) have much lower success rates (e.g., 50% vs. 95%). Moreover, collecting sufficient physical layer fingerprints requires multiple monitoring points and their accuracy can be reduced by varying a client's transmit power [18, 19]. Patwari *et al.* [131] describe a similar location distinguishing physical layer fingerprint based on measuring a device's multipath signal propagation signature. However, it is not possible to collect these signatures with commodity hardware (software defined radios were used).

Differences in radio transients induced by manufacturing defects in different radios might also be used to fingerprint devices. Some work has shown this to be true at least for a small number of devices [16, 70, 143]. Other persistent signal and modulation differences induced by manufacturing defects can be used to accurately fingerprint devices [36]. However, all these fingerprinting techniques require expensive signal analyzers to perform, so only well-funded adversaries are likely to be able to deploy networks of these monitors at threatening scales.

This dissertation focuses on logical layer fingerprints and message inference and does not explicitly address physical layer implicit identifiers. Therefore, an eavesdropper with specialized equipment, such as expensive signal analyzers, may still be able to track users when our solutions are applied. However, it is much less likely that such expensive equipment would be deployed as part of surveillance networks rather than commodity hardware. Moreover, our solutions are a necessary first step in defending against such threats.

## 2.3 Experimental Setup

This section describes the evaluation criteria we use to determine how well several implicit identifiers can be used to track users.

**The Adversary.** Strong adversaries, such as service providers and large monitoring networks, obviously pose a large threat to our location privacy. However, the significance of the threat posed by 802.11 is that *anyone* that wishes to track users can do so.

Therefore, we consider an adversary that runs readily available monitoring software, such as `tcpdump` [156], on one or more laptops or on less conspicuous commodity 802.11 devices [165]. We further restrict adversaries by assuming that their devices listen passively. That is, they never transmits 802.11 frames, not even to associate with a network. This means that the adversary *can not be detected* by other radios. The adversary deploys monitoring devices in one or more locations in order to observe 802.11 traffic from nearby users. By considering a weak adversary, we place a lower bound on the accuracy with which users can be tracked, as stronger adversaries would be strictly more successful.

**The Environments.** An adversary's tracking accuracy will depend on the 802.11 networks he or she is monitoring. Since implicit identifiers are not perfectly identifying, it will be more difficult to distinguish users in more populous networks. In addition, different networks employ different levels of security, making some implicit identifiers invisible to an adversary. We consider the three dominant forms of wireless deployments today: public networks, home networks, and enterprise networks.

Public networks, such as hot spots or metro-area networks [117], are typically unencrypted at the link-layer. Although many public networks employ access control—for example, to allow access to only a provider's customers—most do so via authentication above the link-layer (e.g., through a web page) and by using MAC address filtering thereafter. Very few use 802.11i-compliant protocols that also enable encryption. Hence, identifying features at the network, link, and physical layers would be visible to an eavesdropper in such an environment. Unfortunately, this is the most common type of network today due to the challenge of secure key distribution.

Home and small business networks are small, but detecting when specific users are present is increasingly challenging due to the high density of access points in urban areas [8]. In addition, these networks are more likely to employ link-layer encryption, such as WEP or WPA, because the set of authorized users is typically known and is small. In cases where link-layer encryption is employed, an eavesdropper will not be able to view the payloads of data packets. However, features that are derived from frame sizes or timing, which are not masked by encryption, or from 802.11 management frames, which are

always sent in the clear, remain visible.

Finally, security conscious enterprise networks are likely to employ link-layer encryption. Moreover, if the only authorized devices on the network are provided by the company, there will be less diversity in the behavior of wireless cards. For example, Intel corporation issues similar corporate laptops to its employees. We consider a enterprise network where only one type of wireless card and configuration is in use, so users can not be identified by differences in device implementation. However, features derived from the networks that users visit or the applications and services they run remain visible.

**The Monitoring Scenario.** We assume that users use different pseudonyms during each wireless session in each of these environments, as Gruteser *et al.* [66] propose. As a result, explicit identifiers can not link their sessions together. Sessions can vary in length, so we assume that every hour, each user will have a different pseudonym. We define a *traffic sample* to be one user's network traffic observed during one hour.

Although it is possible for users to change their MAC addresses more frequently, this is unlikely to be very useful in practice because other features, such as received signal strength, can link pseudonyms together at these timescales [13, 154]. Moreover, changing a device's MAC address forces a device to re-associate with the access point and, thus, disrupts active connections. In addition, it may require users to revisit a web page to re-authenticate themselves, since MAC addresses are tied to user accounts in many public networks. Users are unlikely to tolerate these annoyances multiple times per session.

Of course, the ability to link traffic samples together does not help an adversary detect a user's presence unless the adversary is also able to link at least one sample to that user's identity. In Section 2.1, we showed that identity can sometimes be revealed by correlating implicit identifiers with out-of-band information, such as that provided by the Wigle [167] location database. However, if the adversary knows the user he wishes to track, he can likely obtain a few traffic samples known to come from that user's device. For example, an adversary could obtain such samples by physically tracking a person for a short time. We assume the adversary is able to obtain this set of *training samples* either before, during, or after the monitoring period. Our results show that on average, only 1 to 3 training samples are sufficient to track users with each implicit identifier (see Section 2.5.2). The monitor itself collects samples that the adversary wants to test, which we call *validation samples*.

**Evaluation Criteria.** There are a number of questions an adversary may wish to answer with these validation samples. Who was present? When was user $U$ present? Which samples came from user $U$? Essential to answering all these questions is the ability to classify samples by the user who generated them. In other words, given a validation sample, the

| | sigcomm | | ucsd | | apt | |
|---|---|---|---|---|---|---|
| | training | validation | training | validation | training | validation |
| Duration (hours) | 37 | 54 | 10 | 11 | 119 | 345 |
| Total Samples | 1974 | 3391 | 587 | 1240 | 638 | 1473 |
| Frames Per Sample (median) | 289 | 284 | 1227 | 1128 | 57 | 92 |
| Total Users | 377 | 412 | 225 | 371 | 97 | 196 |
| Profiled Users | 337 | 337 | 153 | 153 | 39 | 39 |
| Samples per Profiled User (mean) | 5.5 | 9.1 | 3.1 | 4.7 | 14.7 | 32.2 |
| Users per Hour (mean) | 53 | 64 | 59 | 113 | 5 | 4 |

Table 2.1: Summary of relevant workload statistics and parameters. The duration reports only hours with at least one active user.

adversary needs to answer the following question for one or more users $U$:

**Question 1** *Did this traffic sample come from user $U$?*

Section 2.5 evaluates how well an adversary can answer this question with each of our implicit identifiers.

To demonstrate how well implicit identifiers can be used for tracking, we also evaluate the accuracy in answering the following:

**Question 2** *Was user $U$ here today?*

This question is distinct from Question 1 because an adversary can observe many traffic samples at any given time, any one of which may be from the target user $U$. In addition, a single affirmative answer to Question 1 does not necessitate a affirmative answer to Question 2 because an adversary may want to be more certain by obtaining multiple positive samples. Section 2.6 details the interaction between these questions and evaluates how many users can be tracked with high accuracy in each of the 802.11 networks described above.

## 2.4 Wireless Traces

We evaluate the implicit identifiers of users in three 802.11 traces. We consider `sigcomm`, a 4 day trace taken from one monitoring point at the 2004 SIGCOMM conference [47], `ucsd`, a trace of all 802.11 traffic in U.C. San Diego's computer science building on November 17, 2006 [43], and `apt`, a 19 day trace monitoring all networks in an apartment

building, which we collected. All traces were collected with `tcpdump`-like tools and only contain information that can be collected using standard wireless cards in monitor mode. The `ucsd` trace is the union of observations from multiple monitoring points. IP and MAC addresses are anonymized but are consistent throughout each trace (i.e., there is a unique one-to-one mapping between addresses and anonymized labels). Link-layer encryption (i.e., WEP or WPA) was not employed in either the `sigcomm` or `ucsd` network and neither trace recorded application packet payloads. In our analysis, we show that implicit identifiers remain even when we emulate link layer encryption and that we do not need packet payloads to identify users accurately. The `apt` trace only recorded broadcast management packets due to privacy concerns; hence, we only use it to study the one implicit identifier that is extracted from these packets.

We distinguish unique users by their MAC address since it is not currently common practice to change it. To simulate the effect of using pseudonyms, we assume that every user has a different MAC address each hour. Hence, we have one sample per user for each hour that they are active. To simulate the training samples collected by an adversary, we split each trace into two temporally contiguous parts. Samples from the first part are used as training samples and the remainder are validation samples. We choose a training period in each trace long enough to profile a large number of users. For the `sigcomm` trace, the training period covers the time until the end of the first full day of the conference. For the `ucsd` trace, the training period covers the time until just before noon. We skip one hour between the training and validation periods so user activities at the end of the training period are less likely to carry over to the validation period. For the `apt` trace, the training period covers the first 5 days. We consider a user to be present during an hour if and only if she sends at least one data or 802.11 probe packets during that time; i.e., if the user is actively using or searching for a wireless network.[1]

Table 2.1 shows the relevant statistics about each trace. Note that since can we only compute accuracy for users that were present in both the training and validation data, those are the only users that we profile. Therefore, results in this chapter refer to 'Profiled Users' as the total user count and not 'Total Users.'

[1]We ignore samples that only contain other 802.11 management frames, such as power management polls. Including samples with these frames would not appreciably change the characteristics of the `sigcomm` workload, but would double the number of total "users" in the `ucsd` workload. This is because many devices observed in the `ucsd` trace were never actively using the network; we ignore these idle devices.

## 2.5 Implicit Identifiers

In this section, we describe four novel implicit identifiers and evaluate how much information each one reveals. Our results show that (1) many implicit identifiers are effective at distinguishing individual users and others are effective at distinguishing groups of users; (2) a non-trivial fraction of users are trackable using any one highly discriminating identifier; (3) on average, only 1 to 3 training samples are required to leverage each implicit identifier to its full effect; and (4) at least one implicit identifier that we examine accurately identifies users over multiple weeks.

### 2.5.1 Identifying Traffic Characteristics

**Network Destinations.** We first consider netdests, the set of IP <address, port> pairs in a traffic sample, excluding pairs that are known to be common to all users, such as the address of the local network's DHCP server. There are several reasons to believe that this set is relatively unique to each user. It is well known that the popularity of web sites has a Zipf distribution [35], so many sites are visited by a small number of users. In fact, in the `sigcomm` and `ucsd` training data, each <address, port> pair is visited by 1.15 and 1.20 users on average, respectively. The *set* of sites that a user visits is even more likely to be unique. In addition, users are likely to visit some of the same sites repeatedly over time. For example, a user generally has only one email server and a set of bookmarked sites they check often [155].

An adversary could obtain network addresses in any wireless network that does not enable link layer encryption. Even if users sent all their traffic through VPNs, the case for several users in the `sigcomm` trace, the IP addresses of the VPN servers would be revealing. No application or network level confidentiality mechanisms, such as SSL or IPSec, would mask this identifier either.

**SSID Probes.** Next we consider ssids, the set of SSIDs in 802.11 probes observed in a traffic sample. Windows XP and OS X add the SSID of a network to a preferred networks list when the client first associates with the network. To simplify future associations, subsequent attempts to discover *any* network will try to locate this network by transmitting the SSID in a probe request. As we observed in Section 2.1, SSID names can be distinguishing.[2] In addition, probes are never encrypted because active probing must be able to

[2] A recent patch [113] to Windows XP allows a user to disable active probing, but it remains enabled by default because disabling it would break association in networks where the access point does not announce itself. In addition, revealing probes or beacons are still required for devices to discover each other in ad hoc

occur before association and key agreement.

There are two practical issues that limit the use of ssids as an implicit identifier. First, the preferred networks list changes each time a user adds a network, and thus a profile may degrade over time. Second, clients transmit the SSIDs on their preferred networks lists only when attempting to discover service. Therefore, clients may not probe for distinguishing SSIDs very often. While this is true, our results show that when distinguishing SSIDs are probed for, they can often uniquely identify a user. Since all users in the monitoring area are likely to use the SSIDs of the networks being monitored, these SSIDs are not distinguishing and we do not include them in the ssids set.

**Broadcast Packet Sizes.** We now consider bcast, the set of 802.11 broadcast packet sizes in each traffic sample. Many applications broadcast packets to advertise their existence to other machines on the local network. Due to the nature of this function, these packets often contain naming information. For example, in our traces, we observed many Windows machines broadcasting NetBIOS naming advertisements and applications such as FileMaker and Microsoft Office advertising themselves.

Since these packets vary in length, their sizes can reveal information about their content even if the content itself is encrypted. Packet sizes alone appear to distinguish users almost as well as <application, size> tuples. For example, in the `sigcomm` trace, there are only 16% more unique tuples than unique sizes. Table 2.2 lists the most unique broadcast packet sizes we observed and the application port that generated them. Broadcast packets are sent to a known broadcast MAC address; thus, an adversary can distinguish them from other traffic even if link encryption is employed and the adversary is not granted network privileges. This set would remain identifying even when user behavior changes because most broadcast packets are emitted automatically.

Two types of broadcast packets, standard DHCP requests and power management beacons, are common to all users, since a device must send a DHCP request in order to obtain an IP address and sends power management beacons when in low power mode. Thus, we do not include these packets' sizes in the bcast set. These packets have distinct sizes (336 and 36 payload bytes, respectively) so they can be filtered even when link-layer encryption is enabled.

**MAC Protocol Fields.** Finally, we consider fields, the specific combination of 802.11 protocol fields visible in the MAC header that distinguish a user's wireless card, driver, and configuration. The fields included are the 'more fragments,' 'retry,' 'power management,' and 'order,' bits in the header, the authentication algorithms offered, and the supported

mode.

| Application | Port | Number of Sizes |
|---|---|---|
| wireless driver or OS | NA | 14 |
| DHCP | 67 | 14 |
| sunrpc | 111 | 1 |
| NetBIOS | 138 | 7 |
| groove-dpp | 1211 | 1 |
| Microsoft Office v.X | 2222 | 1 |
| FileMaker Pro | 5003 | 7 |
| X Windows | 6000 | 1 |

Table 2.2: A list of the most unique broadcast packets observed in the `sigcomm` trace. The third column shows the number of packet sizes that were emitted by at most 2 users.

transmission rates. Some card configurations can be more or less likely to emit different values in each of these fields, so they can distinguish users with different wireless cards. Although this identifier is unlikely to distinguish users uniquely, it can be combined with others to add more evidence. Moreover, many of these fields are available in any 802.11 packet, so they can almost always assist in identification. Furthermore, the likelihood of any particular field combination is unlikely to change for a user unless she obtains a new wireless device or driver; thus, fields should remain identifying over long time periods.

### 2.5.2 Evaluating User Distinctiveness

To show much information each identifier reveals, we now evaluate how accurately an adversary can answer Question 1 (see Section 2.3) using each implicit identifier.

#### Methodology

We construct a classifier $C_U$ for each user $U$ in our traces. Given a traffic sample $s$, $C_U$ returns "Yes" if it believes the sample came from user $U$ and "No" otherwise. We use a naïve Bayes classifier due to its effectiveness in application traffic classification [116, 172, 177]. More sophisticated classifiers exist, but this simple one is sufficient to demonstrate that implicit identifiers are a problem. Specifically, from each traffic sample, we extract a vector of features $(f_1, \ldots, f_m)$. In our case, $m \leq 4$, one feature per implicit identifier present in the sample. Each of our features has a different source, so we assume that they are independent. For each feature $f_i$, we estimate the posterior probability distribution $\Pr[s \text{ has } f_i | s \text{ is from } U]$ and the prior probability distribution

$\Pr[s \text{ has } f_i]$ from training data. We are interested in $\Pr[s \text{ is from } U | s \text{ has } f_1, \ldots, f_m] =$

$$\frac{\prod_i^m (\Pr[s \text{ has } f_i | s \text{ is from } U]) \cdot \Pr[s \text{ is from } U]}{\prod_i^m \Pr[s \text{ has } f_i]} .$$

We classify a sample as being from $U$ if and only if this value is greater than a threshold $T$. We also estimate the prior $\Pr[s \text{ is from } U]$ from training data, though this could also be based on a priori knowledge of how frequently the adversary believes his target will be present.

**Feature Generation.** To compute these probabilities, we must convert each of our implicit identifiers into a categorical or real-valued feature. We treat the fields identifier as a categorical feature by having each field combination represent a different value. Each of the other three identifiers is defined as a *set* of discrete *elements*; e.g., netdests is a set of network addresses. The following procedure describes how this set is converted into a real-valued feature that measures how similar it is to the target user's expected set.

We first construct a profile set, $Profile_U$, comprising all the elements in the union of all training samples for user $U$. To obtain a numeric value from the set of elements from a sample $s$, $Set_s$, we use a weighted version of the Jaccard similarity index [159] of the profile and the sample sets. The Jaccard index of two sets computes $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. However, some elements in each set are more discriminating than others (i.e., those that we observe in fewer users' traffic). Hence, we weight each element $e$ by $w(e)$, the inverse of the number of users that accessed it. We learn these weights from the training data. Hence, given the profile $Profile_U$, the feature we compute for sample $s$ is:

$$feature_U(s) = \frac{\sum_{e \in Profile_U \cap Set_s} w(e)}{\sum_{e \in Profile_U \cup Set_s} w(e)} .$$

This value quantifies how similar the set seen in the sample is to the user's profile. Since this procedure computes a real-valued feature, we estimate the probability distributions using a density estimator. We use the default estimator in the R statistical package [135], which uses multiple Gaussian kernels.

### Accuracy Metrics

Implicit identifiers are not perfectly identifying. Therefore, to evaluate Question 1, we quantify the *accuracy* of our classifier. Accuracy has two components: (1) the true positive rate (TPR), or the fraction of validation samples that user $U$ generates that we correctly

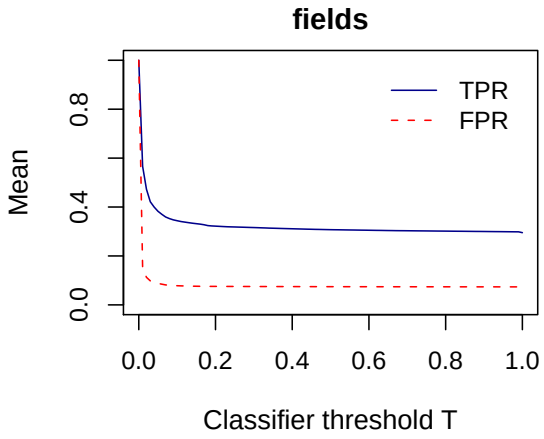


Figure 2.1: Mean TPR and FPR as the classifier threshold $T$ is varied for fields.

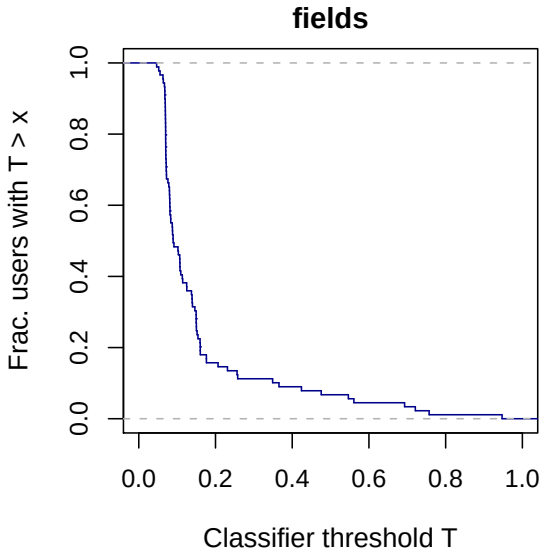


Figure 2.2: CCDF of classifier thresholds $T$ that achieve FPR = 0.01 for different users

classify, and (2) the false positive rate (FPR), or the fraction of validation samples that user $U$ does not generate that we incorrectly classify. The former tells us how often $U$'s traffic will identify her, while the later tells us how often we will mistake $U$ as the source of other traffic. We measure accuracy with TPR and FPR instead of just precision (i.e., the fraction

| | Fraction of users trainable | |
|---|---|---|
| | sigcomm | ucsd |
| netdests | 0.89 | 0.84 |
| ssids | 0.81 | 0.55 |
| bcast | 0.70 | 0.65 |
| fields | 1.00 | 1.00 |

Table 2.3: The fraction of profiled users that we could train using each feature.

of all samples classified correctly) because the vast majority of samples are negative (i.e., not from the target user). Hence, classifiers that mark a larger fraction samples as negative would score higher in precision even if they marked the same fraction of true positives incorrectly.

**Trainable Users.** When evaluating each identifier, we consider only those users that have at least one training sample that contain it, since we can't build profiles for those with no such samples. Table 2.3 shows the number of profiled users that exhibit each feature in the training period. Each implicit identifier is exhibited by a different subset of users. In both workloads, each implicit identifier is exhibited by a majority of profiled users. The fraction of users that exhibited the ssids feature is lower in the ucsd workload (55% vs 81%) because fewer users sent SSID probes to search for a network. This may be because many ucsd users already established a high preference for the UCSD network, having used it previously. sigcomm users were all new to the SIGCOMM network and initiated broader searches for their preferred networks before association.

**Classifier Thresholds.** We evaluate each classifier across several thresholds $T$ in order to determine the trade-off between TPR and FPR. As $T$ increases, FPR and TPR decrease because the classifier requires more evidence that a user is present in order to answer positively. This is exemplified in Figure 2.1 for the classifier using the fields feature. We assume that an adversary desires a target FPR, such as 1 in 100, and chooses a threshold $T$ based on that target. Ideally, the target FPR would be low. Due to variance in each user's training data, an adversary may need to use different thresholds to achieve the same FPR for different users. This is exemplified in Figure 2.2, which shows a complementary cumulative distribution function (CCDF) of thresholds that achieve FPR = 0.01 for each user's classifier using the fields feature. An adversary would train a different classifier for each user that he is tracking. In practice, an adversary would have to select $T$ without a priori knowledge of the FPR achieved on the validation data. In Section 2.6.1, we show that an adversary can select $T$ to achieve a desired FPR without this knowledge when using multiple features in combination.

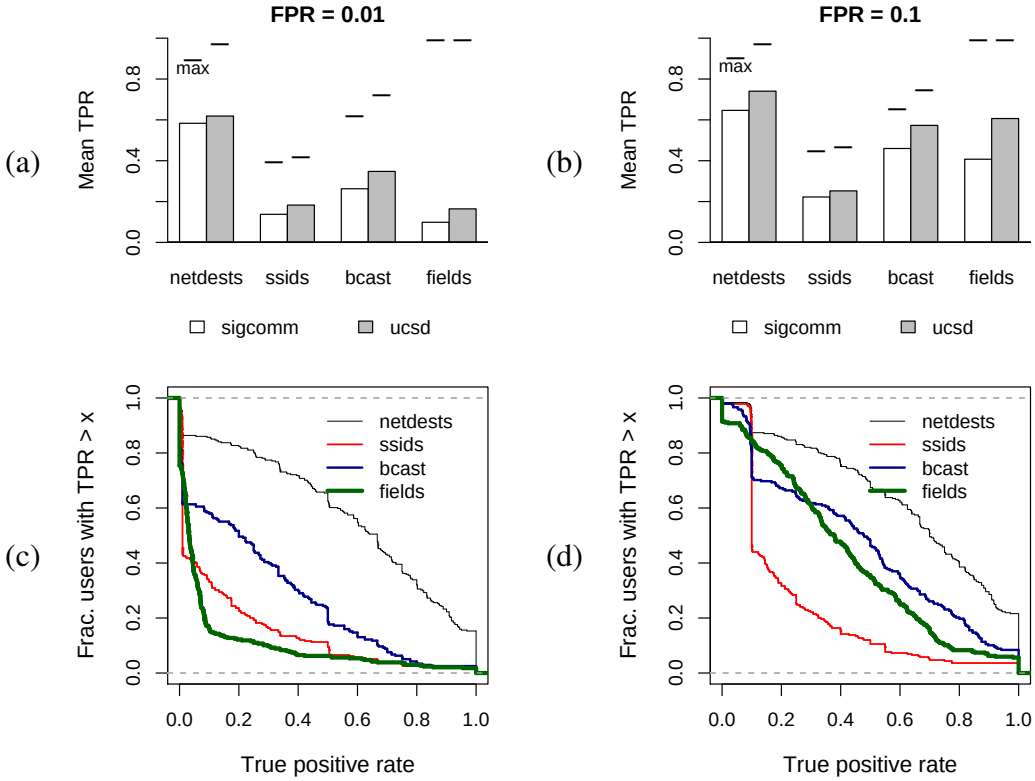


Figure 2.3: Classification accuracy using each feature. The top two graphs show the mean achieved TPR for (a) FPR = 0.01 and (b) FPR = 0.1. The line above each bar show the maximum expected TPR given a perfect classifier on that feature. The bottom two graphs show a CCDF of the achieved TPR on `sigcomm` users for (c) FPR = 0.01 and (d) FPR = 0.1.

### Results

In order to examine the characteristics of each individual implicit identifier, we now focus on the TPR achieved for different FPR targets using each identifier in isolation.

**Mean Accuracy.** Figure 2.3(a) and (b) shows the mean TPR achievable with each implicit identifier in isolation for FPR = 0.01 and FPR = 0.1, respectively. For example, when using netdests, we can identify samples from the average user in both workloads about 60% of the time for FPR = 0.01. The line above each bar indicates the maximum expected TPR that a perfect classifier would achieve on that implicit identifier—i.e., a classifier that

always classifies a sample correctly if it has that implicit identifier, but guesses randomly otherwise. This line is below 1.0 because some validation samples do not contain a particular implicit identifier and, hence, even a perfect classifier on this identifier would not do better than random guessing on those samples. For example, many samples have no SSID probes and, thus, are missing the ssids identifier.

Figure 2.3(a) shows that the average user sometimes emits an implicit identifier that is highly distinguishing. netdests, ssids, and bcast all achieve moderate TPRs (about 60%, 18%, and 30%, respectively) even for a very low FPR (1%). The lower TPR for ssids is expected, since users usually only emit distinguishing SSIDs when they are searching for a network. Indeed, the theoretical maximum TPR achievable by a perfect classifier is only about 40%. Also, as expected, fields is not able to identify many samples on its own since it only distinguishes wireless cards and drivers.

Figure 2.3(b) shows that the TPR for fields improves to 40% and 60% when FPR = 0.1, for the `sigcomm` and `ucsd` workloads, respectively. Thus, the fields identifier is good at classifying users into groups, and can aid in identifying users in those cases when no unique identifier is observed. This is expected, since fields only distinguishes wireless cards and divers. The TPR of the other three features improves much less dramatically when we increase the allowable FPR from 0.01 to 0.1. This is because most of the other implicit identifiers either uniquely identify a user, or are not identifying at all. Thus, the TPR gains observed when we increase FPR are mostly due to less conservative random guessing on the remaining samples.

This effect can be seen in Figure 2.4, which shows the variation in mean TPR and FPR across classification thresholds for `sigcomm` users. The $x = y$ line shows how well random guessing is expected to perform. The TPR of all the features except for fields grows roughly linearly toward 1.0 after the initial spike, which is the effect that progressively less conservative random guessing would have.

For all features, users in the `ucsd` workload are slightly more identifiable than those in the `sigcomm` trace. This is probably because there are more total users in the `sigcomm` workload and, thus, a higher likelihood that two users exhibit the same traits. We examine the effect population size has on tracking in Section 2.6.2.

**Variation Across Users.** Accuracy for some users is better than others. Thus, Figure 2.3(c) and (d) shows a CCDF of achieved TPR over all users in the `sigcomm` workload, for FPR = 0.01 and FPR = 0.1, respectively. For example, consider netdests when FPR = 0.01. In this case, 65% of users achieve a TPR of at least 50%.

Each of the first three implicit identifiers distinguishes some users very often. Figure 2.3(c) shows that 65%, 11%, 24% of users have samples that are identified at least half

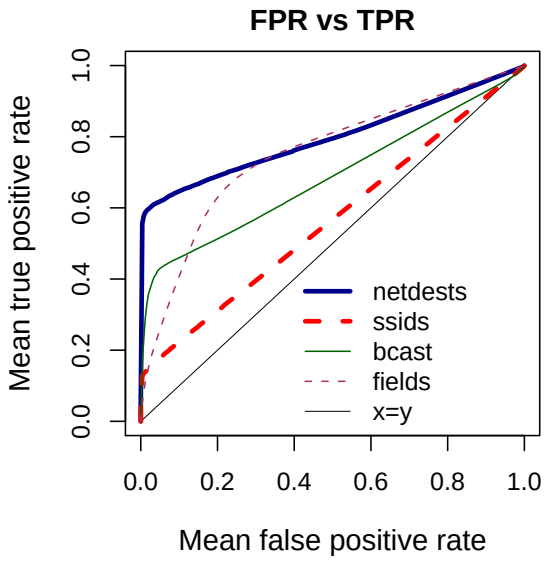


Figure 2.4: The mean achieved TPR and FPR for `sigcomm` users as we vary the classification threshold $T$ using each feature alone. The $x = y$ line shows how well random guessing would perform.

of the time with an FPR of only 0.01 using netdests, ssids, and bcast, respectively. This implies that a non-trivial number of users are trackable even if only one of these features is available.

Nonetheless, when FPR = 0.1, 12%, 53%, and 29% of users have a TPR of at most 0.1 as well using netdests, bcast, and ssids, respectively (see Figure 2.3(d)). This means that our classifier does not perform any better than random guessing on these users. These users are simply not identifiable. For example, for the netdests feature, this means that these users only visited popular destinations during the training period or did not revisit any site in the subsequent days. This result also implies that the mean TPR shown in Figure 2.3(a) and (b) actually underestimates the TPR for the users that are identifiable at all, since this fraction of non-identifiable users drags the mean down. We conclude that there is a large variation in user distinctiveness.

**Training Sample Sensitivity.** To explore the variability in classifier accuracy for different users, we examine whether users observed more often during the training period are more identifiable. Figure 2.5 shows the mean TPR achieved for FPR = 0.01 for sets of `sigcomm` users with different numbers of training samples. The error bars show 95% confidence

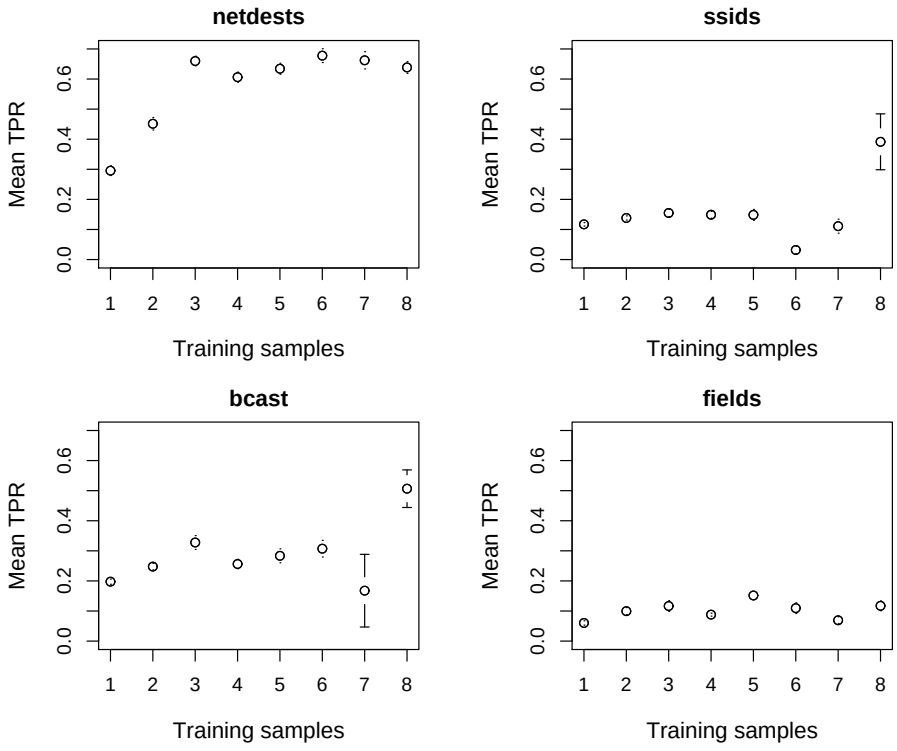


Figure 2.5: Sensitivity to the number of training samples for each feature. The mean TPR achieved for FPR = 0.01 for `sigcomm` users with different numbers of training samples. The error bars indicate 95% confidence intervals.

intervals, which are negligible for most points.

Figure 2.5 shows that the mean TPR noticeably increases with more training samples for netdests and bcast. For netdests, TPR stabilizes after 3 training samples. The TPR of ssids and fields does not change dramatically with more training samples, probably because these identifiers are generated without user interaction and, thus, are nearly always identical when emitted. Artifacts near the right hand side of each graph, such as large confidence intervals, are mostly due to small sample sizes for those points. We conclude that an adversary can build a more accurate classifier with more samples, but needs very few to build one that is useful.

**Accuracy Over Time.** One concern is that the accuracy of ssids may degrade over time since a user's preferred networks list can change. Figure 2.6 shows how the mean TPR varies over two weeks in the `apt` trace, the only trace of that duration, fixing FPR

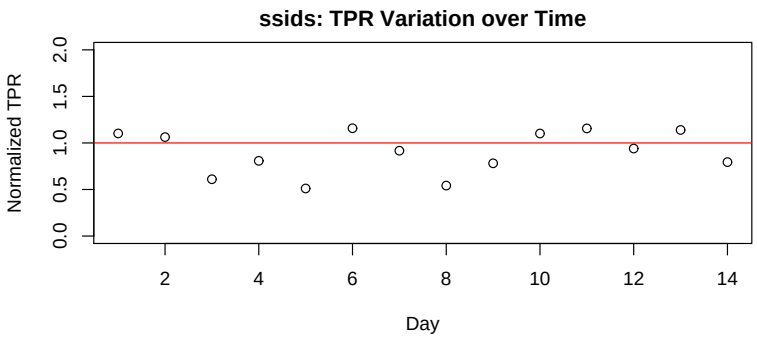


Figure 2.6: Accuracy over time. Normalized mean TPR on each day in the `apt` trace for FPR = 0.01. Each TPR value is normalized to the mean TPR for the entire period, evaluated over the users present during that day. The mean TPR for the entire period over all profiled users is 42%.

= 0.01. Each value is normalized by the mean TPR for the entire period. Even after two weeks, normalized values are close to 1, which suggests that the SSIDs that users emit are relatively stable over time.

## 2.6 When can we be tracked?

In this section, we evaluate how accurately an adversary can answer Question 1 and Question 2 in each of the wireless environments described in Section 2.3. The previous section evaluated how well an adversary could use implicit identifiers independently to determine whether a sample came from a given user, but in practice, an adversary would not be restricted to using identifiers in isolation.

Without link-layer encryption, public networks reveal features both at the link and network layers. In contrast, home networks that employ encryption reveal only link-layer features. Encrypted enterprise networks comprised of homogeneous devices might reveal only link-layer features that vary due to application and user behavior; features that vary due to driver- and card-level differences provide no useful information since they would not vary. Therefore, we evaluate each environment with the following features visible to an adversary:

- Public network: netdests, ssids, fields, bcast.

| | % users with FPR error ¡ 0.01 | |
|---|---|---|
| | median error | 90th percentile error |
| Public | 97% | 82% |
| Home | 80% | 64% |
| Enterprise | 79% | 68% |

Table 2.4: Stability of classifier threshold $T$ across different validation sub-samples. The percentage of users that have FPR errors that are less than 0.01 away from the target FPR of 0.01.

- Home network: ssids, fields, bcast.
- Enterprise network: ssids, bcast.

Since measurements from these environments can be difficult to obtain due to legal and ethical restrictions, we use our analysis of the `sigcomm` trace to estimate answers to these questions. In all three scenarios, we consider users with devices that will have a different pseudonym each hour of the day as in our analysis in the previous section.

Many users in both the `sigcomm` and `ucsd` traces expose implicit identifiers of all four types, so we conjecture that populations in other environments are unlikely to differ substantially beyond the identifiers available. The population sizes will differ, however, so we vary the population size in our experiments. Enterprise networks may be more homogeneous, but the identifiers we consider vary due to user behavior and the applications that they run. ssids will remain distinguishing as long as users visit other networks with their devices, and bcast will remain distinguishing as long as laptops run Windows and use or search for different names, since a large number of broadcast packets are due to NetBIOS.

### 2.6.1 Q1: Did this Sample come from User U?

First, we evaluate how well an adversary can answer Question 1 using features in combination. Since all profiled users had at least one training sample with each of the four features in our training sets, we can evaluate the accuracy on *all* profiled users, not just a fraction, as was the case when using individual features (see Table 2.3).

Figure 2.7 shows how accurately we can answer Question 1 for the average user when varying the threshold $T$ in each of our three environments. Figure 2.8 shows the CCDF of TPR achieved for users in public, home, and enterprise networks for several FPR = 0.01.

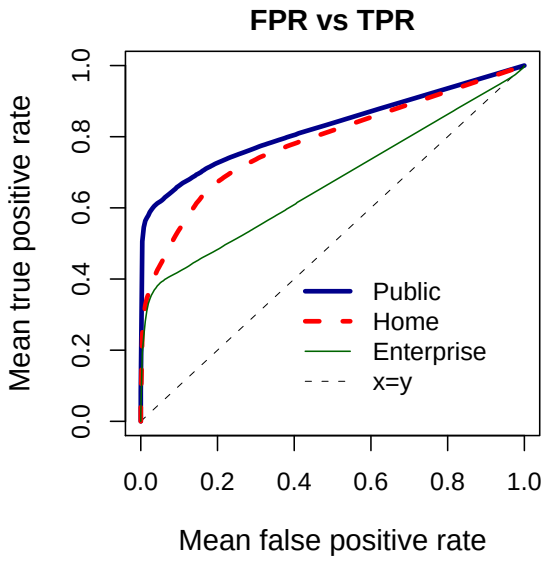


Figure 2.7: Classification accuracy for Question 1 if `sigcomm` users where in typical public, home, and enterprise networks.

When more features are visible, classification accuracy is better. In public networks, user samples are identified 56% of the time with a very low FPR (1%), on average. This TPR is slightly lower than that observed for netdests in Figure 2.3(a) because here we are considering all users, not only the 89% that exhibited netdests in their training samples. The average TPR in home and enterprise networks is 31% and 26%, respectively, when FPR = 0.01. Figure 2.8 shows that when FPR = 0.01, 63%, 31%, and 27% of users are identifiable at least 50% of the time in public, home, and enterprise networks, respectively. As expected, users are more identifiable in environments with more features.

**Selecting the Classifier Threshold.** As mentioned in Section 2.5.2, an adversary would have to select a classifier threshold $T$ to achieve a desired target FPR. In practice, he would have to select the threshold without knowing a priori the resulting FPR of the validation data. Instead, an adversary would have to choose a $T$ that achieves a target FPR in *previous* samples he has collected (e.g., as part of training). Therefore, in order to achieve the desired accuracy, the adversary requires that the $T$ chosen in this manner achieves approximately the FPR target in yet unknown validation data.

To test whether this requirement is met, we ran the following experiment on the `sigcomm` workload: An adversary selects $T$ that achieves FPR = 0.01 on a random 20% subsample

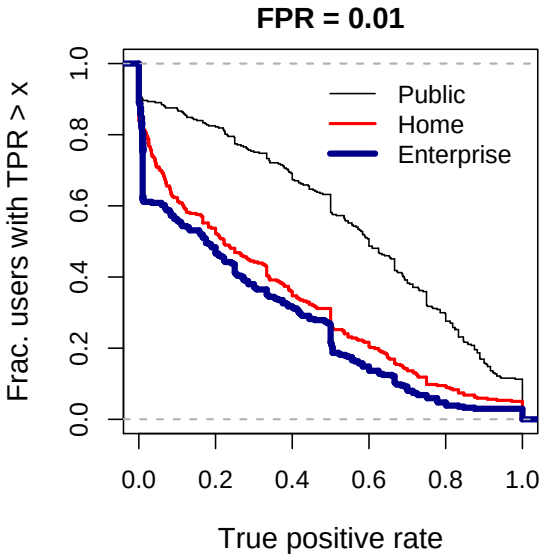


Figure 2.8: CCDF of TPR for Question 1 if `sigcomm` users were in a typical public, home, or enterprise network for FPR = 0.01.

of the validation data and tests whether the same $T$ achieves a similar FPR in a different random 20% subsample. We perform 10 trials of this experiment per user and measure the absolute FPR errors, i.e., the difference between the achieved FPR and the target FPR. Table 2.4 shows the number of users that have median and 90th percentile errors that are less than 0.01 away from the target FPR. 79-97% of users in all scenarios have errors less than 0.01 away from the target most of the time. This suggests that an adversary would be able to select $T$ that achieves an FPR very close to a desired target in most circumstances.

### 2.6.2 Q2: Was User U here today?

Now we consider Question 2. We consider an adversary that wants to accurately detect the presence of a user during a particular 8 hour work day. In this section, we answer the following two questions: (1) How many users can be detected with high confidence? (2) How often does a user have to be active in order to be detected?

## Methodology

**Accuracy Estimation.** Consider an environment with $N$ users present each hour during an eight hour day. User $U$ operates a laptop during $active$ different hours this day and thus an adversary obtains $active$ samples from $U$. The adversary also obtains up to $N$ samples each hour from the other users.

Suppose an adversary would like to determine whether $U$ is present during this day with a TPR of at least $TPR_{target}$ and an FPR of no more than $FPR_{target}$. In section 2.5.2, it was shown that an adversary could use features in combination to answer whether a particular traffic sample came from $U$ with a moderate TPR ($tpr_{Q1}$) and a very low FPR ($fpr_{Q1}$), on average. Unfortunately, even a very low $fpr_{Q1}$ could result in the misclassification of a sample because during an eight hour day, there would be up to $8N$ opportunities to do so. Therefore, to boost the adversary's accuracy, he could answer Question 2 affirmatively only when multiple samples are classified as being from $U$.

Specifically, suppose the adversary only answers Question 2 affirmatively when at least one sample from $belief$ different hours is classified as from $U$. That is, he believes $U$ is present during at least $belief$ different hours. If we assume that the observations made during each hour are independent, when $U$ is active during at least $active \geq belief$ hours,

$$TPR_{target} \geq \Pr[X \geq belief],$$

where $X$ is a binomial random variable with parameters $n = active$ and $p = tpr_{Q1}$. In addition,

$$FPR_{target} \leq \Pr[Y \geq belief],$$

where $Y$ is a binomial random variable with parameters $n = 8$ and $p \leq 1 - (1 - fpr_{Q1})^N$, the probability that at least 1 sample not from $U$ during one hour is misclassified. We show below that the independence assumption is not unreasonable.

In order for an adversary to answer Question 2 with $TPR_{target}$ and $FPR_{target}$, he would determine if there exists a threshold $T$ for $U$'s classifier that would satisfy these constraints. In the process, he would also determine the minimum number of hours that $U$ would have to be active ($active$). For example, when all four features are available, we show that quite a few users can be detected when they are active for several hours even if an adversary desires 99% accuracy (i.e., $TPR_{target} \geq 99\%$ and $FPR_{target} \leq 1\%$).

**Dependence.** The constraints above assume that the observations made during each hour are independent. That is, the likelihood of observing a true or false positive is not depen-

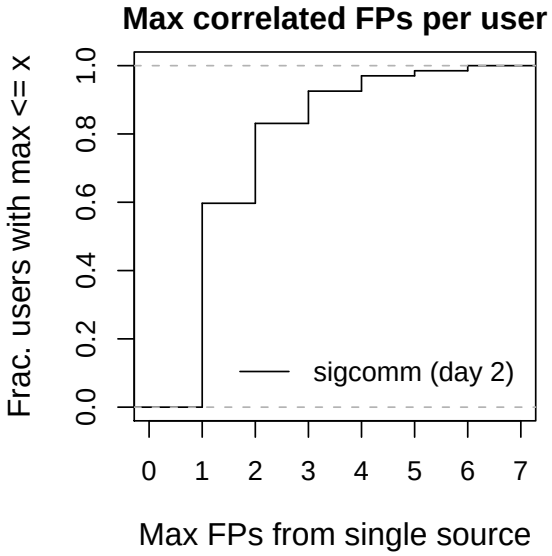


Figure 2.9: Limited dependence in the `sigcomm` trace. CDF of the maximum number of false positives (FPs) generated by any one user for each user.

dent on the adversary's past observations. The following analysis of the `sigcomm` trace shows that there is some dependence in reality, but that the dependence is small.

There are two primary concerns. The first concern is that our classifier may often confuse user $U$ with another user $Q$, so that if $Q$ is active, then the false positive rate will be high regardless of the number of hours that the adversary samples. This concern is mitigated by two factors that add randomness to the sampling process: 1) users enter and depart from the environment and 2) user behavior is variable to begin with. Consider our classifier on all features using a classification threshold $T = 0.5$. Figure 2.9 shows, for each user that exhibits any false positives during the second full day of the `sigcomm` trace, the maximum number of false positives that are contributed by any other single user. From this cumulative distribution function (CDF), we see that for 60% of users, no single other user is responsible for more than 1 false positive, and for over 95%, no single user is responsible for more than 3 false positives. Therefore, most of the time the two factors mentioned prevent a large number of false positives from being correlated to a single user. In addition, since the user set is relatively static at a conference, there is likely to be more churn in the population of most other environments, further reducing the dependence.

The second concern is that there may be temporal locality in either true or false positive

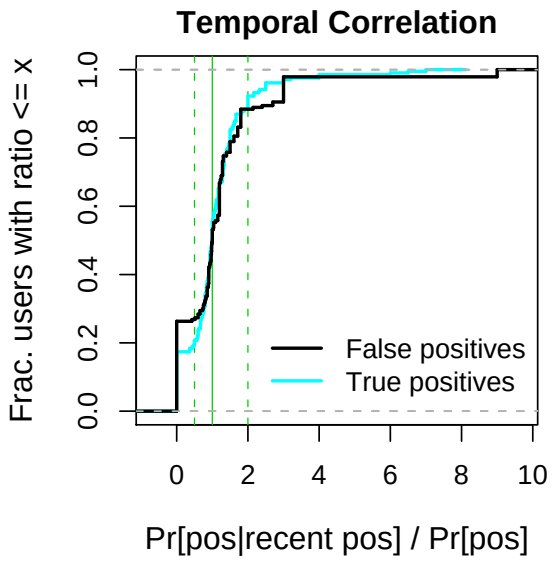


Figure 2.10: Limited dependence in the `sigcomm` trace. CDF of how much more likely a true or false positive is given that one was observed recently.

samples. For example, we might expect that a user is much more likely to exhibit a particular feature if he has done so in the recent past. If temporal correlation was substantial then the ratio

$$\frac{\Pr[\text{positive} \mid \text{positive in the last } t \text{ hours}]}{\Pr[\text{positive}]}$$

would be much larger or smaller than 1. Figure 2.10 shows a CDF of this ratio for each users' true and false positives when $t = 2$ using the same classifier as above. For true positives, we only consider times during which the user is active. For false positives, we only consider the active 9 hours of the last 2 days of the conference since false positives are obviously less likely to occur when fewer people are present. If there was no temporal correlation, we would obtain a vertical line at $x = 1$. We note that 60 and 70% of users' true and false positives are within a factor of 2 of this line, meaning that if a true (false) positive was seen in the last two hours we are no more than 2 times more or less likely to observe another true positive than otherwise. Moreover, given the small number of positives for each user, much of this variation is probably due to randomness. Therefore, temporal dependence is small.

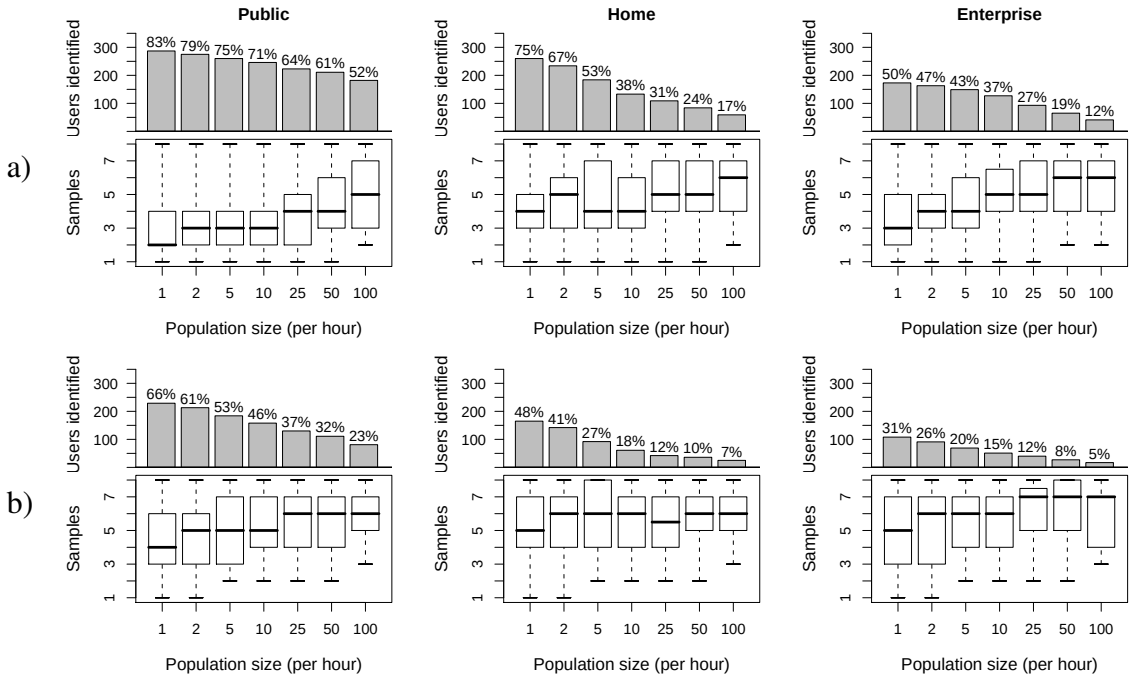


Figure 2.11: The number of of users detectable and the number of hours they must be active to be detected with (a) 90% accuracy and (b) 99% accuracy. The x-axis in each graph varies the population size. The top portion shows the number and percentage of users it is possible to detect. The bottom portion shows a box plot of the number of hours during which they must be active to be detected. That is, the thick line through the middle of each box indicates the median, the ends of each box demark the middle 50% of the distribution, and the whiskers indicate the minimum and maximum values.

## Results

Figure 2.11 shows the number of users detectable and the number of hours they must be active to be detected with (a) 90% accuracy, (b) 99% accuracy. The x-axis in each graph varies the number of users present each hour. The top half of each graph shows the number of users an adversary can detect and, above each bar, the percentage of profiled users the number represents. The bottom half of each graph shows a box plot of the number of hours during which these users must be active to be detected. That is, the thick line within each box shows the median number of hours a detectable user has to be active to be detected, while the ends of each box demark the first and third quartiles. The whiskers mark the minimum and maximum.

For example, part (a) shows the results if the adversary desires an accuracy of 90% (i.e., $TPR_{target} \geq 90\%$ and $FPR_{target} \leq 10\%$). Consider the public networks figure. The fourth bar from the left in top part shows that when there are 10 users present per hour, we can detect 71% of users if they are active during all 8 hours when present. The box and whiskers just below that in the bottom part shows that shows that most of these users do not need to be active all 8 hours to be detected. Of the 71% of users that can be detected, 75% of them only need to be active for 4 hours to be detected, 50% for at most 3 hours, and 25% for at most 2 hours.

**Conclusions.** We make two overall conclusions. First, an adversary can successfully combine multiple implicit identifiers from a few samples to detect many users in common networks with high accuracy. The majority of users can be detected with 90% accuracy when active often enough in public networks with 100 concurrent users or less. At least 27% of users are detectable with 90% accuracy in all of the networks when there are 25 concurrent users or less. This implies that many users can be detected with high confidence in small to medium sized networks regardless of type if they are active often enough. Even in large networks with 100 users, 12% to 52% remain detectable.

Second, some users are detectable with very high accuracy. Even if an adversary desires 99% accuracy, the fraction of detectable users is between 12% and 37% in all networks with 25 users when they are active often enough. Therefore, even applying existing best network security practices will fail to protect the anonymity of a non-trivial fraction of users.

Indeed, several usage patterns in home and enterprise networks make detection more likely than the overall results suggest. In home networks, very few users are likely to be active during each hour. For example, even when monitoring all the networks in our `apt` trace, we only observed 4 users per hour, on average. Therefore, the results closer to the left side of each graph are more representative of home environments. Since users of a enterprise network are probably employees, they are more likely to be active for the entire observation period. Thus, the top half of each graph is probably a good estimation of the fraction of users that an adversary can detect on a typical day.

## 2.7 Summary, Implications, and Limitations

This chapter demonstrated that users can be tracked using *implicit identifiers*, traffic characteristics that remain even when unique addresses and names are removed. Although we found that our technique's ability to identify users is not uniform—some users do not

display any characteristics that distinguish themselves from others—most users can be accurately tracked. For example, the majority of users can be tracked with 90% accuracy when active often enough in public networks with 100 concurrent users or less. Some users can be tracked with even higher accuracy. Therefore, pseudonyms are insufficient to provide location privacy for many users in 802.11 networks.

Moreover, our results showed that even a single implicit identifier, such as netdests, ssids, or bcast, can be highly discriminating and that an adversary needs only 1 to 3 samples of users' traffic to track them successfully, on average. We note that by considering a subset of all possible implicit identifiers and a weak, passive adversary, our results only place a lower bound on the accuracy with which users can be tracked.

### 2.7.1 Implications

We believe our study illustrates three inherent shortcomings of the 802.11 protocol beyond exposing explicit identifiers, none of which is trivially fixed. These shortcomings afflict not only 802.11 but many wireless protocols, including Bluetooth and ZigBee.

**Identifying information exposed at higher layers of the network stack is not adequately masked.** For example, even with encryption, packet sizes can be identifying. Padding, decoy transmissions, and delays may hide information exposed by size and timing channels, but increase overhead. For example, Sun *et al.* [150] found that 8 to 16 KB of padding is required to hide the identity of web objects. The performance penalty due to this overhead would be especially acute in wireless networks due to shared nature of the medium.

**Identifying information during service discovery is not masked.** 802.11 service discovery can not be encrypted since no shared keys exist prior to association. This raises the more general problem of how two devices can discover each other in a private manner, which is expensive to solve [1]. This problem arises not only when searching for access points, but also when clients want to locate devices in ad hoc mode, such as when using a Microsoft Zune to share music or a Nintendo DS to play games with friends.

**Identifying information exposed by variations in implementation and configuration is not masked.** Each 802.11 implementation typically supports different 802.11 features (e.g., supported rates) and has different timing characteristics. This problem is difficult to solve due to the inherent ambiguity of human specifications and manufacturers' and network implementers' desire for flexibility to meet differing constraints.

Balancing the costs involved in rectifying these shortcomings with the incentives nec-

essary for deployment is itself a challenge. Nonetheless, rectifying these flaws at the protocol level is important so that users need not limit their activities in order to protect their location privacy. By measuring the magnitude with which each flaw contributes to the implicit identifier problem, our study provides insight into the proper trade-offs to make when correcting these design flaws in future wireless protocols. We describe how future protocols should address these problems in the next chapter. In the short term, our study may give guidance to individuals that are willing to pro-actively hide their identity in existing wireless networks.

### 2.7.2 Study Limitations

In our measurement analysis, we relied on wireless traces with durations of at most several days. In addition, we did not examine wireless traffic from some common environments such as hotspots. We do not believe our analysis would be substantially altered in these environments because most of the implicit identifiers we examined are generated "automatically" by applications running on a device rather than due to particular user behavior. In addition, our longer-term study of SSIDs suggests that at least one implicit identifier is stable over time. This matches our intuition that re-configuration of user devices and applications, which can change the implicit identifiers a device exhibits, occurs rarely. Nonetheless, it would still be useful to complement our study with a longer-term study of wireless traffic from users in more diverse environments to validate these conjecture and to understand how implicit identifiers may evolve over time. Such a study would aid in understanding whether pseudonym schemes may be sufficient to mitigate tracking threats at longer time scales and different types of locations.

# Chapter 3

# Mitigating Eavesdropping Threats

Identifiers and addresses have always played important roles in network protocols. For example, the structure of IP addresses is critical to scalable IP routing. In addition, application layer discovery protocols such as DNS expose identifiers in order to support rendezvous between clients and services. Identifiers in IP, transport, and application layer protocols pose privacy threats to users when eavesdroppers can intercept messages being routed in the middle of networks, e.g., by ISPs or on LANs where link layer traffic is unencrypted. For example, these eavesdroppers can track when a user is online and determine which parties are communicating. A number of defenses have been developed to guard against these traffic analysis attacks, such as mix networks [49] and cover traffic [68, 121, 161], but they are heavy-weight and incur substantial performance penalties. Fortunately, wired networks can be protected with physical security and link layer encryption, and only powerful adversaries that collude with ISPs can carry out eavesdropping attacks in the middle of the network.

Wireless link-layer protocols, however, are much more vulnerable to weak adversaries because anyone can eavesdrop on communications from devices nearby using only commodity hardware. Although link-layer encryption such as WPA can obscure identifiers in IP, transport, and application layer protocols, link-layer protocols such as 802.11 and Bluetooth expose identifiers themselves. Two types of identifiers play important roles in these protocols also:

- *device/service identifiers*: Device and service identifiers typically persist over long time scales (months, years, or longer) and are transmitted when devices try to set up connections. The process of service discovery and rendezvous, such as with an available access point (AP), requires a service to announce its existence with an

explicit, recognizable identifier or for a client to probe for it (e.g., a network's SSID and/or BSSID in 802.11). Either way, the process relies on the transmission of a service name explicitly.

- *connection identifiers*: Connection identifiers or addresses are used to identify messages sent by the two devices participating in a connection. Thus, they must persist for at least the duration of a connection. A destination address (802.11) or connection identifier (WiMAX) allows a device to decide whether it is the destination of a message by using a simple compare operation. Mechanisms such as ARP that translate between addresses at different layers rely on identifiers that persist for the duration of a connection.

In the previous chapter, we showed that users can often be tracked even when MAC addresses periodically changed. This is because eavesdroppers can still observe temporary addresses and network names in transmissions in addition to other traffic properties that serve as implicit identifiers. Concealing specific fields, such as MAC addresses, leaves open the possibility of tracking and inventorying by other fields that have not been protected. Furthermore, because sequences of encrypted packets within a session remain linked by a connection identifier, side-channels can reveal sensitive information about their contents. For example, a distinct pattern of packet sizes and timings is sometimes sufficient to identify the keys a user types [147], the web pages he views [150], the videos he watches [142], the languages he speaks [171], and the applications he runs [172]. In this chapter, we present the first design and prototype of a wireless protocol that conceals *all* explicit information that can be used as identifiers, including device identifiers, connection identifiers, and all other explicit message fields.

The obvious difficulty with simply removing identifiers is that they play key roles in the efficient operation of existing protocols. For example, a connection identifier allows a device to decide whether it is the destination of a message by using a simple compare operation. Mechanisms such as ARP that translate between addresses at different layers rely on identifiers that persist for significant periods of time. And the process of service discovery and rendezvous, such as with an available access point (AP), requires a service to announce its existence with an explicit, recognizable identifier or for a client to probe for it. Either way, the process relies on the transmission of a service name explicitly.

This chapter presents SlyFi, an 802.11-like protocol that encrypts entire packets to remove explicit identifiers while retaining efficiency comparable to 802.11 with WPA. No explicit information in SlyFi messages can be used by third parties to link them together. We show that all features that rely on identifiers—service discovery, packet filtering, and address binding—can be supported without exposing them. Different mechanisms are

used for service discovery and subsequent data transfers, but in both cases a device can determine whether it is the recipient of a message with lightweight table look-ups. We have implemented SlyFi on commodity 802.11 NICs and our experiments show that SlyFi's performance impact is modest. In particular, we show that a SlyFi client can discover and associate with services even faster than 802.11 with WPA using PSK authentication. SlyFi's overhead results in a throughput degradation that is only slightly greater than that of WPA with CCMP encryption (10% vs. 3%).

**Chapter outline.** Section 3.1 discusses the limitations of previous pseudonym proposals. Section 3.2 presents the requirements of a solution and an overview of SlyFi. Section 3.3 presents the design of two main mechanisms it uses, while Section 3.4 discusses practical details and our prototype implementation. Section 3.5 demonstrates SlyFi's security properties formally and Section 3.6 analyzes SlyFi's robustness to different types of attacks. Section 3.7 presents performance evaluation results. Section 3.8 concludes this chapter.

## 3.1 Related Work

A number of proposals have argued for obscuring identifiers in network protocols at different layers of the protocol stack. These proposals advocate for three types of pseudonyms: unilateral pseudonyms, negotiated pseudonyms, cryptographically generated pseudonyms.

**Unilateral pseudonyms.** Some wireless protocol identifiers can be changed over time without altering the functionality of a wireless protocol. For example, in 802.11, MAC addresses serve as connection identifiers (source and destination addresses), but persist for the lifetime of a device's network interface hardware. These identifiers can be changed by a device without negotiating with any other party [67, 84, 170]. In other words, they can be implemented by a client without AP support. We analyzed the limitations of using unilateral MAC address pseudonyms for 802.11 in Chapter 2. This solution is insufficient because service identifiers and connection identifiers (e.g., SSIDs) remain exposed, revealing implicit identifiers. Moreover, each client will have to establish a new connection with an AP each time they change their MAC address, disrupting end-to-end connections at higher layers.

**Negotiated pseudonyms.** One way to change connection identifiers more frequently is to have both ends of a connection (e.g., the client and the AP) periodically negotiate a new pseudonym. For example, GSM's connection identifier for a client, the Temporary Mobile Subscribed Identifier (TMSI), can be refreshed periodically by the base station. Negotiated pseudonyms can also be used to make device identifiers, such as a GSM mobile

phone's International Mobile Subscriber Identity (IMSI), more ephemeral. In GSM, a mobile phone typically must transmit its globally unique IMSI in the clear to identify itself to a base station before it can establish a connection. Once a mobile has been authenticated, however, it can negotiate a new IMSI pseudonym to identify itself the next time it tries to establish a connection with the GSM network [73].

Explicitly negotiated pseudonyms would still pose two problems for common wireless protocols such as 802.11 and Bluetooth. First, because both ends of a connection must reliably agree on the new pseudonym before it can be used, at least one pair of messages must be exchanged. This would add substantial overhead if very frequent pseudonym changes are desired (e.g., on a per-message basis). Second, when used in discovery and rendezvous, at least one party to the rendezvous may transmit the same pseudonym multiple times (e.g., [48]). This is because discovery must involve probes for a service or announcements from the service, and because neither the client or the service can negotiate a new pseudonym until they rendezvous again, any probes or announcements that do not elicit a response must be repeated with the same pseudonym. This enables eavesdroppers to track at least one party. This is not a significant concern in GSM because base stations do not care about location privacy; thus, they can authenticate themselves to a mobile phone before the phone reveals its own identity. However, in 802.11 and Bluetooth, both the client and service may be personal mobile devices (e.g., a mobile phone communicating with a laptop), so both parties require location privacy.

One way to mitigate these problems is to negotiate multiple pseudonyms for each device pair and use one per message. A number of proposed RFID protocols use this solution. RFID protocols involve wireless readers and tags, where the role of tags is to identify themselves (e.g., with a serial number) to readers that query them. No other information except a tag's device identifier is transmitted, so RFID protocols are comparable to basic discovery protocols in 802.11 and Bluetooth. To protect a tag's identifier from unauthorized readers, researchers have proposed storing a pseudonym table on tags and authorized readers [124, 87], so that a tag can emit a different identifier from the table each time it is queried. One problem with this approach is that, once all pseudonyms in the table are exhausted, any new message must repeat a pseudonym. Thus, due to the overhead of storing a table per device pair, this approach is impractical when the number of messages sent between renegotiations is large (e.g., for high volume data transfers) or can be unbounded (e.g., for generic discovery messages).

**Cryptographic pseudonyms.** A number of cryptographic schemes have been proposed to generate pseudonyms without explicit negotiation for each pseudonym change. These schemes typically use pseudorandom sequences or public key cryptography.

Arkko *et al.* [10] and Lindqvist *et al.* [108] argue for replacing explicit identifiers at all layers of the network stack to improve privacy. Arkko *et al.* propose replacing identifiers in messages with cryptographically secure pseudorandom sequences, where each message contains a new element from the sequence. For example, for a sender and receiver that share a symmetric key, Arkko *et al.* show how identifiers in TCP can be replaced with pseudorandom sequences. Pseudorandom sequences, such as pseudorandom functions (PRFs) constructed using cryptographic hash functions and pseudorandom permutations (PRPs) constructed using symmetric block ciphers, play an important role in most cryptographic pseudonym schemes that use symmetric keys, including SlyFi. There are two main research questions in how to incorporate pseudorandom sequences into network protocols: how do senders and receivers synchronize these sequences, and, since a receiver must store one symmetric key per potential sender, how can they scalably maintain and lookup these keys?

Pseudorandom sequence schemes have been proposed for use in a number of specific protocols. Some proposed RFID protocols [87, 163, 89] use pseudorandom sequences to compute the table of pseudonyms, mentioned above, on the fly. However, these protocols disagree on how synchronization of sequences on readers and tags should be done. Cox *et al.* [46] uses pseudorandom sequences for an application-level discovery protocol. Singelée and Preneel [144] propose using pseudorandom sequences to replace connection identifiers in Bluetooth. Armknecht *et al.* [11] propose using them in encrypted 802.11 headers. Linqvist *et al.* [107] propose a pseudonym-based discovery protocol that avoids the synchronization problem by having the client include a *nonce* value in its probe that is used to derive the cryptographic pseudonym in a service's response.

While each of these proposals address particular components of wireless protocols, none of them are complete, leaving out important functions such as either service discovery, authentication, data transport, higher-layer binding, etc. Some also have performance deficiencies. For example, the scheme proposed by Cox *et al.* uses a hash-chain that evolves with real time to compute a pseudorandom sequence. Thus, a device that is asleep for a while would have to compute every intermediate address before obtaining the current address to use. This application-layer protocol tolerates this extra expense because its addresses change very infrequently. A fundamental problem with Linqvist *et al.*'s approach is that it requires one party to try every key it has to decode a message. Therefore it is inefficient when a receiver has many keys and receives packets not destined for it, e.g., when it sees competing background traffic. Similarly, the scheme proposed by Armknecht *et al.* requires a receiver to try every key it has to decode packets with no matching address. Therefore, this scheme can be inefficient in real environments.

Cryptographic schemes based on public key cryptography have also been proposed

| | 10 ms | 100 ms | 1 sec | 1 min | 1 hr |
|---|---|---|---|---|---|
| SIGCOMM 2004 | 1.4 | 3.2 | 7.6 | 24.7 | 80.1 |
| OSDI 2006 | 4.6 | 9.0 | 20.6 | 60.8 | 221.3 |
| UCSD 2006 | 2.4 | 7.1 | 17.9 | 76.6 | 176.6 |

Table 3.1: Mean number of devices that send or receive 802.11 data packets at different time intervals at two conferences (SIGCOMM [138], OSDI [40]) and one office building (UCSD [43]). Intervals with no data packets are ignored. UCSD has observations from multiple monitors.

to protect identifiers in discovery and rendezvous messages. For example, Abadi and Fournet [1] present a public key scheme for private authenticated discovery. Practical implementations of these schemes in wireless protocols are inefficient, however, due to the high overhead of public key cryptography. We compare pseudonym schemes based on symmetric and public key cryptography in greater technical depth in Section 3.3.1.

In addition to the limitations enumerated above, very few of these cryptographic pseudonym proposals have been implemented (only [46], to our knowledge). Thus, SlyFi is the first to show that real devices that can implement these proposals in a practical manner. Practical implementations are important to understand economic feasibility and deployability.

## 3.2 Problem and Solution Overview

Our goal is to build a wireless link layer protocol that allows clients and services to communicate without exposing identifiers to third parties. This section outlines the threat model we consider. We then discuss our security requirements and the challenges in meeting them and present an overview of SlyFi, an efficient identifier-concealing link layer protocol based on 802.11.

### 3.2.1 Threat Model

**Attack.** The previous section outlined three types of attacks enabled by low-level identifiers not obscured by existing security mechanisms: the inventorying, tracking, and profiling of users and their devices. Users can be subjected to these attacks without their knowledge because an adversary can carry them out without being visibly or physically

present. In addition, users are vulnerable even when using the best existing security practices, such as WPA. Thus, these attacks violate common assumptions about privacy. The effectiveness of these attacks is dependent on an adversary's ability to link packets sent at different times to the same device. The easiest way for adversaries to link packets is by observing the same low-level identifier in each.

Thus, our goal is to limit two forms of *linkability*: First, information should not be provided in individual packets that explicitly links the packets to the identities of the sender or intended receiver. Second, to prevent the profiling, fingerprinting, and tracking of sequences of related packets, packets from the same sender should not be linkable to each other, irrespective of whether any one of them may be linked explicitly to its source. In other words, when there are $k$ potential devices and an adversary observes a packet, he should only be able to infer that the packet is from (or to) one of those $k$ devices, not which one. Profiling a device's packet sequences would be more difficult even at short timescales if many devices are active simultaneously. Table 3.1, which shows the average number of active devices observed at different time intervals, shows that there are indeed many simultaneously active devices in three 802.11 traces.

**Potential Victims.** The aforementioned attacks are damaging to both wireless clients, such as laptops, and wireless services, such as APs, particularly since the distinction between client and service devices is becoming increasingly blurred; e.g., a client game station sometimes provides wireless service to others as an ad hoc AP. Thus, we want to limit the linkability of packets transmitted by both clients and services.

We assume that clients and services have (possibly shared) cryptographic keys prior to communication. These keys can be obtained in the same way as in existing secure 802.11 and Bluetooth networks. For example, devices can leverage traditional credentials from trusted authorities (e.g., for RADIUS authentication) or bootstrap symmetric keys using out-of-band pairing techniques [152]. We believe that most private services will be known beforehand (e.g., a home 802.11 AP) and can bootstrap keys using these methods. Nonetheless, in previous work [129] we also proposed methods to privately bootstrap keys with unknown services by leveraging transitive trust relationships.

The mere possession of cryptographic keys does not immediately yield satisfactory solutions, however, as clients and services have limited computational resources. As a consequence, solutions should not enable denial of service attacks that exploit this limitation. For example, simply encrypting the entirety of a packet is not sufficient if a receiver can not quickly determine whether it is the intended recipient or not. This is because an adversary would then be able to exhaust a device's computational resources by broadcasting "junk" packets that the device would expend a non-trivial amount of resources to discard.

**Adversary.** We are concerned with limiting the packet linking ability of *eavesdroppers*, i.e., parties other than the original sender or intended recipient of those packets. For example, packets sent between an 802.11 client and an 802.11 AP are exposed to anyone within radio range, but only the client and service should be able to link them together. We are not concerned with preventing the service from linking together the client's packets (or vice versa), as techniques used to hide a client's identity from a service in wired networks (e.g., [50]) are also applicable in wireless networks.

We assume adversaries have commodity 802.11 radios and are able to observe all transmitted packets, but they are not privy to the cryptographic keys that clients and services have prior to communication. As with most practical systems, we assume that adversaries are computationally bounded and thus can not successfully attack standard cryptosystems such as RSA, ElGamal, and AES.

**Limitations.** SlyFi's removal of low-level identifiers makes it much more difficult for third parties to link packets together or to a particular user, thus improving privacy. Nonetheless, packet sizes, packet timings, and physical layer information may still sometimes act as side channels that link packets together. We briefly discuss SlyFi's resilience to these types of attacks in Section 3.6, but a thorough analysis is outside the scope of this dissertation. However, without explicit identifiers linking together packets, it becomes a more difficult probabilistic task to separate the transmissions of different sources. Such attacks are less accessible as they usually require sophisticated attackers [153] or non-commodity hardware [131].

We note that packet sizes and timing can be hidden using well-known packet padding and cover traffic techniques that make all packets appear uniform. These techniques can be applied at the network layer, so SlyFi complements them by encapsulating their output packets in an identifier-free link layer. The primary disadvantage of these existing techniques is that they must add significant overhead to ensure that all side-channels are hidden (e.g., by padding all packets to the maximum packet length). We discuss these trade-offs and the circumstances under which such overhead might be needed in Section 3.6.

### 3.2.2 Security Requirements

We want to be able to deliver a message from $A$ to $B$ without identifiers, but still ensure that $B$ can verify it was sent by $A$. More specifically, consider a procedure $F$ that computes $c \leftarrow F(A, B, p)$, where $A$ and $B$ are the identities of the sender and recipient, respectively, $p$ is the original message payload, and $c$ is the result which $A$ transmits. (Shared cryptographic key state is an additional, implicit input to $F$, but we omit it here for brevity.) We

want $F$ to have the following four properties. We denote security properties in this chapter using small caps.

STRONG UNLINKABILITY. To protect against tracking and profiling attacks, a sequence of packets should not be linkable. More formally, any party other than $A$ or $B$ that receives $c_1 = F(A, B, p_1)$ and $c_2 = F(A, B, p_2)$ should not be able to determine that the sender or receiver of $c_1$ or $c_2$ are the same. In particular, this implies that $c_1$ and $c_2$ must not contain consistent identifiers. We note that some packet types, such as discovery messages, are less vulnerable to short-term profiling and thus only need to be unlinkable at coarser timescales to prevent long-term tracking. Consequently, we outline a relaxed version of this property in Section 3.3.3 to efficiently handle these packets.

AUTHENTICITY. To restrict the discovery of services to authorized clients and prevent spoofing and man-in-the-middle attacks, recipients should be able to verify a message's source. More formally, $B$ should be able to verify that $A$ was the author of $c$ and that it was constructed recently (to prevent replay attacks).

CONFIDENTIALITY. No party other than $A$ or $B$ should be able to determine the contents of $p$. In contrast to existing wireless confidentiality schemes, not even fields and addresses in the header should be decipherable by third parties.

MESSAGE INTEGRITY. Finally, as with existing 802.11 security schemes, receivers should be able to detect if messages were tampered with by third parties. More formally, $B$ should be able to derive $p$ from $c$ and verify that it was not altered after transmission.

We give more formal definitions for these properties in Section 3.5.

### 3.2.3 Challenges

The principal approach to concealing 802.11 client identities has been to use MAC address pseudonyms [67, 84]. Pseudonym proposals do not meet our strong unlinkability requirement because all packets sent under one pseudonym are trivially linkable. Moreover, the use of pseudonyms does not conceal other information in headers, such as capabilities, that can be used to link packets together [127]. Furthermore, the proposals focus on data delivery alone, and do not address important network functions, such as authentication and service discovery.

Prior approaches are limited because meeting all our security requirements while maintaining important wireless functionality is nontrivial. Consistent destination addresses al-

low devices to quickly filter messages intended for others so efficient data transport is difficult without them. Moreover, cryptographic authenticity is difficult to provide without identifiers. Message recipients typically need to know which cryptographic key to use to verify a message, and it is hard to tell the recipient which one without explicitly identifying it. Finally, removing identifiers completely from the process of service discovery is hard because wireless clients and services typically rendezvous by broadcasting an agreed upon identifier. A service might be willing to expose its identifier through announcements to save potential clients from having to expose it in probes. No such straightforward solution exists to conceal both client and service identities.

### 3.2.4 System Overview

In light of the shortcomings of existing solutions, we introduce the SlyFi protocol that meets our security requirements using two identity-concealing mechanisms, *Tryst* and *Shroud*, while providing functionality similar to 802.11. Before describing these mechanisms, we first give an overview of SlyFi in this section.

The SlyFi link layer is designed to replace 802.11 for managed wireless connectivity between clients and APs. The privacy protecting mechanisms of the protocol explicitly protect all bits transmitted by the link layer. A client wishing to join and send data to a SlyFi network sends a progression of messages similar to 802.11 (Figure 3.1). Instead of sending these messages in the clear, they are encapsulated by the two identity-hiding mechanisms we describe in Section 3.3.

A client first transmits probes, encapsulated by Tryst, to discover nearby APs it is authorized to use. A probe is encrypted such that: 1) only the client and the networks named in the probe can learn the probe's source, destination, and contents, and 2) messages encapsulated for a particular SlyFi AP sent at different times cannot be linked by their contents. An AP that receives a probe verifies that it was created by an authorized user and sends an encrypted reply, indicating its presence to that client. If the client wishes to establish a link to the AP, it sends an authentication request, also encapsulated by Tryst, containing session information including keys for subsequent data transmission, which are used to bootstrap Shroud. Obviously, SlyFi APs cannot send clear-text beacons if they wish to protect service identities. However, they may do so if they wish to announce themselves publicly. Such a public announcement could immediately be followed by a confidential authentication request from an interested client, and thus would not compromise client privacy.

After a link has been established by an authentication response, Shroud is used to con-

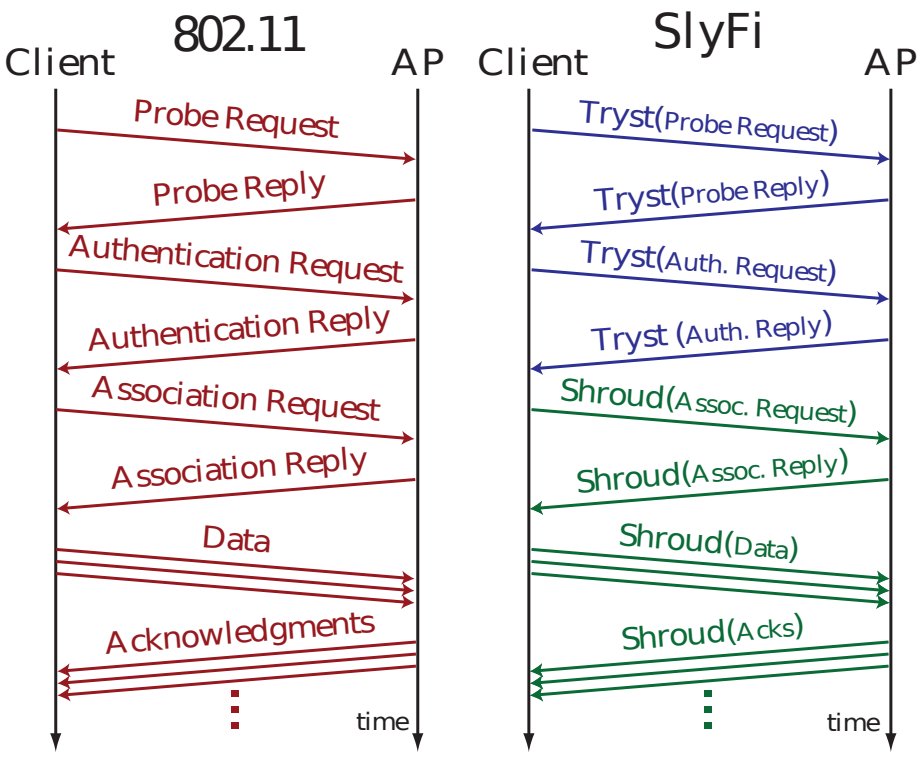


Figure 3.1: The SlyFi protocol.

ceal the addresses and contents of future messages delivered on the link. An eavesdropper can not use the contents of any two messages protected by Shroud to link them to the same sender or receiver.

Both Tryst and Shroud essentially encrypt the entire contents of each message, including addresses normally found in the header. The essential differences between them arise due to the different requirements of discovery, link establishment, and data transfer.

## 3.3 Identifier-Free Mechanisms

Identifiers are used in wireless protocols for two general functions: 1) as a handle by which to discover a service and establish a link to it, and 2) to address packets on a link and allow unintended recipients to ignore packets efficiently. Tryst and Shroud address each of these functions, respectively. To motivate our mechanisms, we first describe two straw man mechanisms that meet our security requirements, but are inefficient. We then discuss Tryst and Shroud, which are enabled by minor relaxations of these requirements or additional assumptions made possible by their intended uses. We conclude the section by discussing how SlyFi can still support other protocol functions, such as higher layer binding.

To illustrate each mechanism we consider the scenario when $A$ sends a message $p$ to $B$. Each mechanism consists of three key elements: the *bootstrapping* of cryptographic keys that the sender and receiver require to compute the procedure $F$ (described in Section 3.2.2); the *construction* of $c \leftarrow F(A, B, p)$ by the sender; and the *message filtering* by a receiver to determine if $c$ is intended for him.

### 3.3.1 Straw Man: Public Key Mechanism

We first sketch public key, a mechanism based on a protocol that Abadi and Fournet [1] prove meet the aforementioned security requirements.

**Bootstrapping.** This mechanism assumes that $A$ and $B$ each have a public/private key pair and each have the public keys of the other.

**Construction.** We sketch this mechanism here, but refer the reader to the first protocol discussed in [1] for details. To provide authenticity, $A$ digitally signs the statement $s = \{A, B, T\}$ where $T$ is the current time. A message header is constructed as an encryption of $s$ and the digital signature, using $B$'s public key. By using a public key encryption scheme that does not reveal which key is used, such as ElGamal [24], identities of neither sender nor intended recipient are revealed.[1] In addition, this achieves strong unlinkability because the ElGamal encryption scheme is randomized so each encrypted header appears random. The payload can be encrypted via conventional means (e.g., as described later in Section 3.3.3).

[1] In practice, we would still use RSA for faster signatures; we just require each party to have both ElGamal and RSA key pairs.

**Message filtering.** When $B$ receives a message, he will attempt to decrypt this header. If the decryption fails (i.e., the result does not include the statement $\{j, B, T\}$, for a known identity $j$), the message is not intended for $B$ and can be discarded. If decryption succeeds, $B$ then checks the signature and the time to verify that the message was recently generated by $j$ before accepting it.

Although this protocol achieves the security properties we desire, it is slow because it uses public key cryptography. In particular, on AP and consumer electronics hardware, a single private key decryption can take over 100 milliseconds—several orders of magnitude greater than the time required to transmit the message (see Section 3.7). Since $B$ must attempt to decrypt the header for every message he receives whether he is the intended recipient or not, he can be backlogged just by processing ambient background traffic. One way to avoid this overhead would be to use a public key protocol only for infrequently occurring messages, such as 802.11 probes. These messages can be used to exchange a symmetric key for future messages. One problem with this approach is that it still leaves receivers susceptible to denial-of-service attacks: anyone can broadcast "junk" probes and force receivers to become back-logged processing them. Moreover, we find that in practice even non-adversarial 802.11 probe traffic can be large enough to back log receivers (see Figure 3.9).

### 3.3.2 Straw Man: Symmetric Key Mechanism

Next we sketch symmetric key, a similar mechanism based on symmetric keys that addresses this pitfall.

**Bootstrapping.** This mechanism assumes that $A$ and $B$ share a symmetric key.

**Construction.** Using symmetric keys shared only between $A$ and $B$, we can use a construction intuitively similar to public key. $A$ encrypts the statement $s$ using symmetric encryption such as AES-CBC. We can omit $A$ and $B$ from $s$ since it is implied by the use of their symmetric key. $A$ then computes a message authentication code (MAC) over the encrypted value so $B$ can verify its authenticity. A random initialization vector (IV) is used so that the resulting cipher text and MAC appear random and thus are unlinkable to any other message.

**Message filtering.** Upon receipt of a message, $B$ verifies the MAC in the header using the same key $A$ used to construct the message. If the MAC does not verify, then this message is not for $B$ and he can discard it.

Of course, since the message does not indicate to $B$ which key was used to generate

| Symbol | Definition |
|---|---|
| $I$ | The length of each Tryst time interval. |
| $T, T_0, T_i$ | Respectively, the current time, the time Tryst keys were bootstrapped, and the start of time interval $i$: $T_0 + i \cdot I$. |
| $k_p$ | A one-time use key for encrypting a payload. |
| $k_{AB}^{Enc}, k_{AB}^{MAC}, k_{AB}^{addr}$ | Long-term keys to encrypt, MAC, and compute addresses for Tryst messages sent from $A$ to $B$. |
| $k_{s:AB}^{Enc}, k_{s:AB}^{MAC}, k_{s:AB}^{addr}$ | Session keys to encrypt, MAC, and compute addresses for Shroud messages sent from $A$ to $B$. |
| $\mathsf{AES}_k(b)$ | Encipher single 128-bit block $b$ with key $k$ using the AES cipher. |
| $\mathsf{AES\text{-}CTR2}_k(i, m)$ | Encrypt $m$ with symmetric key $k$ and 128 bit IV $i$ using the AES cipher in CTR2 mode [139]. |
| $\mathsf{AES\text{-}CMAC}_k(m)$ | Construct 128-bit message authentication code (MAC) of $m$ with key $k$ using AES-CMAC [148]. |
| $\mathsf{SHA1}_{128}(m)$ | Return first 128 bits of a cryptographic hash of $m$. |

Table 3.2: Cryptographic terminology used in Section 3.3.3–Section 3.4. All keys are 128 bits.

the MAC—indeed it cannot, or it will no longer be unlinkable—and $B$ has a symmetric key for each client from whom he can receive messages, $B$ must try all these keys to verify the MAC. There is locality when keys are used (e.g., $A$ may know that he expects a reply from $B$ after sending a message to him) so we can sort keys in most-recently-used order, but, for messages not intended for $B$, he must try all keys before discarding them. Thus, filtering is inefficient for clients or APs that have many keys.

### 3.3.3 Discovery and Binding: Tryst

We now describe Tryst, the mechanism we use for transmitting discovery and binding messages such as 802.11 probes and authentication messages. Tryst builds upon the symmetric key straw man, but leverages the following properties of these messages in order to enable efficient message processing:

| $s =$ | $mac =$ | $etext =$ | $emac =$ |
|---|---|---|---|
| $\{addr^i_{AB}, \mathsf{AES}_{k^{Enc}_{AB}}(k_p)\}$ | $\mathsf{AES\text{-}CMAC}_{k^{MAC}_{AB}}(s)$ | $\mathsf{AES\text{-}CTR2}_{k_{p1}}(0^{128}, p)$ | $\mathsf{AES\text{-}CMAC}_{k_{p2}}(etext)$ |
| 32 bytes | 16 bytes | variable | 16 bytes |

Figure 3.2: Tryst packet format.

*Infrequent Communication.* Individual devices send discovery and binding messages infrequently. For example, 802.11 clients send probes only when they are searching for an AP and send authentication messages only at the beginning of a session or when roaming between APs.

*Narrow Interface.* Unlike data packets, which can contain arbitrary contents, there are very few different messages that are used for discovery and binding. Thus, it is unlikely that their evolution at short time scales exposes many sensitive side channels of information when individual messages are not decipherable. It is only the ability to link these messages together at long time scales (e.g., hours or days) that threatens location privacy.

Based on these two observations, we define a relaxed version of the strong unlinkability property:

LONG-TERM UNLINKABILITY. Let $t(m)$ be the time a message $m$ was generated. Any party other than $A$ or $B$ that receives $c_1 = F(A, B, p_1)$ and $c_2 = F(A, B, p_2)$ should not be able to determine that the sender or receiver of $c_1$ or $c_2$ were the same if $|t(c_2) - t(c_1)| > I$, for some time interval $I$. In practice, $I$ would be several minutes and may be different for each client-service relationship.

Tryst achieves this relaxed form of unlinkability, which is sufficient for discovery and binding messages because very few are likely to be generated during any given interval $I$. Even if an adversary is able to force multiple discovery messages to be generated during one interval, e.g., by jamming the channel to force all clients to reassociate, the ability to link them together is unlikely to be threatening.

For clarity, we list the cryptographic terminology we use in the subsequent description in Table 3.2.

**Bootstrapping.** Similar to symmetric key, $A$ and $B$ each have symmetric keys $k^{Enc}_{AB}, k^{MAC}_{AB}, k^{addr}_{AB}$ for constructing messages from $A$ to $B$ (and another set of keys for $B$ to $A$). They also remember the time they exchanged these keys as $T_0$.

**Temporary unlinkable addresses.** A client $A$ and a service $B$ that share a symmetric key can independently compute the same sequence of unlinkable addresses and thus will at any given time know which address to use to send messages to the other. Specifically:

$$addr^i_{AB} = \mathsf{AES}_{k^{addr}_{AB}}(i), \quad \text{where } i = \lfloor (T - T_0)/I \rfloor$$

In other words, $addr^i_{AB}$ is a function of the $i$th time interval after key negotiation. The crucial property we leverage is that, without knowledge of $k$, it is intractable for an adversary to distinguish a set of address values $\{\mathsf{AES}_k(j_1), \mathsf{AES}_k(j_2), \ldots, \mathsf{AES}_k(j_n)\}$ from random bits.[2] As a consequence, for any two values $\mathsf{AES}_{k_1}(i_1)$ and $\mathsf{AES}_{k_2}(i_2)$ where $i_1 \neq i_2$, it is intractable for a third party to determine whether $k_1 = k_2$, even if $i_1$ and $i_2$ are known. Thus, these addresses are unlinkable without knowledge of $k^{addr}_{AB}$.

In practice, $B$ computes $addr^i_{AB}$ once at time $T_i = T_0 + i \cdot I$. $B$ maintains a hash table containing the addresses for messages he might receive. At time $T_i$, he clears the table and inserts the key-value pair ($addr^i_{jB}$, $j$) for each identity $j$ he has keys for, so that he can anticipate messages sent with these addresses and determine that he should use $j$'s keys to process them. When $A$ wants to send a message to $B$ at time $T$, he also computes $addr^i_{AB}$. Section 3.4.1 discusses how we deal with clock skew.

**Construction.** Tryst$(A, B, p)$ is computed as follows (Figure 3.2):

1. Generate a random key $k_p$.
2. $header \leftarrow \{s, mac\}$, where:

$$s = \{addr^i_{AB}, \mathsf{AES}_{k^{Enc}_{AB}}(k_p)\},$$
$$mac = \mathsf{AES\text{-}CMAC}_{k^{MAC}_{AB}}(s).$$

   $header$ proves to $B$ that $A$ is the sender and $B$ the receiver because only $A$ and $B$ have $k^{Enc}_{AB}$ and $k^{MAC}_{AB}$. Moreover, it proves to $B$ that it was constructed near the current time $T$ because $addr^i_{AB}$ is a cryptographic function of $T$. This provides authenticity.

   To third parties, $mac$ appears to be random because it is computed over the encryption of random key $k_p$, so neither it nor the encipherment of $k_p$ can link it to other messages. $addr^i_{AB}$ is sent "in the clear" and will be used in all messages sent during time interval $T_i$, but $addr^{i1}_{AB}$ and $addr^{i2}_{AB}$ for any $i1 \neq i2$ are unlinkable, thus providing long-term unlinkability.

[2] We make the standard assumptions that AES is a good pseudorandom permutation and that $n \leq 2^{64}$.

3. $ctext \leftarrow \{etext, emac\}$, where:

$$etext = \mathsf{AES\text{-}CTR2}_{k_{p1}}\left(0^{128}, p\right),$$
$$emac = \mathsf{AES\text{-}CMAC}_{k_{p2}}\left(etext\right).$$

   $k_{p1}$ and $k_{p2}$ are pseudo-random keys derived from $k_p$ (e.g., $k_{p1} = k_p$ and $k_{p2} = \mathsf{SHA1}_{128}\left(k_p\right)$). $ctext$ is an encryption of the payload $p$ along with a MAC which, given $k_p$, verifies that the payload was not altered during transmission. We derive two keys from $k_p$ so that different keys are used for encryption and MAC. Since $k_p$ is random, $ctext$ will be different from previous messages even when an identical payload $p$ was sent before. (Note that since key $k_p$ is only used once, the IV $0^{128}$ is still a nonce for that key's "session.")

4. $c \leftarrow \{header, ctext\}$.

A more formal demonstration of Tryst's security properties can be found in Section 3.5.

The overhead (64–80 bytes per message) is acceptable since discovery and binding messages are sent infrequently.

**Message filtering.** Upon reception of such a message, $B$ simply checks his hash table to determine if he has an address $addr^i_{AB}$. If he does, it will be associated with the keys for $A$, which can be used to verify and decrypt the $header$. If not, he can discard the message. Once $header$ is decrypted, he obtains $k_p$, which can be used to decrypt and verify $ctext$ to retrieve the original $p$. In contrast to the straw man mechanisms, this protocol enables devices to discard messages not intended for them efficiently, using hash table lookups instead of costly cryptographic operations.

### 3.3.4 Data Transport: Shroud

Tryst is insufficient for identifier-free data transport because data messages are neither infrequent nor do they have a narrow interface. Thus, to defend against side-channel analysis, we want strong unlinkability rather than just long-term unlinkability. Shroud maintains this property efficiently by leveraging a key assumption about data transport:

> *Connected Communication.* Whereas discovery messages are often sent at times when they will not be received, data messages are only sent after a link has been established. Thus, a sender and receiver can assume that, barring message loss, their messages will be received by their intended recipient.

In effect, this assumption enables Shroud to compute a sequence of unlinkable addresses on a per packet basis, as we will describe shortly.

**Bootstrapping.** We bootstrap Shroud with random session keys $k^{addr}_{s:AB}, k^{Enc}_{s:AB}, k^{MAC}_{s:AB}$ for messages from $A$ to $B$. These keys are exchanged in SlyFi's authentication messages (see Figure 3.1) and thus are protected by Tryst.

**Per-packet unlinkable addresses.** The only design choice in Tryst that sacrifices strong unlinkability is the use of the same $addr^i_{AB}$ for all packets during time interval $i$. Thus, we can essentially use Tryst, provided that we can compute addresses $addr^i_{AB}$ per packet rather than per time interval. To do this in Shroud, $addr^i_{AB}$ is computed as a function of the $i$th *transmission* since link establishment:

$$addr^i_{AB} = \mathsf{AES}_{k^{addr}_{s:AB}}(i), \quad \text{where } i = \text{transmission number}$$

Since a connection has been established, $B$ will receive every packet sent by $A$ on this link barring message loss, and, hence, $B$ only needs to compute address $i+1$ after the receipt of message $i$; i.e., $B$ computes the address he expects in the next message.

Of course, message loss in wireless networks is common, so we would like to be able to tolerate the loss of $w$ consecutive losses for some $w$. Thus, on receipt of message $i$, a receiver computes the $(i+1)$th to $(i+w)$th addresses and inserts them all into its hash table (removing all addresses $j \leq i$). Note that, except for the first message received (e.g., the association request or reply in SlyFi), which requires the computation of $w$ addresses, only one additional address needs to be computed for each subsequent packet sent to $B$, unless there are message losses; $B$ performs no address computation for packets destined for other devices that it overhears. Section 3.4.2 discusses how we choose $w$ and perform link layer retransmissions.

To avoid the overhead of storing these addresses, an alternative design would be to use the same address until a message transmission is successful. However, this approach is problematic when each message is only transmitted a small number of times before giving up, as in 802.11. In these circumstances, the same address can be used on many distinct consecutive packets, which violates strong unlinkability. This is particularly problematic because an adversary that actively jams a channel can cause the loss of packets, forcing a sender to use the same address for many consecutive packets.

**Construction.** With per-packet unlinkable addresses, we could use the Tryst construction and achieve the desired security properties and filter packets efficiently. However, we can make additional optimizations. Shroud$(A, B, p)$ is computed as follows (Figure 3.3):

1. $header \leftarrow addr^i_{AB}$.

| $header = addr^i_{AB}$ | $etext = \mathsf{AES\text{-}CTR2}_{k^{Enc}_{s:AB}}(header, p)$ | $emac = \mathsf{AES\text{-}CMAC}_{k^{MAC}_{s:AB}}(header, etext)$ |
|---|---|---|
| 16 bytes | variable | 16 bytes |

Figure 3.3: Shroud packet format.

Unlike in Tryst, no Shroud address will ever appear in two different messages; thus no one can successfully record and replay them. As a consequence, $addr^i_{AB}$ itself proves to $B$ that $A$ sent the message to $B$ and that it was message transmission $i$, which $B$ expects. This provides authenticity. Each message will have a different address and addresses are strongly unlinkable.

2. $ctext \leftarrow \{etext, emac\}$, where:

$$etext = \mathsf{AES\text{-}CTR2}_{k^{Enc}_{s:AB}}(header, p),$$
$$emac = \mathsf{AES\text{-}CMAC}_{k^{MAC}_{s:AB}}(header, etext).$$

In Tryst, we use a random key to perform the encryption to ensure that the encrypted payload and MAC are unlinkable to previous messages even if their contents are the same. Since each Shroud address is different and is used only once, $header$ effectively serves as a nonce that we can use as an IV to the encryption of the payload. This ensures that $etext$ is unlinkable to previous messages even if their contents are the same and we use the same key $k^{Enc}_{s:AB}$ for encryption. Similarly, we include $header$ in the computation of $emac$ to ensure that it is unlinkable to previous messages even if $p$ is null.

3. $c \leftarrow \{header, ctext\}$.

A more formal demonstration of Shroud's security properties can be found in Section 3.5.

We note that $addr^i_{AB}$ implies the three Ethernet addresses in $p$ so they can be removed, saving 18 bytes. Therefore, Shroud's additional 16 bytes of overhead can, in practice, be a net savings of 2 bytes.

**Message filtering.** As in Tryst, $B$ determines whether a message is for him by looking up $addr^i_{AB}$ in the hash table containing his precomputed addresses. In fact, since the address is located in the same position in both Tryst and Shroud packets, there is no need to distinguish the two message types and a single hash table can be shared by both. The value associated with each address key in the hash table will indicate whether it should be demultiplexed to Tryst or Shroud.

### 3.3.5 Other Protocol Functions

Tryst and Shroud make the crucial elements of a link layer protocol—discovery, binding, and data transport—identifier-free, but other protocol functions must be supported as well. In this section, we explain how SlyFi can support these functions without introducing identifiers.

**Broadcast.** Shroud supports identifier-free broadcast transmissions in managed mode. Broadcasted frames are encrypted with a key and sequence number that are shared by the AP and all clients on the local network. As in 802.11's managed mode, a client forwards frames to the AP that it wishes the AP to broadcast. In Shroud, the transmission to the AP is protected by the per-client shared key used for unicast transmissions. (A client optionally may divulge his identity to all associated stations by including his source address.) Upon reception at the AP, the frame is decrypted and then re-encrypted with the shared broadcast key. The shared key and current sequence number are managed by the AP and conveyed to each of its clients during association. Although SlyFi currently does not support broadcast key revocation, we believe we can apply a scheme similar to that of 802.11i [80]; this is is a topic of future work.

**Binding to higher layer identifiers.** There is often a need to bind higher layer names to link layer addresses. For example, ARP binds Ethernet addresses to IP addresses. Obviously, we do not want to have to re-establish this binding for every Shroud address change. Instead, we have the AP negotiate with each client a pseudonym address that remains consistent for that session, but that is not sent in actual messages. The client informs the AP of its IP to pseudonym binding whenever its IP address changes. Thus, the AP can answer all ARPs.

**Announcement.** Beacons are broadcasted in the clear to announce an 802.11 AP. While SlyFi does not prevent beacons, an AP that wants to hide its presence obviously cannot use them. To discover APs in SlyFi, a client must have the necessary Tryst keys to probe for it. We do not believe this is a hindrance, since existing secure 802.11 networks already require secure out-of-band channels to exchange keys before association.

**Time synchronization.** Beacons are also used to convey timestamps so that clients and APs can synchronize their clocks. With synchronized clocks, clients need only turn on their radios at designated times to receive packets when operating in low power modes. Since only clients on the local network need to synchronize their clocks, this information can be encrypted using the broadcast encryption key described above.

**Roaming.** Clients sometimes use probes or beacons after association to search for better APs to roam to. Using Tryst to send these probes might be expensive if a client sends them frequently. However, these APs are usually in the same administrative domain and thus could share a broadcast key, which could be used to encrypt these messages instead of using Tryst. In addition, Shroud session state could be migrated between APs in advance, similar to how WPA pre-authentication is performed.

**Coexistence.** Our implementation of SlyFi can coexist with normal 802.11 devices because we encapsulate SlyFi messages in management frames that normal 802.11 devices ignore (see Section 3.4.3). The medium access protocol is unchanged.[3] Thus, SlyFi can be deployed incrementally. In a mixed environment, a SlyFi-enabled client can first search for a SlyFi-enabled AP using Tryst probes. If no such AP is found, then a client willing to fall-back to a normal 802.11 AP can listen for beacons and associate normally.

## 3.4 Implementation Details

This section discusses practical considerations involved in implementing Tryst, Shroud, and our SlyFi prototype.

### 3.4.1 Tryst: Practical Considerations

**Clock skew.** In practice $A$ and $B$ will not have perfectly synchronized clocks. To allow for clock skew up to $k \cdot I$ between devices, $B$ should anticipate the addresses that may be used for any messages sent in the time range $[T_{i-k}, T_{i+k}]$ at time $T_i$. Thus, he also inserts (or keeps) $addr_{jB}^{i-k}, addr_{jB}^{i-k+1}, \ldots, addr_{jB}^{i+k}$ into the table for all identities $j$ for which he has keys. Note that messages sent by $A$ will still only use the address $addr_{jB}^{i}$ for one time interval of length $I$. $B$ will simply accept messages with that address for longer.

**Scoped broadcast.** A client may want to send the same discovery message to multiple services (e.g., to discover any one of them). To do this, $A$ constructs one $header$ for each intended recipient, but includes the same $k_p$ in each $header$; e.g., he sends $\{header1, \ldots, headerN, ctext\}$. Hence, any party that can interpret any one $header$ can obtain $k_p$ and decrypt the payload. However, each party can only interpret the $header$ intended for them, so the identities of the other parties remain obscured. One problem

[3] We do not yet support RTS/CTS because our software implementation is not fast enough to perform filtering at the timescale required, but we note that RTS/CTS is rarely used in actual managed networks.

with this approach is that a legitamite receiver can not tell where $ctext$ begins in the message. To discover it, he could try verifying $cmac$ starting from every block offset after his $header$. Alternatively, we could encrypt $N$ within each header itself (at the cost of another 16 byte block per $header$).

**Forward security.** One concern is that $k_{AB}^{addr}$ is stored for a long time and if it is compromised, an adversary could compute the addresses of all messages that $A$ ever sent to $B$. We mitigate this risk by computing a new key each day using a forward-secure pseudo-random bit generator [22]; i.e., the key for day $j$: $k_{AB}^{addr(j)} \leftarrow \mathsf{SHA1}_{128}\left(k_{AB}^{addr(j-1)}\right)$. Both $A$ and $B$ discard the old key and use the new key for computing addresses. The address computation remains the same, but an adversary that obtains $k_{AB}^{addr(j)}$ would only be able to compute addresses for days $j$ and after.

**Public services** . Mutual confidentiality is not always required. For example, 802.11 APs at hotspots are immobile and public and do not need to hide their presense to eavesdroppers—indeed, they *want* to be discovered so that clients will pay to use them. Although Tryst can still be used in these circumstances, it will hinder the discovery of public services that clients do not yet know about; clients would have to discover the existence of these APs through out-of-band means and exchange a key (e.g., by asking the hotstop owner). Futhermore, large networks of public hotspots, such as AT&T Wi-Fi and T-Mobile Hotspots, may have hundreds of thousands or millions of registered clients. Using Tryst, they would need to compute and store new addresses for each of these clients every time interval $T$, which may be burdensome.

When the service does not need to hide its presence during discovery, it can, instead, use a variant that publically authenticates the service and only conceals the identity of the client. For example, JFKi [7], a protocol originally designed for IPSec key exchange, can be used in place of Tryst to establish a session key for Shroud. The basic idea is that, because the service is public, it can announce and authenticate its identity using a certificate chain (as in SSL). With judicious use of nonces, a client can use the standard Diffie-Hellman protocol to exchange a key with the service without revealing its identity to anyone except the service.

### 3.4.2 Shroud: Practical Considerations

**Choosing $w$.** $w$ determines the number of consecutive packet losses Shroud can endure. In practice, burst losses of more than 50 packets are extremely rare on usable links [136] so we use $w = 50$. A larger burst loss will result in a higher level timeout and require re-

establishing the link. The overhead required to maintain these addresses is not prohibitive; even a heavily loaded AP with 256 clients (the max supported by the standard MadWifi driver [110]) requires only 1MB. Most clients, which only have one association at a time, could easily check message addresses in hardware with no more delay or energy than existing NICs. We show in Section 3.7.4 that even software filtering incurs little overhead.

**Acknowledgments.** Every unicast 802.11 data packet is acknowledged by the receiver to manage message loss. In principle, link-layer acknowledgments can simply acknowledge the address of the received Shroud packet, since the sender knows the last address used. However, our current implementation is in software and thus is unable to send this ack within the 16 microseconds allotted to it. Therefore, we currently use software acks that selectively acknowledge cumulative windows of data packets. Each acknowledgment and message retry is processed anew by Shroud.

### 3.4.3 Prototype Implementation

We implemented SlyFi in C++ using the Click Modular Router [94], incorporating Tryst and Shroud with its existing 802.11 implementation (which is by the authors of Roofnet [140]). Since existing 802.11 NICs will not send frames without proper 802.11 headers, each Tryst or Shroud message is encapsulated in an "anonymous" 802.11 header, i.e., one with constant fields and addresses. NICs are placed in promiscuous mode so that they receive all these frames and perform filtering in software. We use the cryptographic routines in libgcrypt [106] and ran our software as a Linux kernel module.

We note that the following evaluation section uses a SlyFi implementation that is slightly different from the protocol we described in the previous sections. This variant, described in [63], uses CBC mode rather than CTR2 mode to encrypt message payloads (see Table 3.2). CTR2 mode is uses the same number of AES operations, has less message overhead, can be computed in parallel, and is provably secure. Therefore, the results we present for SlyFi in the following section are (slightly) conservative.

## 3.5 Formal Security Analysis

In this section, we define and demonstrate SlyFi's confidentiality and unlinkability properties (outlined in Section 3.2.2) with more formal rigor. We omit a more detailed analysis of the authenticity and message integrity properties because both Tryst and Shroud make standard usage of CMAC to authenticate and maintain the integrity of the message payload

($etext$), so they are directly dependent on the security of CMAC [83]. More specifically, we show in this section that the output of Tryst's and Shroud's encapsulation functions are indistinguishable from random bits to a computationally bounded adversary. This property implies confidentiality and unlinkability.

### 3.5.1 Preliminaries

We use the standard definitions of security properties and proof model pioneered in the work of Bellare and Rogaway [20, 23, 139, 25]. The reader is referred to this body of work for in-depth coverage of these definitions and the security proof framework. We also list the definitions relevant to our security analysis in this section.

We use the standard definitions of random functions, permutations, random bits, and nonce-based encryption schemes, which are defined as follows [2, 32, 139]:

- $\mathrm{Rand}(\mathcal{D}, \mathcal{R})$ is the family of pseudorandom functions from $\mathcal{D} \rightarrow \mathcal{R}$.
- $\mathrm{Perm}(\mathcal{D}, \mathcal{R})$ is the family of permutations from $\mathcal{D} \rightarrow \mathcal{R}$.
- $\$^n$ is a sequence of $n$ random bits.
- $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a nonce-based encryption scheme where $\mathcal{K} = \{0,1\}^n$ is the set of keys, $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \rightarrow \mathcal{C}$ is the set of encryption functions, $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{C} \rightarrow \mathcal{M}$ is the set of decryption functions, $\mathcal{M}$ is the set of plaintexts, $\mathcal{C}$ is the set of ciphertexts, and $\mathcal{N} = \{0,1\}^n$ is the set of all nonces. We call $n$ the security parameter of the encryption scheme. Each member of $\Pi(n)$ is distinguished by a key in $\mathcal{K}$.
- $E \xleftarrow{R} \mathcal{E}$ denotes the uniformly random selection of an element $E$ from the set $\mathcal{E}$.

**Encapsulation functions.** We define an *encapsulation function*, a generalization an encryption function, as follows: $F(n, l, k)$ is a family of encapsulation functions $\mathcal{D}_1 \times \mathcal{D}_2 \times \ldots \times \mathcal{D}_l \rightarrow \mathcal{R}$ if $k$ inputs $\mathcal{D}_{i1}, \ldots, \mathcal{D}_{ik} = \{0,1\}^n$, $1 \leq k < l$. We call these inputs the function's *keys*. Each member of $F$ is distinguished by fixing its keys. We call $k \cdot n$ the security parameter of the encapsulation function family. For brevity, we often use $F$ to refer to $F(n, l, k)$ when the parameters are clear from context. Note that a nonce-based encryption function family $\mathcal{E}$ is an instance of $F(n, 3, 1)$.

For brevity, we use $f \xleftarrow{R} F$ to denote $F_{K_1 \ldots K_k}$ where $K_1 \xleftarrow{R} \mathcal{D}_{i1}, \ldots, K_k \xleftarrow{R} \mathcal{D}_{ik}$ and $\mathcal{D}_{i1}, \ldots, \mathcal{D}_{ik}$ are $F$'s keys. In other words, $f$ is a member of $F$ that is selected by randomly selecting each of $F$'s keys.

**Security properties.** We are interested in showing that the Tryst and Shroud encapsulation functions in SlyFi are ENCAP-IND$-CPA-secure, a property that we define below. Intuitively, a function that is ENCAP-IND$-CPA-secure means that its outputs are indistinguishable from random sequence of bits of the same lengths (i.e., $\$^{|\mathrm{tryst}(\ldots)|}$ for tryst), given that the adversary is computationally bounded but can choose all the inputs and observe all the outputs. ENCAP-IND$-CPA is practically identical to IND$-CPA (indistinguishability from random bits under a chosen plaintext attack), another common property in the literature that we define below. The only difference is that ENCAP-IND$-CPA refers to an arbitrary encapsulation function whereas IND$-CPA refers to an encryption scheme. We distinguish these two properties for clarity (Lemma 6 shows that IND$-CPA implies ENCAP-IND$-CPA for the encryption function). These properties imply the standard notions of confidentiality and key-indistinguishability [139] (key-indistinguishability is synonymous with unlinkability — an adversary can not link a message to a particular encryption key).

**Adversaries, oracles, and security games.** In the following definitions, we consider the following standard experiment or "game." A chosen plaintext probabilistic polynomial time adversary $A$ is given access to one of two oracle functions (e.g., one that is our encapsulation function or one that returns truly random bits). We use $A^{\mathrm{oracle}(\cdot)}$ to denote that the adversary $A$ is given the oracle $\mathrm{oracle}(\cdot)$. The adversary does not know which oracle he was given but it can make $q$ queries to it and spend $t$ time computing. In addition, the total size of all queries submitted may not exceed $\sigma$. $(t, q, \sigma)$ are the resource bounds on the adversary. We use only $(t, \sigma)$ to describe the resource bounds of adversaries attacking block ciphers, which have fixed sized inputs and outputs; we omit the number of queries $q$ since $q = O(\sigma)$.

The adversary's goal is to to determine which oracle it was given; it returns 0 if it thinks it is the first, and 1 if it thinks it is the second. We want to minimize an adversary $A$'s *advantage* $\mathsf{Adv}_F^{\mathrm{prop}}(A)$, a measure of how likely that it guesses correctly when attacking security property $\mathrm{prop}$ of encryption scheme $F$. We use the notation $\mathsf{Adv}_F^{\mathrm{prop}}(t, q, \sigma)$ to denote the maximal advantage of all adversaries that spend up to $t$ time computing and make $q$ queries with total size $\sigma$. We define an adversary's advantage more formally below.

This game corresponds to the scenario where an eavesdropper observes all packets sent and wants to determine whether they are random bits or actual Tryst and Shroud packets. This is a strictly easier problem than trying to decrypt them or to link them to a particular

key (i.e., sender or receiver). Therefore, if an adversary can not distinguish, it also can not decrypt or link packets. Moreover, we consider a strong adversary that is able to determine the inputs (e.g., plaintext payloads) of all packets sent. Therefore, this game represents a strictly more powerful adversary than one would probably encounter in practice. The purpose of considering such an adversary is to show that our schemes are secure even with minimal assumptions about the adversary.

**Useful definitions.** We enumerate relevant security definitions and lemmas below.

**Definition 3.5.1** *(*ENCAP-IND$-CPA*) Let $F(n, m, k)$ be a family of encapsulation functions $\mathcal{D}_1 \times \mathcal{D}_2 \times \ldots \times \mathcal{D}_l \rightarrow \mathcal{R}$, and let $A$ be a an algorithm that takes an oracle and returns a bit. Define*

$$\mathsf{Adv}_F^{\text{ENCAP-IND\$-CPA}}(A) \stackrel{\text{def}}{=} \Pr\left[f \xleftarrow{R} F : A^{f(\cdot,\ldots,\cdot)} = 1\right] - \Pr\left[f \xleftarrow{R} F : A^{\$^{|f(\cdot,\ldots,\cdot)|}} = 1\right]$$

We say an encapsulation function family $F(n, l, k)$ with $k$ keys is ENCAP-IND$-CPA-secure if $\mathsf{Adv}_F^{\text{ENCAP-IND\$-CPA}}(A) \leq \epsilon(k \cdot n)$ for all adversaries $A$, where $\epsilon(k \cdot n)$ is negligible.[4]

**Definition 3.5.2** *(*IND$-CPA*) [2, 139] Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based encryption scheme, $n$ be the security parameter, and let $A$ be a nonce-respecting chosen plaintext attack adversary. A nonce-respecting adversary is one that does not submit any query $(N, \cdot)$ after submitting a query $(N, M)$, $N \in \mathcal{N}$. Define*

$$\mathsf{Adv}_{\Pi(n)}^{\text{IND\$-CPA}}(A) \stackrel{\text{def}}{=} \Pr\left[k \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_k(\cdot)}(n) = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K} : A^{\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]$$

We say an encryption scheme family $\Pi(n)$ is IND$-CPA-secure if $\mathsf{Adv}_{\Pi(n)}^{\text{IND\$-CPA}}(A) \leq \epsilon(n)$ for all adversaries $A$, where $\epsilon(n)$ is negligible.

[4] A function $\epsilon(n)$ is negligible if for every polynomial $P(\cdot)$ there exists an $N$ such that for all integers $n > N$, $\epsilon(n) < P(n)$.

**Definition 3.5.3** *(*IND$-CPA*') Let* $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a nonce-based encryption scheme,* $n$ *be the security parameter, and let* $A$ *be a random-nonce chosen plaintext attack adversary. A random-nonce adversary is one that selects* $N \xleftarrow{R} \mathcal{N}$ *for each query* $(N, M)$ *that it submits. Define*

$$\mathsf{Adv}^{\text{IND\$-CPA}'}_{\Pi(n)}(A) \stackrel{\text{def}}{=} \Pr\left[k \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_k(\cdot)}(n) = 1\right] - \Pr\left[k \xleftarrow{R} \mathcal{K} : A^{\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]$$

**Definition 3.5.4** *(*prf*) [32] Let* $F : \mathcal{D} \to \mathcal{R}$ *be a family of functions where* $\mathcal{R} = \{0,1\}^n$ *for* $n \geq 1$*, and let* $A$ *be an algorithm that takes an oracle and returns a bit. Then*

$$\mathsf{Adv}^{\text{prf}}_F(A) \stackrel{\text{def}}{=} \Pr[f \xleftarrow{R} F : A^{f(\cdot)} = 1] - \Pr[\rho \xleftarrow{R} \text{Rand}(\mathcal{D}, \mathcal{R}) : A^{\rho(\cdot)} = 1]$$

**Definition 3.5.5** *(*prp*) [32] Let* $F : \mathcal{D} \to \mathcal{R}$ *be a family of functions* $\mathcal{R} = \{0,1\}^n$ *for* $n \geq 1$*, and let* $A$ *be an algorithm that takes an oracle and returns a bit. Then*

$$\mathsf{Adv}^{\text{prp}}_F(A) \stackrel{\text{def}}{=} \Pr[f \xleftarrow{R} F : A^{f(\cdot)} = 1] - \Pr[\pi \xleftarrow{R} \text{Perm}(\mathcal{D}, \mathcal{R}) : A^{\pi(\cdot)} = 1]$$

**Lemma 1** *(PRF/PRP Switching) [32] Fix* $n \geq 1$*. Let* $A$ *be an adversary that asks at most* $q$ *queries. Then*

$$\left|\Pr[\pi \xleftarrow{R} \text{Perm}(\mathcal{D}, \mathcal{R}) : A^{\pi(\cdot)} = 1] - \Pr[\rho \xleftarrow{R} \text{Rand}(\mathcal{D}, \mathcal{R}) : A^{\rho(\cdot)} = 1]\right| \leq \frac{q(q-1)}{2^{n+1}}$$

*where* $\mathcal{D} = \mathcal{R} = \{0,1\}^n$ *for* $n \geq 1$*.*

### 3.5.2 Security of Each Part

We first show that each constituent part of the Tryst and Shroud encapsulation functions are ENCAP-IND$-CPA- or IND$-CPA-secure. Note that each part itself is also an encapsulation function. In the next section, we demonstrate how to describe the security of the combination of these parts.

**Security of** $addr$**.** Define $\mathcal{NO}[F]$ as shown in Figure 3.4. That is, we can think of a function family $F : \mathcal{K} \times \mathcal{M} \to \{0,1\}^n$, such as a block cipher, as a nonce-based encryption function $\mathcal{NO}[F] : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \{0,1\}^n$ where $\mathcal{NO}[F]_K$ throws away the second argument and just enciphers the nonce (and $\mathcal{N} = \mathcal{M}$). Obviously, we can't extract $M$ from the result, so the decryption function is empty.

**Algorithm** $\mathcal{NO}[F].\text{Encrypt}_K(N, M)$:
**return** $F_K(N)$

**Algorithm** $\mathcal{NO}[F].\text{Decrypt}_K(C)$:
**return** $\star$

Figure 3.4: "Nonce-only" function $\mathcal{NO}[F]$.

**Lemma 2** *(*IND$-CPA *for* $addr$*) Let* $F$ *be a family of functions* $\mathcal{K} \times \mathcal{M} \to \{0,1\}^n$*. Assume that all* IND$-CPA *adversaries are nonce-respecting. Then*

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{NO}[F]}(t, q, \sigma) \;\leq\; \mathsf{Adv}^{\text{prf}}_{F}(t', q, \sigma)$$

*where* $t' = t + O(\sigma)$.

**Proof Sketch** Suppose we have an IND$-CPA adversary $A$ attacking $\mathcal{NO}[F]$. We construct a prf adversary $B$ attacking $F$ as follows:

1. When $A$ makes a query $(N, M)$ to its oracle, pass $N$ to $B$'s oracle.
2. When $A$ outputs $b$, $B$ outputs $b$.

Note that, from the perspective of $A$, $B$ simply implements $\mathcal{NO}[F]$, where $F$ is $A$'s oracle.

$$\begin{aligned}
\mathsf{Adv}^{\text{prf}}_{F}(B) &= \Pr[k \xleftarrow{R} \mathcal{K} : A^{\mathcal{NO}[F]_k(\cdot,\cdot)} = 1] - \Pr[\rho \xleftarrow{R} \text{Rand}(\mathcal{M}, \{0,1\}^n) : A^{\mathcal{NO}[\rho](\cdot,\cdot)} = 1] \\
&= \Pr[k \xleftarrow{R} \mathcal{K} : A^{\mathcal{NO}[F]_k(\cdot,\cdot)} = 1] - \Pr[\rho \xleftarrow{R} \text{Rand}(\mathcal{M}, \{0,1\}^n) : A^{\rho(\cdot)} = 1] \\
&= \Pr[k \xleftarrow{R} \mathcal{K} : A^{\mathcal{NO}[F]_k(\cdot,\cdot)} = 1] - \Pr[\rho \xleftarrow{R} \text{Rand}(\mathcal{M}, \{0,1\}^n) : A^{\$^{|\mathcal{NO}[F]_k(\cdot,\cdot)|}} = 1] \\
&= \mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{NO}[F]}(A)
\end{aligned}$$

The third equality can be shown as follows: $A$ is nonce-respecting so each query it submits to $\rho(\cdot)$ will be different. Moreover, the range of $F$ is $\{0,1\}^n$ so outputs of $\mathcal{NO}[F]_k(\cdot,\cdot)$ are all of length $n$. Therefore, the behavior of $\rho(\cdot)$ is indistinguishable from $\$^{|\mathcal{NO}[F]_k(\cdot,\cdot)|}$. ■

**Security of** $enckey$**.** The following Lemma shows $enckey$ is IND$-CPA-secure.

**Lemma 3** *(*IND$-CPA *for* $enckey$*) Assume that all* IND$-CPA$'$ *adversaries are random-nonce adversaries and all* IND$-CPA *adversaries are nonce-respecting. Then*

$$\mathsf{Adv}^{\text{IND\$-CPA}'}_{\mathcal{NO}[F]}(t, q, \sigma) \;\leq\; \mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{NO}[F]}(t', q, \sigma) + \frac{q(q-1)}{2^{n+1}}$$

*where* $t' = t + O(\sigma)$.

**Proof Sketch** If IND$-CPA$'$ adversary $A$ never submits the same query twice, then it is nonce-respecting and has at most the same advantage as the best nonce-respecting adversary. The likelihood that it submits the same query twice is at most $\frac{q(q-1)}{2^{n+1}}$ (the birthday bound). ■

**Security of $mac$ and $emac$.** The following Lemma from Iwata *et al.* [83] shows that CMAC is prf-secure. By Lemma 2, it is also IND$-CPA-secure if the input is different each time.

**Lemma 4** *(*prf *for CMAC) [83] Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be the underlying block cipher in CMAC (a.k.a. XCBC). Then*

$$\mathsf{Adv}^{\mathrm{prf}}_{\mathrm{CMAC}}(t,q,\sigma) \leq \frac{3\sigma^2}{2^n} + \mathsf{Adv}^{\mathrm{prp}}_{E}(t',\sigma)$$

*where $t' = t + O(\sigma)$.*

**Security of $etext$.** The following Lemma from Rogaway [139] shows that CTR2 mode encryption is IND$-CPA-secure.

**Lemma 5** *(*IND$-CPA *for $etext$) [139] Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be the underlying block cipher in CTR2. Let $n, \sigma \geq 1$. Then*

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathrm{CTR2}}(t,q,\sigma) \leq \frac{\sigma^2}{2^n} + \mathsf{Adv}^{\mathrm{prp}}_{E}(t',\sigma)$$

*where $t' = t + O(\sigma)$.*

Note that although we always use $0^{128}$ as the IV in Tryst's encryption of the payload $p$, we use a different encryption key each time. Therefore, the protocol is still nonce-respecting in that the IV is only used once per encryption key.

**Combining components.** Finally, the following two Lemmas demonstrate how to describe the security when combining IND$-CPA-secure and ENCAP-IND$-CPA-secure constituent parts.

**Lemma 6** *(*IND$-CPA $\Rightarrow$ ENCAP-IND$-CPA*) Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Assume that all* ENCAP-IND$-CPA *and* IND$-CPA *adversaries are nonce-respecting. Then*

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{E}}(t,q,\sigma) \leq \mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(t',q,\sigma)$$

*where $t' = t + O(\sigma)$.*

**Algorithm** $\mathcal{FC}[f_1, \ldots, f_m](D_{11}, \ldots, D_{1k}, \ldots, D_{m1}, \ldots, D_{mk})$:
**return** $f_1(D_{11}, \ldots, D_{1k}) \| \ldots \| f_m(D_{m1}, \ldots, D_{mk})$

Figure 3.5: Encapsulation function "combiner" $\mathcal{FC}[F_1, \ldots, F_m]$

**Proof Sketch** Suppose we have an ENCAP-IND$-CPA adversary $A$ attacking $\mathcal{E}$. Construct an IND$-CPA adversary $B$ attacking $\Pi(n)$ as follows:

1. When $A$ asks a query to its oracle, pass it to $B$'s oracle.
2. When $A$ outputs bit $b$, output $b$.

$B$'s oracle is the same as $A$'s oracle (an encryption function chosen randomly from $\mathcal{E}$, or a random sequence of bits of the same length). Therefore $B$ will answer correctly if and only if $A$ answers correctly. ■

Let $F_1(n, l_1, k_1), \ldots, F_m(n, l_m, k_m)$ be encapsulation function families such that $F_i : \mathcal{D}_{i1} \times \ldots \times \mathcal{D}_{ik} \rightarrow \mathcal{R}_i$. Define a encapsulation function family $\mathcal{FC}[F_1, \ldots, F_m]$ as in Figure 3.5, where each distinct member of $\mathcal{FC}[F_1, \ldots, F_m]$ is defined by choosing one function in each family $F_i$. That is, a function $\mathcal{FC}[f_1, \ldots, f_m]$ simply comprises one function $f_1 \in F_1$, $f_2 \in F_2$, etc., takes as input all the inputs of each $f_i$, and returns the concatenation of the result of each. The members of $\mathcal{FC}[f_1, \ldots, f_m]$ are distinguished by the combined keys of all its components. Therefore, it has $\sum_i^m l_i$ inputs and its security parameter is $n \cdot \sum_i^m k_i$.

**Lemma 7** *(*ENCAP-IND$-CPA *Combiner) Let $F_1(n, l_1, k_1), \ldots, F_m(n, l_m, k_m)$ be encapsulation function families such that $F_i : \mathcal{D}_{i1} \times \ldots \times \mathcal{D}_{il_i} \rightarrow \mathcal{R}_i$. Then*

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_m]}(t, q, \sigma) \leq \sum_i^m \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_i}(t', q, \sigma)$$

*where $t' = t + O(\sigma)$.*

**Proof Sketch** We prove this claim using induction on $m$.

**Base case.** $\mathcal{FC}[f_1] = f_1$. Therefore, $\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1]}(t, q, \sigma) = \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_1}(t, q, \sigma)$.

**Inductive case.** Assume the inductive hypothesis:

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_m]}(t, q, \sigma) \leq \sum_i^m \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_i}(t', q, \sigma)$$

We want to show the following:

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_{m+1}]}(t,q,\sigma) \leq \sum_{i}^{m+1} \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_i}(t',q,\sigma)$$

Suppose we have an ENCAP-IND\$-CPA adversary $B$ attacking $\mathcal{FC}[F_1,\ldots,F_{m+1}]$. To prove the above claim, we construct two ENCAP-IND\$-CPA adversaries from $B$: $A_1$ attacks $F_{m+1}$ and $A_2$ attacks the inductive hypothesis.

**Adversary $A_1$:**

1. For $i \in \{1,\ldots,m\}$ construct $f_i \xleftarrow{R} F_i$.
2. When $B$ makes an oracle query $(D_{1,1},\ldots,D_{1,l_1},\ldots,D_{m+1,1},\ldots,D_{m+1,l_{m+1}})$, pass $(D_{m+1,1},\ldots,D_{m+1,l_{m+1}})$ to $A_1$'s oracle to obtain the result $o$. Return $f_1(D_{1,1},\ldots,D_{1,l_1})\|\ldots\|f_m(D_{m,1},\ldots,D_{m,l_m})\|o$.
3. When $B$ outputs bit $b$, output $b$.

Note that in the case where $A_1$'s oracle is $f_{m+1}$, this adversary implements $\mathcal{FC}[F_1,\ldots,F_{m+1}]$ as the oracle for $B$. Therefore, $A_1$ has the following advantage:

$$\begin{aligned}\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_{m+1}}(A_1) \;=\; & \Pr\left[f_1 \xleftarrow{R} F_1,\ldots,f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}[f_1,\ldots,f_{m+1}]}=1\right] - \\ & \Pr\left[f_1 \xleftarrow{R} F_1,\ldots,f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}_R[f_1,\ldots,f_{m+1}]}=1\right]\end{aligned}$$

where $\mathcal{FC}_R[f_1,\ldots,f_{m+1}]$ is a function that takes $(D_{1,1},\ldots,D_{1,l_1},\ldots,D_{m+1,1},\ldots,D_{m+1,l_{m+1}})$ returns $\$^{|f_1(D_{1,1},\ldots,D_{1,l_1})|}\|\ldots\|\$^{|f_m(D_{m,1},\ldots,D_{m,l_m})|}\|f_{m+1}(D_{m+1,1},\ldots,D_{m+1,l_{m+1}})$.

**Adversary $A_2$:**

1. Construct $f_{m+1} \xleftarrow{R} F_{m+1}$.
2. When $B$ makes an oracle query $(D_{1,1},\ldots,D_{1,l_1},\ldots,D_{m+1,1},\ldots,D_{m+1,l_{m+1}})$, pass $(D_{1,1},\ldots,D_{1,l_1},\ldots,D_{m,1},\ldots,D_{m,l_m})$ to $A_2$'s oracle to obtain the result $o$. Return $o\|\$^{|f_{m+1}(D_{m+1,1},\ldots,D_{m+1,l_{m+1}})|}$.
3. When $B$ outputs bit $b$, output $b$.

Note that in the case where $A_2$'s oracle is $\$^{|\mathcal{FC}[F_1,\ldots,F_m]|}$, this adversary implements $\$^{|\mathcal{FC}[F_1,\ldots,F_{m+1}]|}$. Therefore, $A_2$ has the following advantage:

$$\begin{aligned}\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_m]}(A_2) &= \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}_R[f_1,\ldots,f_{m+1}]} = 1\right] - \\ &\quad \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\$^{|\mathcal{FC}[f_1,\ldots,f_{m+1}]|}} = 1\right]\end{aligned}$$

where $\mathcal{FC}_R[f_1, \ldots, f_{m+1}]$ is defined above.

We then have:

$$\begin{aligned}\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_{m+1}]}(B) &= \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}[f_1,\ldots,f_{m+1}]} = 1\right] - \\ &\quad \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\$^{|\mathcal{FC}[f_1,\ldots,f_{m+1}]|}} = 1\right] \\ &= \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}[f_1,\ldots,f_{m+1}]} = 1\right] - \\ &\quad \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}_R[f_1,\ldots,f_{m+1}]} = 1\right] + \\ &\quad \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\mathcal{FC}_R[f_1,\ldots,f_{m+1}]} = 1\right] - \\ &\quad \Pr\left[f_1 \xleftarrow{R} F_1, \ldots, f_{m+1} \xleftarrow{R} F_{m+1} : B^{\$^{|\mathcal{FC}[f_1,\ldots,f_{m+1}]|}} = 1\right] \\ &= \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_{m+1}}(A_1) + \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathcal{FC}[F_1,\ldots,F_m]}(A_2) \\ &\leq \sum_i^{m+1} \mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{F_i}(t', q, \sigma)\end{aligned}$$

The first equality is by the definition of ENCAP-IND$-CPA. The second equality is due to the addition and subtraction of the same quantity. The third equality is by substitution with the adversaries' advantages shown above. The final inequality is by the inductive hypothesis. ■

### 3.5.3 Security of Tryst and Shroud

Finally, in this section, we show how to describe the security of the Tryst and Shroud encapsulation functions, which are combinations of functions described in the previous section.

**Theorem 3.5.1** *(*ENCAP-IND$-CPA *for* tryst*) Let* $\text{tryst} : \mathcal{K} \times \mathcal{N} \times \mathcal{M}$ *be the encapsulation function described in Section 3.3.3, where* $\mathcal{N}$ *is the value enciphered to create the*

*address $addr$ and $\mathcal{M}$ is the plaintext payload. If $\mathsf{SHA1}_{128}(\cdot)$ is a random oracle, then*

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\text{tryst}}(t,q,\sigma) \leq \frac{9\sigma^2}{2^n} + 5 \cdot \mathsf{Adv}^{\text{prp}}_{\text{AES}}(t',\sigma)$$

*where $t' = t + O(\sigma)$ for any adversary that makes at most one query per time interval (i.e., chooses different $N \in \mathcal{N}$ for each query).*

**Proof Sketch** We analyze an adversary's advantage by determining its maximal advantage of attacking each individual part of the output: $addr = addr^i_{AB}$, $enckey = \mathsf{AES}_{k^{Enc}_{AB}}(k_p)$, $mac = \mathsf{AES\text{-}CMAC}_{k^{MAC}_{AB}}(s)$, $etext = \mathsf{AES\text{-}CTR2}_{k_{p1}}(0^{128}, p)$, and $emac = \mathsf{AES\text{-}CMAC}_{k_{p2}}(etext)$. Note that the secret key input the encryption scheme in each of these parts is different ($k_{p1}$ and $k_{p2}$ are random if $\mathsf{SHA1}_{128}(\cdot)$ is a random oracle). Therefore, each component is selected randomly from their respective encryption function families. Thus, we can use Lemma 7 to obtain a bound on the advantage the adversary has on the tryst function as a whole.

If an adversary makes at most one query per time interval, then the $addr$ part of the output is an instance of $\mathcal{NO}[\text{AES}]$ because the nonce input will be different each time. By Lemma 2 and Lemma 6, an IND\$-CPA adversary's advantage attacking the $addr$ part is at most $\mathsf{Adv}^{\text{prf}}_{\text{AES}}(t,q) \leq \mathsf{Adv}^{\text{prp}}_{\text{AES}}(t,q) + \frac{q(q-1)}{2^{n+1}}$ (by Lemma 1).

The $enckey$ part is an instance of $\mathcal{NO}[\text{AES}]$ where each input is random. By Lemma 3, the ENCAP-IND\$-CPA adversary's advantage attacking the $enckey$ part is at most $\mathsf{Adv}^{\text{IND\$-CPA}}_{\mathcal{NO}[\text{AES}]}(t,q) + \frac{q(q-1)}{q^{n+1}}$. By applying Lemma 2, Lemma 6, and Lemma 1 as before, we obtain that the advantage is at most $\mathsf{Adv}^{\text{prp}}_{\text{AES}}(t,q) + \frac{q(q-1)}{2^n}$.

The $mac$ part is an instance of CMAC. By Lemma 4, the prf adversary's advantage attacking the $mac$ part is at most $\frac{3\sigma_m^2}{2^n} + \mathsf{Adv}^{\text{prp}}_{\text{AES}}(\bar{t},\sigma_m)$ where $\sigma_m$ is the sum of the sizes of the inputs to CMAC and $\bar{t} = t + O(\sigma_m)$. The output size is the same for all inputs and all inputs are different, so the prf property becomes indistinguishable from the ENCAP-IND\$-CPA property and this is also a bound on the ENCAP-IND\$-CPA advantage of attacking the $mac$ part.

The $etext$ part is an instance of CTR2. By Lemma 5, the IND\$-CPA adversary's advantage is at most $\frac{\sigma_p^2}{2^n} + \mathsf{Adv}^{\text{prp}}_{\text{AES}}(t',\sigma_p)$ where $\sigma_p$ is sum of the sizes of plaintexts $p$ and $\hat{t} = t + O(\sigma_p)$. By Lemma 6, this is also the bound on the ENCAP-IND\$-CPA advantage of attacking the $etext$ part.

The $emac$ part is an instance of CMAC. By Lemma 4, the prf adversary's advantage attacking the $mac$ part is at most $\frac{3(\sigma_p)^2}{2^n} + \mathsf{Adv}^{\text{prp}}_{\text{AES}}(\hat{t},\sigma_p)$ where $\sigma_p$ is sum of the sizes of

plaintexts $|p|$ and $\hat{t} = t + O(\sigma_p)$. The output size is the same for all inputs, so the $\mathrm{prf}$ property becomes indistinguishable from the ENCAP-IND$-CPA property and this is also a bound on the ENCAP-IND$-CPA advantage of attacking the $mac$ part.

The tryst encapsulation function is just a concatenation of the aforementioned parts, so by Lemma 7, an ENCAP-IND$-CPA adversary's advantage attacking tryst is at most the sum of the aforementioned advantages:

$$\begin{aligned} & 2 \cdot \mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(t, q) + \mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(\bar{t}, \sigma_m) + 2 \cdot \mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(\hat{t}, \sigma_p) + \frac{1.5 \cdot q(q-1)}{2^n} + \frac{3\sigma_m^2}{2^n} + \frac{4\sigma_p^2}{2^n} \\ \leq \; & 5 \cdot \mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(t', \sigma) + 9 \cdot \frac{\sigma^2}{2^n} \end{aligned}$$

where $\sigma$ is the sum of the sizes of all input to the tryst function (i.e., the length of each packet) and $t' = t + O(\sigma)$. ■

Tryst takes $4 = O(1)$ keys, and the bound given by Theorem 3.5.1 is clearly negligible in $4n$ assuming that $\mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(t', q, \sigma)$ is also negligible in $n$ (i.e., AES is a good random permutation). Thus, Tryst is ENCAP-IND$-CPA-secure if we limit the adversary to one query per time interval. However, note that we only require this limitation in order to make the adversary nonce-respecting for the $addr$ part of the Tryst function. Therefore, if we consider all parts of the Tryst function except the $addr$ part, Tryst is ENCAP-IND$-CPA-secure even without this restriction. This result is intuitively obvious since the $addr$ is fixed for each time interval so multiple queries would result in the same $addr$, which is clearly not random.

**Theorem 3.5.2** *(*ENCAP-IND$-CPA *for* shroud*) Let* $\mathrm{shroud} : \mathcal{K} \times \mathcal{N} \times \mathcal{M}$ *be the encapsulation function described in Section 3.3.4, where* $\mathcal{N}$ *is the value enciphered to create the address* $addr$ *and* $\mathcal{M}$ *is the plaintext payload. Then*

$$\mathsf{Adv}^{\text{ENCAP-IND\$-CPA}}_{\mathrm{shroud}}(t, q, \sigma) \leq \frac{5\sigma^2}{2^n} + 3 \cdot \mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(t', \sigma)$$

*where* $t' = t + O(\sigma)$ *for any nonce-respecting adversary (i.e., chooses different* $N \in \mathcal{N}$ *for each query).*

The proof is similar to the proof for Theorem 3.5.1, so we omit the details here. The only significant difference is that we do not limit the adversary to one query per time interval because the nonce input to $addr$ is different for each packet in Shroud.

Shroud takes $3 = O(1)$ keys, and the bound given by Theorem 3.5.2 is clearly negligible in $4n$ assuming that $\mathsf{Adv}^{\mathrm{prp}}_{\mathrm{AES}}(t', \sigma)$ is also negligible in $n$. Thus, Shroud is ENCAP-IND$-CPA-secure.

## 3.6 Robustness Against Attacks

SlyFi is designed to achieve unlinkability, confidentiality, authenticity, and message integrity under the threat model described in Section 3.2.1: we assume scenarios where adversaries are passive eavesdroppers, they are not privy to any of the secret keys shared between senders and receivers, they cannot gain much useful information from timing and physical layer side-channels, and there are a sufficient number of honest senders transmitting simultaneously. The empirical results we presented in Section 3.2.1 (in addition to subsequent work [18, 19]) demonstrate that these assumptions are reasonable in many circumstances. In this section, we discuss the resilience of SlyFi to more powerful adversaries when these assumptions are relaxed.

### 3.6.1 Replay and Active Attacks

An adversary can replay a Tryst message for up to $k \cdot I$ time units after it was initially sent and the receiver will still process it. This is because the receiver must account for $k \cdot I$ units of clock skew. If the receiver is present during the replay, then it will respond with a Tryst probe response. Without additional information, the adversary can not determine that this message is a response to his probe, but even if he can, it does not explicitly reveal any information about the receiver.

However, if an adversary knows the sender or intended recipient of a Tryst probe, the presence or absence of a reply may reveal additional information. For example, an adversary can replay a probe at another location to see if the recipient responds. However, these attacks can only be performed for the short time interval that an address is valid and can be mitigated by simple countermeasures. For example, since an adversary cannot distinguish the content of a response from any other message, if random delays were added to probe responses, an adversary might lose them in the noise of frequent background traffic. In addition, receivers can cache valid probe and authentication requests that they receive for the duration they are valid and ignore replays of those messages. We did not implement these countermeasures since these attacks assume adversaries already know the sender or intended recipient of a message, which can not be learned from the message's contents alone.

Shroud messages that are replayed will not be processed by the receiver because each address is only valid for a single message.

### 3.6.2 Denial-of-Service Attacks

Wireless is a broadcast medium and the unlicensed nature of the 802.11 spectrum makes it easy for an adversary to jam a channel by broadcasting "junk." Junk messages will interfere with valid messages sent and devices that receive junk messages may be forced to process them. An important metric for measuring how susceptible a wireless protocol is to such denial-of-service attacks is the amount of energy an adversary has to spend to prevent a receiver from receiving useful messages [69]. In this respect, SlyFi is no more susceptible to denial-of-service attacks than 802.11 because, as we demonstrate in Section 3.7, it can process all messages that it receives, adversarial or otherwise, at the maximum wireless bit rate. We note that this is in contrast to the straw men protocol we described in Section 3.3.1. Junk messages can interfere with valid SlyFi messages, but the same is true for 802.11 messages.

### 3.6.3 Logical Layer Side-Channel Attacks

To achieve strong unlinkability, SlyFi assumes that an adversary can not distinguish the packets belonging to different packet streams sent simultaneously. In practice, some side-channels, such as packet sizes and the time when packets are sent may hint at the packet stream each packet belongs to. Nonetheless, since the majority of side-channel attacks on short term packet linkability require a relatively large number of linked packets to be effective (e.g., [142, 171, 172]), we believe that Shroud will still make such attacks much more difficult to carry out in environments with many overlapping packet streams.

In scenarios where there is only one transmitter, there will obviously be no ambient traffic to mask that transmitter's packet stream. In Section 3.2.1, we described evidence that may packet streams do overlap in practice, so we believe that these scenarios are rare. In scenarios where packet sizes and timing may be sufficiently distinguishing to perform side channel attacks, existing countermeasures such as packet padding and cover traffic can complement SlyFi and prevent these attacks (albeit at significantly higher cost). Deciding when to employ these additional countermeasures is an important area of future work.

### 3.6.4 Physical Layer Side-Channel Attacks

Radio fingerprints at the physical layer may be detected by adversaries with special equipment [16, 36, 70, 143]. SlyFi does not conceal these fingerprints in Tryst or Shroud packets. However, we believe that such equipment, such as high precision signal analyzers that cost tens of thousands of dollars, are out of reach of casual eavesdroppers and do not

expect large scale surveillance networks constructed using them soon. We also believe that concealing these fingerprints with hardware modifications is not technically difficult (e.g., Brik *et al.* [36] used fixed calibration errors to fingerprint wireless cards, so inserting randomized noise to these errors would mask the fingerprints). However, doing so would likely add to the cost of wireless devices, so a thorough understanding of the economic trade-offs is a crucial topic of future work to determining whether deploying such countermeasures at scale is practical.

More accessible physical layer information such as signal strength may also sometimes act as side channels that link messages together at short time scales. However, these implicit identifiers are typically less distinguishing than logical layer fingerprints. For example, Tao *et al.* [153] and Bahl and Padmanabhan [12] show that signal strength measurements from multiple locations can be used to distinguish the rough locations where they originated. Therefore, when devices are stationary, they can approximately distinguish the messages sent by each one. Nonetheless, Bauer *et al.* [18, 19] find that, while using multiple signal strength measurements to distinguish messages sources can be moderately accurate, the error rate is sufficiently high that certain side-channel attacks (e.g., [105]) have much lower success rates (e.g., 50% vs. 95%). Moreover, collecting sufficient signal strength measurements requires multiple monitoring points and their accuracy can be reduced by varying a client's transmit power [18, 19].

### 3.6.5 Tryst Key Compromise

Tryst secret keys are bootstrapped through an out-of-band mechanism that we assume is secure. As with 802.11 WPA today, a secret password may be used to derive these keys. The strength of Tryst's security would then depends on the strength of the password, which may be weak if it is defined by a human. One way to limit the window of vulnerability to weak passwords is to use keys derived from the password only for the first Tryst probe and response. These messages can contain new truly random keys used for future discovery attempts (using different keys for each receiver). This will ensure that an adversary that compromises the initial key can only compromise the confidentiality and unlinkability of messages if they overhear the first Tryst probe and response. More generally, a sender and receiver can renegotiate keys each time they engage in a session.

If a secret key is compromised in some other manner (e.g., if an adversary extracts it from one of the devices manually), Tryst can not protect the confidentiality or unlinkability of future messages. However, the confidentiality and unlinkability of messages transmitted on previous days remains protected because Tryst uses a forward-secure random bit generator to change keys each day (see Section 3.4.1).

### 3.6.6 Shroud Key Compromise

Shroud session keys are less likely to be compromised because they will only be used for short time periods and need not be resident on long term storage. Moreover, they are randomly generated. To reduce the likelihood of compromise even further, senders and receivers can periodically renegotiate new Shroud keys during a session in a manner similar to WPA. If a set of Shroud keys are compromised, the confidentiality and unlinkability of the session that those keys are protecting are lost.

### 3.6.7 Adversarial Senders and Receivers

SlyFi does not protect users if they do not trust the device that they are communicating with (e.g., if a user does not trust the AP he is using or vis versa). In such scenarios, users could instead associate with untrusted devices anonymously, as users do at open 802.11 APs today if they change their MAC address. To protect the information they transmit through untrusted devices, users would have to rely on existing heavy-weight measures, such as encrypting and tunneling all traffic through a trusted VPN or mix network. SlyFi is an optimization to these solutions when both ends of the wireless link trust each other.

## 3.7 Performance Evaluation

We evaluate two key areas. First, we examine how quickly we can discover and set up a link with Tryst. A quick link establishment improves usability both by reducing the delay before communication can begin and by preventing noticeable interruptions when roaming between APs. Second, we examine the performance penalty incurred when using Shroud to deliver data traffic. In general, we find that SlyFi performs comparably to 802.11 using WPA and substantially out-performs the straw man mechanisms we discussed.

### 3.7.1 Comparison Protocols

We compare our SlyFi implementation to the following baseline protocols and alternatives:

**wifi-open.** The baseline implementation of 802.11 without WEP or WPA in Click. SlyFi uses the same components, simply encapsulating the original packets where needed, allowing us to make a direct comparison to a software implementation without our mechanisms.

**wifi-open-driver.** The 802.11 implementation in the MadWifi driver/firmware [110]. We compare to this second baseline since wifi-open has additional overhead when used for data transport, which we discuss in Section 3.7.4. Neither wifi-open nor wifi-open-driver meet any of our security requirements.

**wifi-wpa.** A baseline implementation of 802.11 with WPA, which provides authentication, message integrity, and confidentiality, but not unlinkability as messages still include Ethernet addresses and network names. We use the standard WPA client and AP implementations on Linux [77], which run on top of wifi-open-driver, so wifi-wpa does not incur the overhead mentioned above. We run wifi-wpa using PSK user authentication and CCMP encryption. PSK is the most widely used standard in small private networks. CCMP is comparable to SlyFi's payload encryption, as both are built around AES. However, wifi-wpa performs AES operations using dedicated hardware on the 802.11 NIC, while SlyFi performs it in software. To compensate, we also evaluate SlyFi with simulated hardware we discuss in Section 3.7.4.

**public key.** The straw man alternative to Tryst for discovery and link setup discussed in Section 3.3.1.

**symmetric key.** The other straw man alternative to Tryst discussed in Section 3.3.2. public key and symmetric key still use Shroud once a link is established (i.e., they only replace Tryst in Figure 3.1).

**armknecht.** A previous 802.11 frame encryption proposal [11] that is an alternative to Shroud for data transport.[5] Like Shroud, armknecht computes per-packet addresses, but only for the next packet it expects, so it would perform comparably when there is no packet loss or competing traffic. However, a receiver that receives a message without one of its known addresses performs a number of cryptographic operations comparable to symmetric key before discarding it. This is because it treats a packet it does not have an address for as an indication of potential loss and uses these operations to try to recover from it. In contrast, Shroud simply precomputes more addresses to manage loss.

### 3.7.2 Setup

We deploy these protocols on a number of Soekris net4801 low-power devices [146]. These devices have hardware comparable to common 802.11 APs and embedded consumer devices. While laptops have more powerful hardware, we demonstrate that our

[5] Our implementation uses AES as the cipher.

mechanisms are usable even on more constrained devices. In our experiments, we designate each device as either an AP or a client.

Each device has a 266 Mhz 586-class Geode processor, 256 MB of RAM, 1 GB of flash storage, and one CM9 Atheros 802.11a/b/g miniPCI card. Each device runs a minimal version of Linux 2.6.16.13. 802.11 frames are sent and received from a raw 802.11 radiotap device created by the standard MadWifi driver. We operate on 802.11a channel 40 to avoid interference from more common 802.11b/g devices. To make a fair comparison, management frames in all protocols are transmitted at the base rate (6Mbps), as is dictated by the 802.11 standard, while data frames are transmitted at the peak rate (54Mbps).

### 3.7.3 Discovery and Link Setup Performance

To evaluate how long a client would need wait before it can start transferring data, we measure the *link setup time*, defined as the delay between when a client begins probing for APs and when it can deliver packets on the established link. In all the protocols except for wifi-wpa, packets can be delivered once an *association response* message is received (see Figure 3.1). wifi-wpa has an additional key negotiation phase after association.

The parameters that impact link setup time are: the number of client accounts on an AP, the number of networks that each client probes for, and the amount of background probing traffic that is overheard by APs and clients. Our results show that, in contrast to public key and symmetric key, Tryst has faster link setup times than wifi-wpa and scales as gracefully as wifi-open when varying each of these parameters. Moreover, the cost of periodically computing addresses is trivial. Unless otherwise indicated, each data point presented in this section is the mean of 30 trials.

**Keys per service.** An AP maintains one key for each client it has an account for, and the total number of keys can impact link setup time. Real networks manage various numbers of client accounts; e.g., home networks will likely have less than a dozen, while the wireless network at the Intel Research lablets, a fairly small organization, has 721. Carnegie Mellon University's wireless network, which may be representative of a large organization, has 36,837 at the time of this writing.

Figure 3.6 shows the link setup time for a client that sends one probe in search of a nearby AP as we vary the number of keys per AP. Before each probe, the AP has its keys sorted in random order, and thus, the performance of the symmetric key protocol degrades with the number of keys, since it must check a *discovery* message against all keys until it finds one that successfully validates the MAC on the header. The other protocols have setup times that are independent of the number of keys per AP. Note however, that the

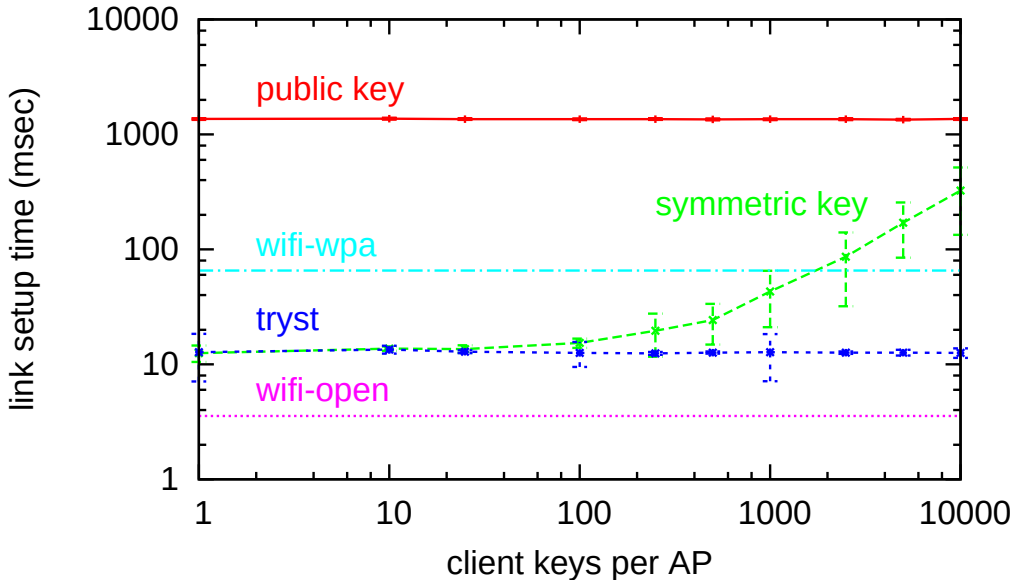


Figure 3.6: Association delay as the number of keys per AP varies. The client probes for 1 AP. Error bars indicate one standard deviation.

public key protocol is still more expensive than all the others, even when the AP has 10,000 keys. Furthermore, although Tryst imposes some overhead over the wifi-open protocol, it has link setup times that are less than wifi-wpa and that, at ∼15 ms, are below the variance in Internet round trip times.

**Probes per client.** Since private APs cannot send beacons, a client may need to probe for several different networks to figure out which one is present. In 802.11, these probes usually contain the names of networks with which the client has previously associated. Figure 3.7 shows a cumulative distribution function of the number of unique network names probed for by clients in three wireless traces (described in Table 3.1). While most users probe for a small number of networks, at least 4% of users in all traces probe for more than 10 and some probe for more than 100.[6] Therefore, it is important that link setup time does not grow substantially with the number of probes.

Figure 3.8 shows the link setup time as a function of the number of different probes a client sends, which are sent as fast as possible. The number of keys per AP is fixed at

[6]Users in the SIGCOMM trace probed for more networks because each SIGCOMM AP had a different network name and the network often was unavailable, prompting clients to send probes for names deeper into their list of networks. We ignored broadcast and random network names.

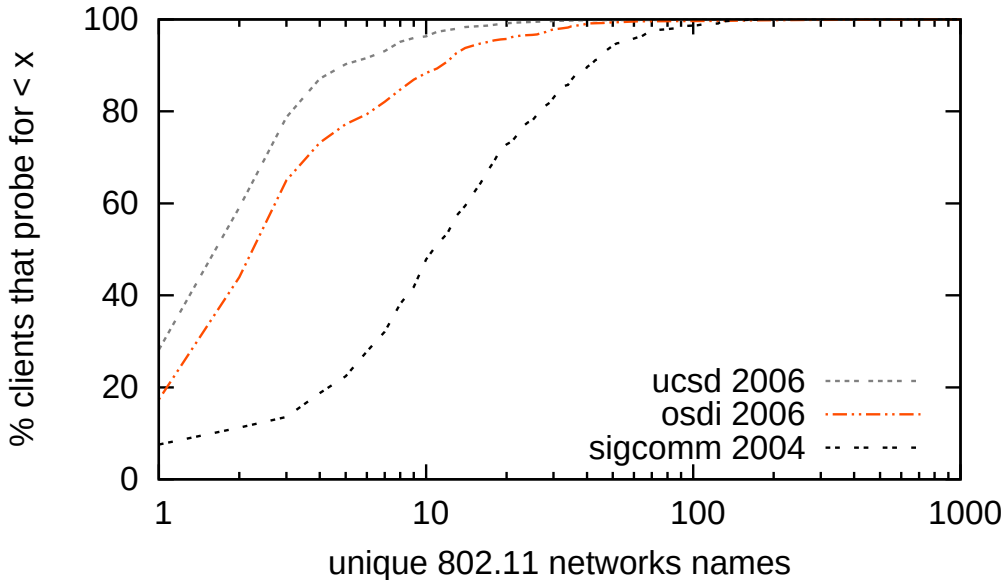


Figure 3.7: CDF of the number of unique network names probed for by each client in three empirical 802.11 traces.

500. For a fair comparison, Tryst sends a separate message for each probe instead of using scoped broadcast (discussed in Section 3.3.3). We omit the line for wifi-wpa because the standard probing behavior is different and incurs more delays. If it also sent probes as fast as possible, it would have scaling behavior similar to the wifi-open line since the probes they send are the same.

Although all protocols scale with the number of probes sent, as there is overhead in processing them and limited bandwidth in the medium, the slopes of the two straw man protocols are steeper, indicating that they incur more overhead per probe. The slopes of the Tryst and wifi-open lines are similar, and both have setup times of at most $\sim$50 ms even when clients send 50 probes.

**Performance breakdown.** Table 3.3 shows the breakdown of link setup time for clients that send 5 probes and APs with 500 keys. The public key protocol spends most of its time in the first two phases, since it must process most public key encryptions, decryptions, and signature checks here. These operations are two orders of magnitude slower than the symmetric key analogs. Nonetheless, the symmetric key protocol still spends significant time in the probing phase, because when the AP first receives a probe, it may try to verify

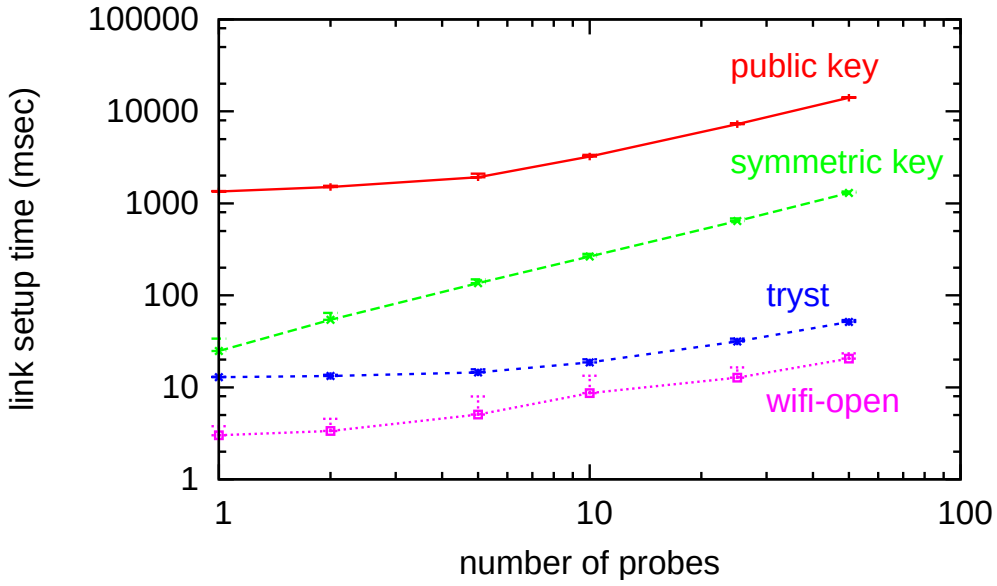


Figure 3.8: Link setup time as the number of probes each client sends varies. The AP has 500 keys. Error bars indicate one standard deviation.

| | probing | openauth | associate | wpa-key | total |
|---|---|---|---|---|---|
| public key | 886.1 | 895.2 | 146.2 | NA | 1927.6 |
| symmetric key | 120.2 | 8.6 | 6.9 | NA | 135.6 |
| tryst | 3.3 | 5.1 | 6.2 | NA | 14.5 |
| wifi-open | 1.4 | 1.5 | 2.2 | NA | 5.1 |
| wifi-wpa | 0.1 | 6.9 | 0.8 | 57.5 | 65.3 |

Table 3.3: Breakdown of link setup time for a client that probes for 5 different networks and an AP with 500 keys. Times are in milliseconds. Each phase corresponds to request/response messages in Figure 3.1, except wpa-key, which involves 2 round trips after association to derive session keys in wifi-wpa.

the MAC with all its keys. Subsequent phases are faster because both the client and the AP re-sort their keys in MRU order, so the expected number of keys they must try before finding the right one decreases appreciably.

Tryst has similar performance to the symmetric key protocol during the last three phases because the number of cryptographic operations is identical. However, the first

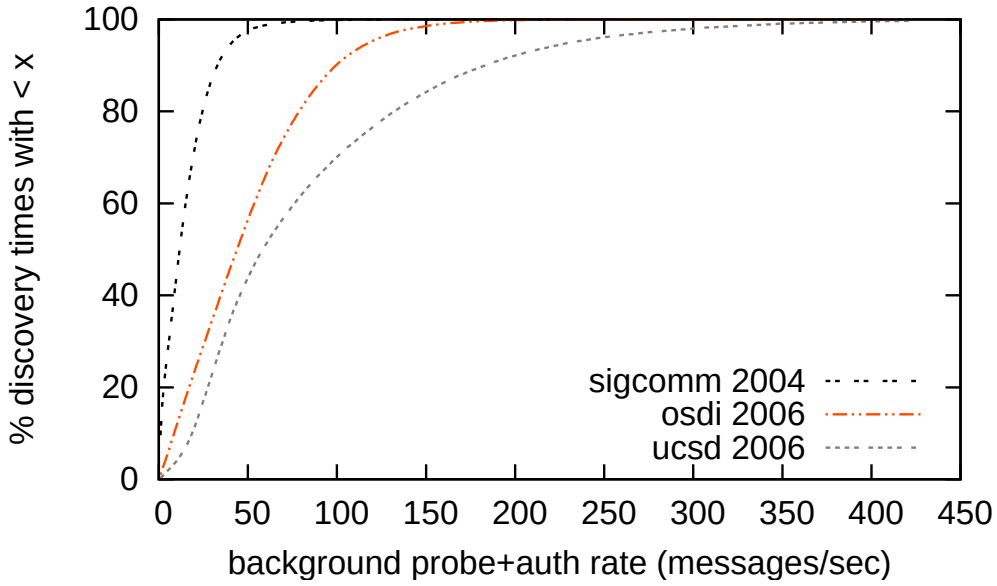


Figure 3.9: CDF of background probe and authentication messages observed each second where discovery was taking place in each of three 802.11 traces. We only count times when there was at least one probe (i.e., times when discovery was taking place).

phase is much faster because the AP looks up the address in a hash table to determine which key to use to verify the message. The open authentication and association phases take slightly longer because they involve computing the initial $w$ Shroud addresses. Even when performing these operations, in addition to standard 802.11 processing, the time it takes for SlyFi using Tryst to setup a link is less than 10 ms more than that of wifi-open, which provides no authentication or confidentiality. Moreover, it is faster than wifi-wpa.[7]

Note that if a client did not know the particular wireless frequency a network was located on, it would spend more time in the probing phase because it would have to wait on each channel to see if a probe response arrives. This waiting time is configured to be 20–200 ms in 802.11.

**Background probing traffic.** The previous experiments assumed no ambient background traffic during the link setup process. However, due to the ad hoc nature of real wireless deployments, stations and APs often overhear messages that are not destined for them. For

[7]We note that wifi-wpa incurs an unnecessary delay in the open authentication phase, but since the bulk of the time is spent in wpa-key for key computation and exchange, removing this delay would not change the ranking of the total link setup times.

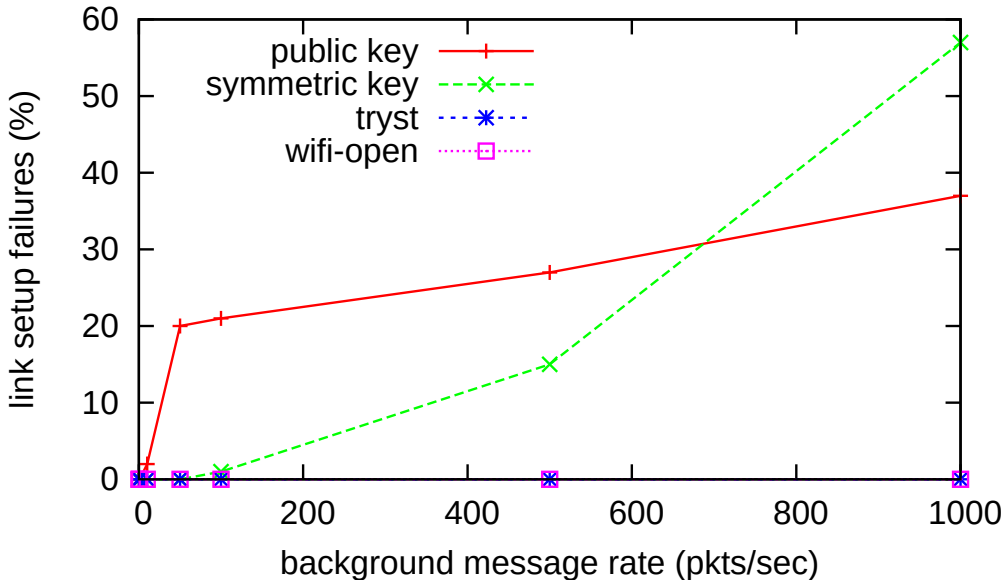


Figure 3.10: Percentage of 100 link setup attempts that fail to complete within 30 seconds as we vary the rate of background probe traffic not destined for the target AP. The client probes for one AP and the AP has 500 keys.

example, Figure 3.9 shows the rate of probe requests, responses, and authentication messages observed by one monitoring point.[8] Although the ambient message rate is generally fairly low, there are times when the rate is over 100 messages per second, due to many clients performing discovery at once. Thus, it is crucial that clients and APs be able to discard these messages quickly.

To evaluate how well SlyFi can manage background probe and authentication messages, we examine a client's link setup time as a function of such traffic. To do this, we introduce a third machine that sends background messages at a specified rate destined neither for the client or the AP. These background messages are encapsulated in the protocols we compare, but we precompute them so that their generation is able to maintain the specified rate. Each protocol queues up to 10 messages (drop tail) if it is busy processing and each client request is retransmitted once per second. We consider a link setup attempt to fail if it does not complete in 30 seconds. The client probes once for an AP with 500 keys.[9]

[8]The UCSD trace merged observations from multiple monitoring points, so it observes more traffic at any given time. The OSDI trace contains more users than the SIGCOMM trace and thus observed a higher rate of traffic.

[9]Note that in this experiment, the client and AP drivers ran in user level, rather than in the kernel, be-

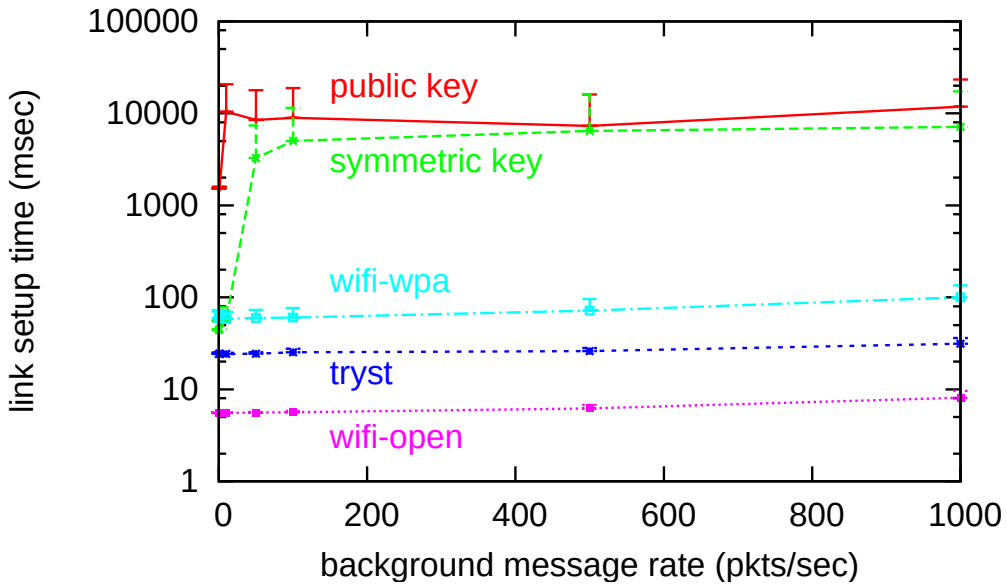


Figure 3.11: Link setup time for successful attempts as we vary the rate of background probe traffic not destined for the target AP. Error bars indicate one standard deviation.

Figure 3.10 shows the percentage of link setup attempts that failed. Due to the processing required by the public key and symmetric key protocols in order to determine whether a message is destined for the receiver, each begins to fail when the background message rate grows. No attempts fail when using Tryst or wifi-open. We omit the line for wifi-wpa because its message retry behavior is different. Figure 3.11 shows the link setup times for the attempts that succeeded. Note that while the symmetric key protocol is able to cope with message rates of up to 100 messages/second before it begins to fail, its link setup times grow to several seconds, and impact perceived performance, even when the rate is 50 messages/second.

Contention for the medium causes Tryst, wifi-open, and wifi-wpa to each have link setup times that grow slightly as the background message rate increases, but their scaling behavior is gradual and roughly consistent. Tryst's ability to discard background messages quickly enables it to scale gracefully. This property is important not only for dealing with ambient discovery traffic, but also for mitigating the impact of malicious denial of service attacks. With the public key and symmetric key protocols, a malicious device only needs

cause when the straw man protocols become overloaded with message processing, the Linux kernel became unresponsive to experimental commands. This imposes a slight overhead on message processing, but is insubstantial compared to each protocol's relative performance.

| # keys | 1 | 10 | 50 | 100 | 500 | 1000 | 10,000 |
|---|---|---|---|---|---|---|---|
| time (msec) | 0.08 | 0.49 | 2.3 | 4.7 | 24 | 47 | 800 |

Table 3.4: The mean time to update Tryst addresses for a single time interval as we vary the number of keys.

to send a small number of messages to prevent a client from setting up a link.

**Address update time.** At the beginning of each time interval, a Tryst node precomputes the message addresses it expects to receive to enable quick message filtering. Table 3.4 shows the time it takes to compute these addresses and update the hash table as we increase the number of keys a device holds. A node computes two addresses per time interval, per key (one for probes and one for authentication messages). Clients, which are unlikely to have more than 100 keys, would spend only a few milliseconds each time interval to update addresses, and time intervals would likely be at least several minutes. Even APs with 10,000 clients would spend less than 1 second.

### 3.7.4 Data Transport Performance

We now examine how well Shroud performs at delivering data packets. We begin with a description of micro-benchmarks that break down how long Shroud takes to send, filter, and receive packets. Then we present an analysis of packet delivery latency and throughput when a SlyFi client and AP are communicating in isolation. Finally, we look at performance in the face of background traffic, and present results describing achievable throughput both as the number of clients managed by the AP and the amount of competing traffic varies.

**Simulated hardware encryption.** Shroud's cryptography operations are implemented in software, which adversely affects performance. To understand how Shroud would perform with hardware support, we simulate the processing times of that hardware. As a result, we provide measurements both for the software-only version, shroud-sw, as well as for the version with hardware simulation, shroud-hw.

Since both wifi-wpa and Shroud use AES to encrypt and MAC packets, we use wifi-wpa's processing times as an estimate for shroud-hw.[10] We estimate wifi-wpa's cryp-

[10] wifi-wpa uses AES counter mode for payload encryption and AES-CBC for MAC computation, while Shroud uses AES-CBC mode for payload encryption and CMAC (a relative of AES-CBC mode) for the MAC. Counter mode is parallelizable, while AES-CBC is not.

| | send | | filter | | receive | | |
|---|---|---|---|---|---|---|---|
| | sw | hw | sw | hw | sw | hw | |
| update $addrs$ | | | | | | | |
| (max message loss) | 15 | 14 | NA | NA | 2047 | 2003 | (50) |
| (no message loss) | 15 | 14 | NA | NA | 119 | 117 | (1) |
| process $etext$ | 951 | 16 | NA | NA | 1541 | 16 | |
| process $emac$ | 740 | 16 | NA | NA | 740 | 16 | |
| Shroud total | 1821 | 120 | 32 | 32 | 3290 | 290 | |
| Click total | 1913 | 215 | 144 | 144 | 3402 | 407 | |

Table 3.5: Breakdown of processing times (see Section 3.3.4) for 1500 byte packets for shroud-sw (sw) and shroud-hw (hw). All times are in microseconds. Numbers in parentheses are numbers of address computations.

tographic processing time (including I/O) as the difference in round trip ping delays between wifi-wpa and wifi-open-driver. Measurements suggest the time to encrypt a 1500 byte ICMP packet is $\sim$16 usec and the time to encrypt a payload-free packet is $\sim$14 usec. I/O overhead dominates, but there is a small linear scaling factor as the packet size increases. Neither encryption nor MAC computation are parallelizable in Shroud, whereas CCMP's encryption may be parallelized in hardware. Thus, we conservatively estimate that shroud-hw would take 14 usec to encrypt a pair of addresses and 32 usec to encrypt and MAC packet payloads. To simulate these times, we modify our code to idle-wait for these times instead of computing the cryptographic operations in software. We note that shroud-hw still includes the actual software processing time of all non-AES operations.

**Micro-benchmarks.** Table 3.5 breaks down the time to send, filter, and receive Shroud messages. On a busy network, a packet received by a client is often intended for someone else, so filtering packets quickly is imperative. The filter column shows that shroud-hw's filtering time (32 usecs) is much faster than the theoretical minimum packet transmission time in 802.11a for a 1500 byte packet ($\sim$225 usecs), suggesting that a receiver could filter packets faster than the medium could supply them.

Sender-side processing of a 1500 byte packet, shown in the send column (215 usec), also edges out the time to transmit it, and thus, shroud-hw should be capable of supporting 802.11a's line speed. Receiver-side processing (the receive column) from the radio takes 407 usec, which is greater than the theoretical time to transmit, but still reasonable,

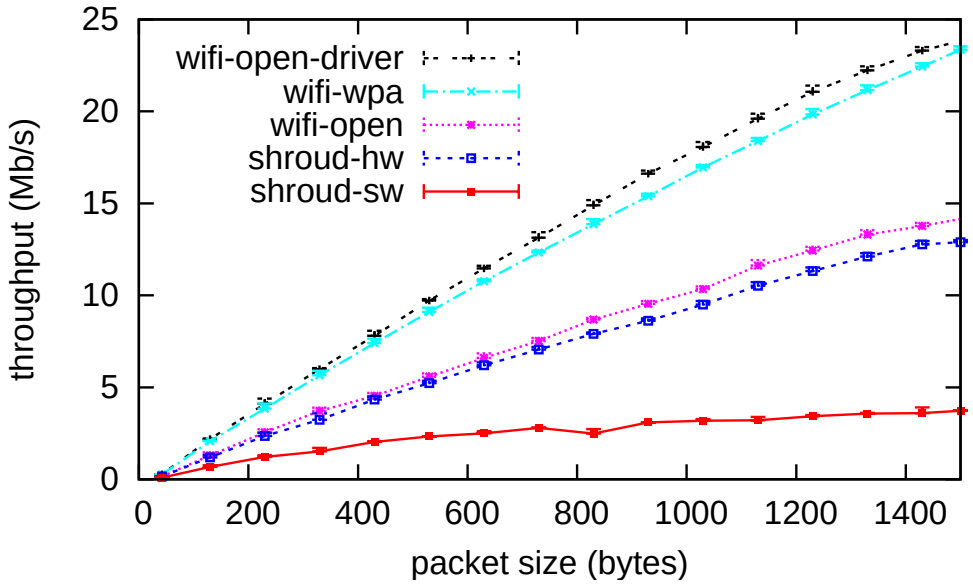


Figure 3.12: Throughput comparison of UDP packets when transmitting at 54 Mbps for 30 seconds. Each point is the average of 50 runs.

since 802.11 rates in practice are much slower (e.g., see Figure 3.12). When packets are lost, additional address computations must be performed after a reception. A reception following the maximum 49 packet burst loss (for $w = 50$) requires Shroud to compute and update 50 new addresses. This takes 2003 usec compared to 117 usec for a single address update (the case with no loss).

The cryptographic operations are much slower when implemented in software than they are in hardware, and thus the performance of shroud-sw is significantly below line speed. Regardless, we present these results to characterize our proof-of-concept implementation that can be used today to protect privacy. Obviously, an engineering effort is required to make use of hardware cryptography.

**Throughput and latency.** Figure 3.12 shows achievable throughput for shroud-hw, shroud-sw, wifi-open, and wifi-open-driver, measured using `iperf`. Shroud is implemented in Click, so wifi-open provides a baseline against which to evaluate its performance. While wifi-open (802.11 implemented in Click) performs worse than wifi-open-driver (the native driver implementation), the throughput degradation of shroud-hw is comparable to wifi-wpa relative to their respective baselines. Optimizing wifi-open is a

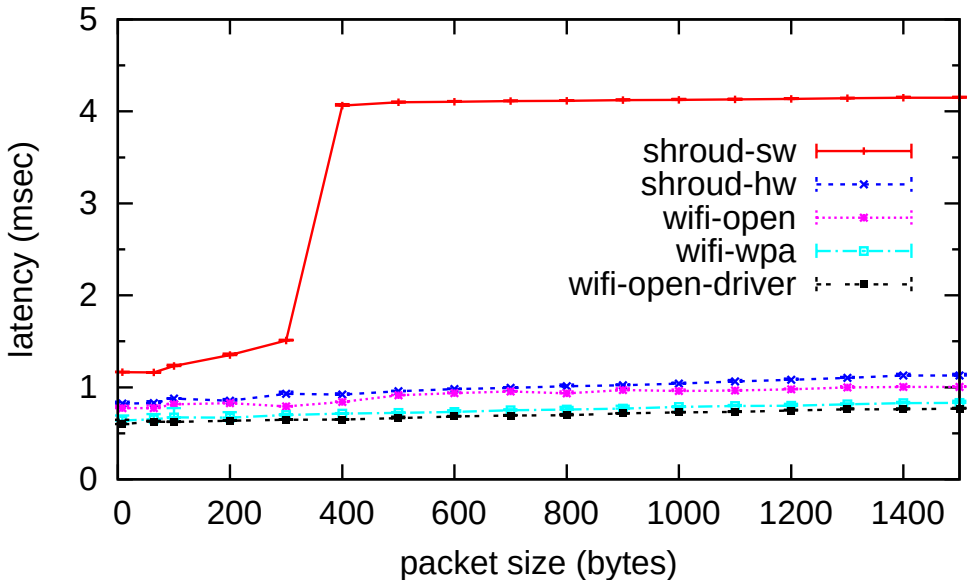


Figure 3.13: Comparison of round trip times of ICMP ping messages for variously sized packets. Each point is the average of 1000 pings; pings that experienced link-layer packet loss, or re-keying delays (in the case of wifi-wpa), were removed.

subject for future work. When sending 1500 byte packets, shroud-hw degrades wifi-open performance by only 1.44 Mbps compared to the 0.71 Mbps degradation from running wifi-wpa. Since both Shroud and wifi-wpa use some non-parallelizable cryptographic operations, the relative performance degradation increases with packet size. shroud-sw experiences a much larger drop in throughput, but still provides a functional link (3.73 Mbps).

Figure 3.13 presents round trip time measurements using `ping`. For 1500 byte packets, two packet payload encryptions and decryptions take ∼60 usec in wifi-wpa and ∼130 usec in shroud-hw; the extra time is due to address encryption. We believe the sudden marked increase in shroud-sw between 300 and 400 byte packets is due to an inefficiency in the Click runtime.[11]

**Background traffic.** Shroud's design is motivated by the requirement that background traffic must be filtered efficiently. To study how well Shroud filters packets, we run an experiment in which a client, $C_1$, sends packets as fast as possible to an access point, $AP_1$.

[11] Short packets get through the Click data path without a context switch from the OS, while longer packets do not.

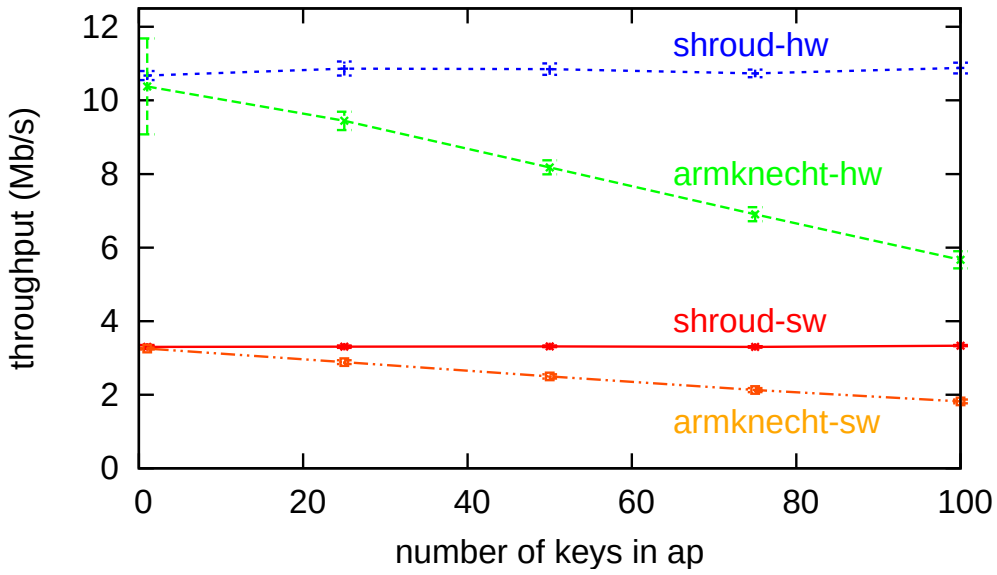


Figure 3.14: Effect of association set size on achievable throughput when exposed to 5 Mbps of background traffic for Shroud's and armknecht's software implementation and hardware simulation. Each run is 30 seconds; each point is the average of 50 runs.

Nearby, we generate background traffic by having another client, $C_2$ send traffic to another AP, $AP_2$. We measure the throughput at $AP_1$. Since the number of keys $AP_1$ manages (i.e., number of associations) and the amount of background traffic both affect throughput, we vary both independently.

Figure 3.14 shows throughput measured at $C_1$ for both the software implementation and hardware simulation of Shroud and armknecht, as the number of keys at $AP_1$ is varied. Since Shroud can filter background packets with just a hash table lookup, its achievable throughput is independent of the number of keys. However, armknecht's performance gets progressively worse as the number of keys increases. This is because clients and APs running armknecht must try every key they have before discarding background packets.

Figure 3.15, which shows throughput achieved as a function of the competing flow rate, depicts a similar effect. As the amount of background traffic increases, throughput decreases for both Shroud and armknecht, but considerably more so for armknecht. E.g., with 10 Mbps of background traffic, throughput is 31% lower for shroud-hw than it is with no competing traffic, but it is 72% lower for armknecht-hw. This reduction results from a combination of two effects: First, background traffic reduces the availability

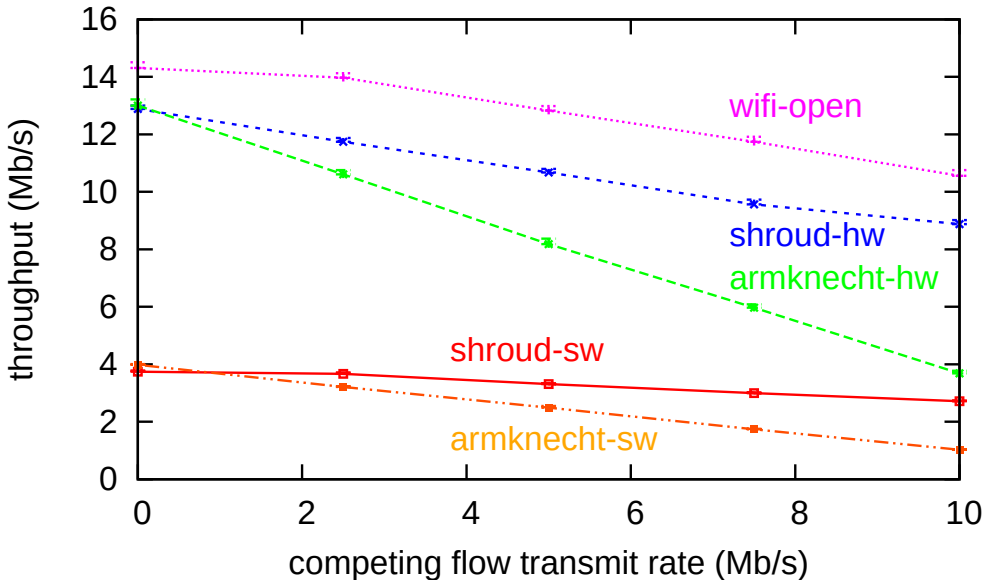


Figure 3.15: Effect of background traffic on achievable throughput from a client to an AP. The APs association set includes 50 keys. Each run is 30 seconds; each point is the average of 50 runs.

of the channel, as is evident in the throughput reduction (26%) of our baseline, wifi-open, which performs no cryptographic operations. This affects Shroud and armknecht similarly. Second, background traffic requires work to filter, which is much more expensive in armknecht.

## 3.8 Summary and Discussion

This chapter presented the design and evaluation of SlyFi, an identifier-free 802.11 link layer that obfuscates all transmitted bits, including addresses. The primary contribution of SlyFi is the development of two mechanisms, Tryst and Shroud, that can perform this obfuscation on existing link layer protocols without sacrificing efficiency or important protocol functions. For example, SlyFi can still perform efficient discovery, link establishment, and data transport, and higher layer name binding. Our evaluation showed that SlyFi performs comparably to WPA and performs substantially better than previously proposed techniques.

### 3.8.1 Discussion

We note that there are four important areas that merit further attention when SlyFi-like protocols are deployed more widely: private bootstrapping, large-scale broadcast discovery, wireless fault diagnosis, and side-channel concealment in practice.

**Bootstrapping shared keys remains a manual process.** In both existing secure wireless protocols and in SlyFi, an important challenge is how to bootstrap the symmetric or public keys that are necessary before two devices are able to rendezvous and authenticate each other for the first time. Existing bootstrapping techniques for key establishment typically fall into the category of *pairing* [152], i.e., having users of the devices use an out-of-band channel to exchange secrets. In many circumstances these techniques are either cumbersome and hinder usability or are inapplicable because no such out-of-band channel exists (e.g., if the user can not physically identify the other device). Therefore, it would be useful to develop automated key establishment techniques that do not require user intervention. We discuss some initial directions in this area in Chapter 5.

**SlyFi does not yet support efficient large-scale broadcast discovery.** We described in Section 3.4.1 how Tryst can use a single packet to discover multiple parties, but the size of this packet scales linearly with the number of parties. Atypical clients that want to discover the presence of hundreds or thousands of services confidentially will incur substantial message overhead with either protocol described above. To reduce this overhead in a presence sharing application, Cox *et al.* [46] propose sharing a single key with cliques of mutually trusting friends. However, this makes changing trust relationships difficult because it requires global agreement among the members of a clique. This is particularly difficult for mobile devices that are often offline.

One attractive option is a class of encryption protocols that enable a sender to broadcast a message that has size *sub-linear* in the number of authorized recipients, but can still only be decrypted by those recipients [55, 119]. Both public and symmetric key variants exist. These protocols are made possible by the additional assumption that no more than $m$ revoked receivers collude or no more than $r$ receivers are ever revoked. However, these protocols require key state and message overhead that are both super-linear in either $m$ or $r$, and therefore would only be more efficient when the number of services a client wants to discover is larger. Moreover, their current instantiations reveal the sender's identity and relationships (e.g., because they include the list of revoked devices). It is an open problem whether they can be made private [17].

Alternatively, one could reduce the number of services a client attempts to discover by ruling out services that are unlikely to be present based on context and location (e.g.,

using GPS). Nonetheless, it is important that using context does not inadvertently expose identity or relationships (e.g., discovery in different contexts should not noticeably change the number of discovery messages sent, lest message volume be used as a fingerprint).

**SlyFi may require novel fault diagnosis techniques.** By concealing all bits in transmitted messages, SlyFi potentially makes diagnosing faults in wireless networks more difficult because any eavesdropping device, including those used for network monitoring, requires the appropriate shared keys in order to interpret any packets on the network. For example, in some scenarios, bootstrapped keys may need to be entered manually into the devices by humans, which is an error-prone process. Distinguishing an erroneous entry from a broken network is more difficult with SlyFi because, from an eavesdropper's perspective, the pattern of packets observed is not easily distinguishable. There is a fundamental trade-off between the ease of diagnosing configuration errors using eavesdropping equipment and the amount of information that is concealed from malicious eavesdroppers. The appropriate trade-off to make will become more apparent when SlyFi-like protocols are used more often in practice.

**SlyFi does not always conceal all side-channels.** SlyFi mitigates the effectiveness of side-channels, such as packet sizes and timing, because different message streams overlap in practice, making it difficult to distinguish them. Initial work by Bauer *et al.* [18, 19] suggests that this traffic mixing does make some known side-channel attacks more difficult to carry out, even if physical layer information such as RSSI, is taken into account. Nonetheless, a more thorough analysis of how accurately eavesdroppers may still be able to detect users' fingerprints would shed light on whether additional countermeasures, such as cover traffic, are needed in practice. In addition, we still lack an empirical understanding of how often different traffic streams overlap in less dense environments, such as small hotspots — we only studied dense wireless environments such as conferences and office buildings. If there are not very many streams overlap in such environments or if they are easily distinguishable using physical layer information, then additional countermeasures against side-channel attacks will be required. Nonetheless, we note that *all* wireless environments are becoming denser due to the continued proliferation of wireless devices. Thus, in the near future, the qualitative aspects of mixing wireless traffic, rather than the quantity of wireless streams, may be more important to study.

# Chapter 4

# Mitigating Threats from Crowd-sourced Location-based Systems

SlyFi conceals identifiers from eavesdroppers during the processes of rendezvous and communication. However, users may still need to report location information and their identities to LBSes. In particular, to locate public Wi-Fi APs to use in the first place (e.g., when visiting a new area), users may have to query a hotspot directory service such as JiWire [85]. Hotspot APs, of course, are immobile and public, so they are unlikely to desire location privacy—indeed, most want to be found so users will pay to use them. However, the users that use them likely do.

Although the location masking techniques described in Section 1.3 can be used to obscure a user's location in queries to hotspot directories, many of these directories rely on user recorded reports in order to populate them (e.g., Hotspotr). Furthermore, users report that some APs block applications [174] and have poorer than advertised end-to-end performance [51], so selecting the best commercial AP is not always straightforward. Thus, it may be useful for these directories to incorporate user-submitted measurements on APs as a *crowd-sourced LBS*. This would enable users to evaluate commercial APs before paying for access.

In this chapter we answer two primary questions:

1. Are crowd-sourced hotspot directory services useful? That is, are such directory services even necessary?

2. How can we build them so that users' location privacy is respected, yet fraudulent reports are limited?

We address the first question by presenting the first measurement study of commercial APs in hotspot settings. Previous war-driving studies [71, 122] performed Wi-Fi measurements from streets or sidewalks, whereas we measure APs from the perspective of a typical Wi-Fi user who is inside an establishment. Our study examines the end-to-end performance and application support of all visible APs at 13 hotspot locations in Seattle over the course of 1 week. We find that there is indeed a wide range of AP performance even among APs very close to each other. Since there is currently no way for a user to determine which AP would be best to run his applications before paying for access, a crowd-sourced hotspot directory that collected performance measurements from users would indeed be useful for AP location and selection.

To address the second question, we present *Wifi-Reports*, a collaborative service that provides clients with historical information to improve AP selection. Wifi-Reports has two main uses: First, it provides users with a hotspot database similar to JiWire but where APs are annotated with performance information. Second, it enables users to more effectively select among APs visible at a particular location. Wireless clients that participate in Wifi-Reports automatically submit reports on the APs that they use. Reports include metrics such as estimated back-haul capacity, ports blocked, and connectivity failures. Using submitted reports, the service generates summary statistics for each AP to predict its end-to-end performance. Obtaining accurate user-submitted reports poses two challenges:

*(1) Location privacy:* As we have already suggested, a user should not have to reveal that he used an AP to report on it. Otherwise he would implicitly reveal a location that he visits. Users may be reluctant to participate in Wifi-Reports if their identities can be linked to their reports. At the same time, however, a few users should not be able to significantly skew an AP's summary statistics because some may have an incentive to submit fraudulent reports, e.g., to promote APs that they own. One way to meet these conflicting goals is to assume the existence of a trusted authority that is permitted to link users to their reports in order to detect fraud (e.g., in the way that eBay manages user reputations). For good reason, users, privacy groups, and governments are becoming increasingly wary about malicious or accidental disclosures of databases that can track large numbers of people [168]. Therefore, we present a report submission protocol that tolerates a few misbehaving users and does not require the disclosure of location related information to anyone, including the Wifi-Reports service. Our protocol leverages blind signatures to ensure that the service can regulate the number of reports that each user submits, but cannot distinguish one user's reports from another's.

*(2) Location context:* Physical obstructions and the distance between a client and an AP affect the quality of the wireless channel. Therefore, we must take location context into account when estimating AP performance or our estimates will not be accurate. We describe

how measurements can be categorized by the different wireless channel conditions under which they were performed. We also describe how to index and retrieve reports based on location without requiring additional equipment such as GPS.

We have implemented the key components of Wifi-Reports and used our measurement study to simulate how well it would work. Our results suggest that even if a user is only selecting among APs at a single location, Wifi-Reports performs close to optimal in more cases than existing techniques such as best-signal-strength and best-open-AP [122] because it provides information on commercial APs that cannot be tested beforehand. We show that Wifi-Reports' summary statistics predict performance accurately enough to make correct relative comparisons between different APs, despite performance variability due to competing traffic. For example, it predicts AP throughput and response time to within a factor of 2 at least 75% of the time. Since different APs' median throughputs and response times differ by up to $50\times$ and $10\times$, respectively, this prediction accuracy enables Wifi-Reports to select the best AP more often in more locations than any previous AP selection approach. Moreover, unlike previous AP selection approaches, Wifi-Reports enables users to examine the characteristics of APs that not in radio range, which is useful when users are mobile.

Although we presented our reporting protocol in the context of a hotspot directory service, it is applicable to crowd-sourced recommender systems more generally. Nonetheless, we note that there are two important limitations. First, the protocol's practicality is dependent on the ability to group items being voted on into subsets that are reasonably small and not revealing. Second, the protocol can not be applied when collaborative filtering is required. We discuss these limitations in greater detail in Section 4.7.

**Chapter outline.** Section 4.1 presents the results of our measurement study. Section 4.2 presents an overview of Wifi-Reports' design. Section 4.3 describes how it preserves privacy and mitigate fraud. Section 4.4 describes how it distinguish client locations. Section 4.5 presents an evaluation of Wifi-Reports. Section 4.6 describes protocols similar to Wifi-Reports' reporting protocol and Section 4.7 concludes this chapter.

## 4.1 Measurement Study

We conducted a measurement study to determine whether existing AP selection algorithms are sufficient to choose an AP that meets a user's needs. We sought answers to three questions that illustrate whether this choice is obvious and whether it can be improved with Wifi-Reports.

**Diversity.** Is there diversity in terms of end-to-end performance and application support of different hotspots' APs? The more diversity, the more likely a user will choose a hotspot with substantially suboptimal performance when selecting randomly from a hotspot directory.

**Rankability.** Is the best choice of AP at a particular location always obvious? If the best APs do not have any observable traits in common, then AP selection algorithms that use the same metric to rank APs at all locations will sometimes pick suboptimal APs.

**Predictability.** Is end-to-end performance predictable enough so that historical information would be useful?

Our study examined hotspots around University Avenue, Seattle, WA, near the University of Washington. We believe this area is representative of commercial districts with multiple Wi-Fi service providers. It is less likely to be representative of areas that only have a single Wi-Fi service provider, such as in many airports. However, since users don't have a choice of AP providers in those environments, selecting a provider to use is straightforward. Wifi-Reports could, however, still help a user decide if purchasing access is worthwhile. Figure 4.1 shows the hotspot locations where we performed measurements, which included those listed in JiWire's database and some additional sites known to us.

All locations are single-room coffee or tea shops. Most APs we measured are not open. In addition to each hotspot's official AP, the APs of hotspots nearby are also usually visible. APs of the free public seattlewifi network are sometimes visible at all locations. APs belonging to the University of Washington network are sometimes visible due to proximity to campus buildings, though these were never the best performing at any location. Our study offers a lower bound on the number and diversity of APs, as more may become available.

### 4.1.1 Setup

**Infrastructure.** To emulate a typical user of Wifi-Reports, we collected measurements with a commodity laptop with an Atheros 802.11b/g miniPCI card attached to the laptop's internal antennas. We implemented a custom wireless network manager for associating to APs and performing measurements after association. Our implementation is based on the Mark-and-Sweep war driving tool [71].

**Methodology.** During each measurement trial at a location, we emulate a typical connection attempt by scanning for visible APs. We then attempt to associate and authenticate

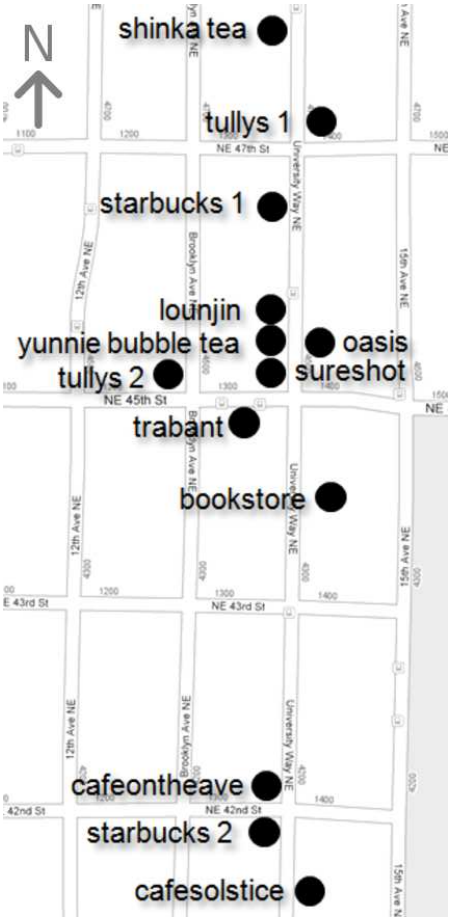


Figure 4.1: Measured hotspot locations near University Avenue, Seattle, WA

with each AP found (identified by its unique BSSID). If successful, we run our battery of measurement tests before moving on to the next AP. We manually obtain authentication credentials, if necessary (e.g., through a purchase). Since many Wi-Fi drivers do not list APs with low signal-to-noise (SNR) ratios, we only attempt to connect to APs when they appear with an SNR $> 10$ dB.[1]

We performed measurements at typical seating locations in each hotspot. Although the exact same location was not used for all measurements in a hotspot, Section 4.4 shows how well we can distinguish performance at different locations.

[1]One physical AP at each starbucks advertised two virtual APs. Since we did not find any service differentiation between these virtual APs after login, we only include one of them in our study. They exist because Starbucks hotspots are migrating from T-Mobile to AT&T Wi-Fi.

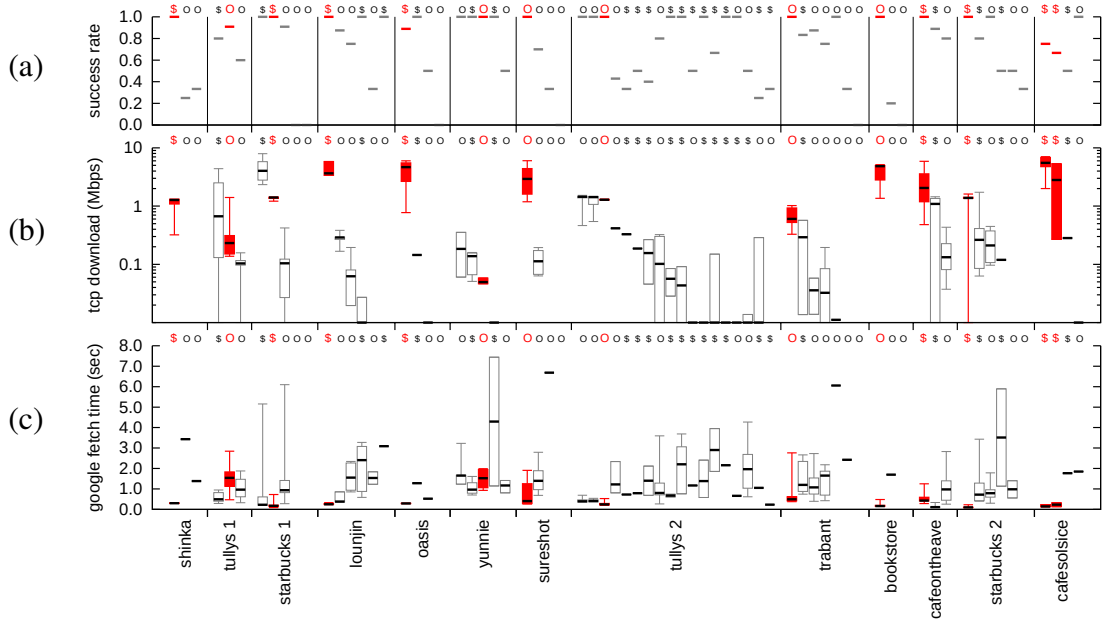


Figure 4.2: (a) The success rate of different APs (i.e., how often we could connect and access the Internet when each AP was visible). Each point represents one AP visible at each location. (b) A box-plot of the measured TCP download throughput through each APs. Note the logarithmic scale. (c) A box-plot of the time to fetch `http://www.google.com` using each AP. The measurements for each AP are grouped by the hotspot location where they were taken, shown on the x-axis. The symbol above each box indicates whether the AP can be accessed for free (O) or not ($). The box for the official AP at each hotspot is a solid color and its symbol is in a larger font. The APs in all graphs are sorted by their median TCP download throughput. Most of the non-free APs at tullys 2 are University of Washington APs in a building across the street.

**Time frame.** Previous studies measured each AP at a single point in time [71, 122]. Since we want to know whether AP characteristics are predictable, we performed 8 to 13 measurements at each location (with the exception of yunnie bubble tea, where we only performed 6 trials). These measurements were taken during 7 week days in October 2008. On each day, at each location, we performed 1-2 measurements at different times of the day, so we have at least one measurement during each 2 hour time-of-day between 9AM and 6PM (or a narrower time window if the hotspot opened later or closed earlier).

### 4.1.2 Results

**Basic connectivity.** Figure 4.2(a) shows the fraction of times we were able to obtain connectivity from each AP at each location (i.e., association and authentication succeeds, we are able to obtain a DHCP address, and able to fetch `www.google.com`; we retry each step up to 3 times and for up to 30 seconds on failure). We only count times when the AP was visible in a network scan. The symbol above each point indicates whether the AP can be accessed for free (O) or not ($). The box for the official AP at each hotspot is shown in a solid color and its symbol is in a larger font.[2]

As expected, most (10 of 13) official hotspot APs were successful 100% of the time. However, some, such as the ones at tullys 1 and cafesolstice, failed several times. These were all DHCP failures and frequent users of cafesolstice say that the AP has always had DHCP problems. However, it would be difficult to detect these problems automatically because even to attempt to access the network, a user has to obtain a WPA password from the cashier. Although unofficial APs visible at hotspots tend to fail with higher regularity due to wireless loss, a few in most (8 of 13) locations succeed whenever they were visible in our network scan. Thus, even this very basic connectivity metric suggests that there is diversity.

**TCP throughput.** Adequate throughput is important for many applications, such as streaming video or VoIP. Figure 4.2(b) shows a box-plot of the TCP download throughput achieved through each AP (i.e., the bar in the middle of each box indicates the median; the ends of each box indicate the first and third quantiles; and whiskers indicate the minimum and maximum). Note the logarithmic scale. We measured throughput over the final five seconds of a ten-second transfer from a high bandwidth server under our control to estimate each AP's sustained throughput after TCP slow start. We do not count the times when we failed to associate with the AP or when TCP timed out during establishment (the failure rate above suggests how often this occurs), so we have fewer measurements for some APs than for others.

First, we note that there is a significant range in available capacities across different hotspot locations. Median capacities range from less than 100 Kbps (yunnie) to over 5 Mbps (starbucks 1 and oasis). There is variability in each AP's throughput measurements, which is attributable mostly to wireless loss or contention (similar UDP throughput measurements had less variability), but the variation at most APs is much smaller than this wide performance range. Therefore, there is diversity in AP capacity, and throughput is

[2] cafesolstice has 2 official APs because it changed APs in the middle of our measurement period. However, both APs suffered from basic connectivity problems.

predictable enough to distinguish them.

Second, we observe that there is also a significant range in capacities among APs visible from a single location. As expected, most (9 of 13) official hotspot APs have the highest median throughputs at their respective locations. However, this is not true at tullys 1, yunnie, starbucks 1, and tullys 2, where better APs were available from an apartment building next door, the public seattlewifi network, a store next door, and a nearby hotel, respectively. Indeed, at starbucks 1 and yunnie, an unofficial AP always gave significantly more throughput than the official one when visible. Recall that these comparisons only include measurements when we were able to successfully pay for and obtain Internet connectivity, so a user without historical information would have to pay before discovering this.

**Response time.** Low network latency is another important attribute for interactive applications such as web browsing. To estimate the latency a typical web browser would experience, we measured the response time to one of the most popular web sites. Figure 4.2(c) shows a box-plot of the time to fetch `http://www.google.com`. Fetch time includes the time to perform a DNS lookup, which is dependent on the DNS server each AP's DHCP server assigns us.[3] Since Google's homepage is only 6KB, fetch time is dominated by latency rather than transfer time. We do not count the times when association failed.

Just as we saw with TCP throughput, there is diversity in response time, which ranges from less than 100 ms to several seconds. Response times of more than 1 second are typically noticeable by an end-user. As expected, most (10 of 13) official APs have the lowest median latency at their respective hotspot locations, but this is not true at tullys 1, yunnie, and cafeontheave. Only the disparity between the best and official APs at tullys 1 is large enough to be noticeable, but even smaller differences may impact more sensitive applications, such as VoIP. In addition, in some cases the AP with the lowest and least variable response time is not the same as the AP with the highest throughput (e.g., at starbucks 1), so ranking is dependent on application requirements. Finally, all official APs, except the one at sureshot, provide predictable response times (first and third quantiles within a factor of 2). At least one unofficial AP at each location is just as predictable.

**Port blocking.** To determine if an AP blocked or redirected certain application ports, we sent 3 probes to each port on a measurement server under our control. For UDP ports, each probe consisted of 44-byte request and response datagrams, while for TCP ports,

[3]The CNAME and A DNS records for `www.google.com` have a TTLs of 1 week and 1 day, respectively, so they are almost always already cached at the DNS server.

each probe tried to establish a connection and download ∼32 bytes of data (in order to check for port redirection). We tested common application ports including: FTP, NTP, SSH, NetBIOS, SMTP, IMAP, SSL, VoIP (SIP), STUN, common VPN ports, World of Warcraft, Counterstrike, Gnutella, and Bittorrent. To account for packet loss, we conclude that a port is blocked only if it was never reachable in any of our measurements.

All APs blocked NetBIOS, most likely because they are configured to do so by default. Three APs blocked non-DNS packets on port 53 and only one (bookstore's official AP) blocked more ports: all non-privileged TCP ports and all UDP ports except DNS and NTP. Nonetheless, this is useful information, as common applications such as VPNs, VoIP, and games would not function.

**Summary.** With respect to *diversity*, we find that there is significant diversity in AP throughput and latency. With respect to *rankability*, the official AP is not the best choice at 30% of hotspot locations, so ranking APs is not always obvious. Finally, with respect to *predictability*, there is variability in performance over time, but this variability is much smaller than the range of different APs' performance, so historical information should be predictable enough to compare APs. Therefore, our results suggest that a collaborative reporting service may improve AP selection.

### 4.1.3 Discussion

**Why not just use official APs?** One might ask whether historical information is really necessary if the official AP is usually the best at 70% of locations. First, in Section 4.5.1, we show that historical information can get us the best AP in the remaining 30%. Second, as hotspot density increases, scenarios like these will likely become more common. Third, many users will be willing to move to find better APs and, without historical information, it is not obvious how to determine where to move to. Finally, if a user is not in range of any APs, he needs historical information to determine where to find a good one.

**Other selection factors.** In practice, users will likely take other factors into account besides AP performance and application support, such as cost and venue. Although these factors are important and reports in Wifi-Reports can include such information, they are also subjective, so we focus our evaluation on AP performance. In particular, we focus on download capacity and latency since these metrics are important for most applications. Our focus demonstrates Wifi-Reports' ability to help users make more informed decisions about which APs to use, whether they take cost and venue into account or not.

## 4.2 Wifi-Reports Overview

Wifi-Reports is a recommendation system [5]. Users rate the services they use and submit these ratings to a report database where they are summarized. Other users download summarized ratings to evaluate services that they are considering. In Wifi-Reports, the users are wireless clients, services are APs, and ratings are key-value pairs of measured performance metrics.

### 4.2.1 Challenges

In contrast to previous recommendation systems, Wifi-Reports faces two unique challenges:

**Location privacy.** By reporting the use of an AP, a user implicitly reveals a location where he has been with an accuracy that is sufficient to identify sensitive places [129]. Thus, users may not be willing to participate in Wifi-Reports if their identities can be linked to their reports. A single user's reports must not even be linkable to each other, otherwise they are vulnerable to inference attacks [27, 64]. Nevertheless, we still want to limit the influence of malicious users that submit fraudulent reports, which is a common problem in recommendation systems [162, 176].

**Location context.** Clients will typically search for summaries by location (e.g., "all APs in Seattle"), and the location of a client with respect to an AP will affect its measured performance due to different wireless channel conditions. Since we would like clients to generate reports automatically, location context must be determined automatically.

### 4.2.2 System Tasks

The operation of Wifi-Reports consists of three main tasks (Figure 4.3). We present an overview of these tasks here. The next two sections describe how they can be done while addressing the challenges discussed above.

**Measure and report.** Clients measure and submit reports on APs that they use. For example, suppose a client attempts to connect to the Internet using AP $X$. If the connection fails (i.e., association, DHCP, or all TCP connections fail), the client generates the report {ap=**X**, SNR=20dB, date=11/14/2008, connectivity=false}.[4] If the connection succeeds,

[4] **X** refers to the AP's BSSID and a hash of its signing key described in Section 4.3.

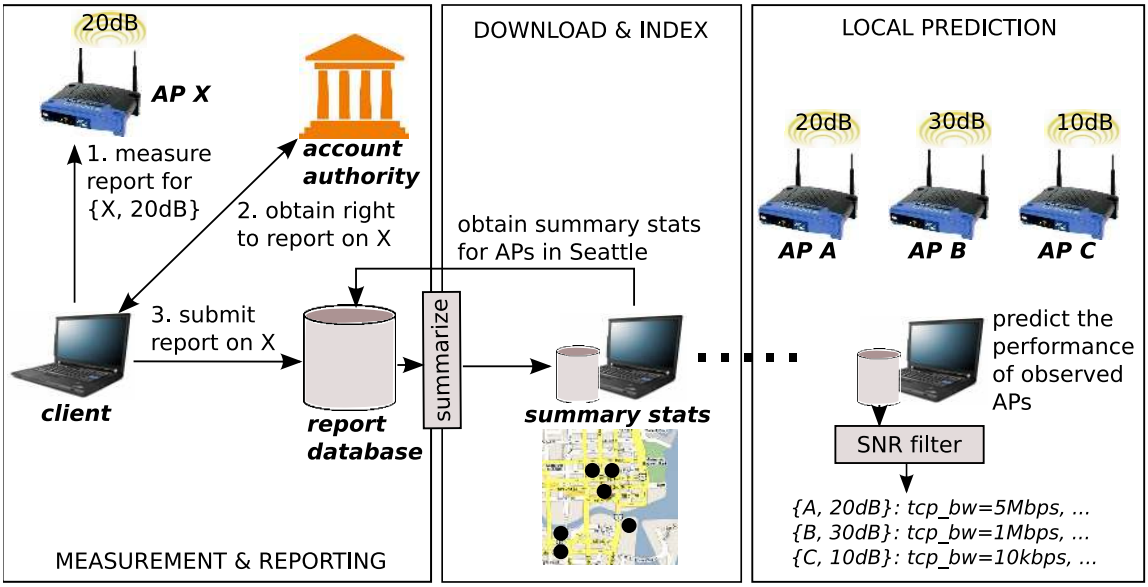


Figure 4.3: Wifi-Reports components and typical tasks.

then the client software estimates performance metrics based on the user's network traffic or using active measurements when the connection is idle.[5] When measurement completes, it generates the report {ap=X, SNR=20dB, date=11/14/2008, connectivity=true, tcp_bw_down=100kbps, google_resp_time=500ms, ...}.

When the client has Internet connectivity again, it contacts an *account authority* to obtain the right to report on $X$, e.g., by receiving a credential. It sends this report along with the credential to a *report database*. An account authority is necessary to prevent a single malicious client from submitting an unbounded number of fraudulent reports. However, to preserve the location privacy of honest clients, neither the account authority nor the report database should learn that the client used AP $X$. We describe the novel protocol we use to address this problem in Section 4.3.

**Download and index.** The database generates summary statistics for each AP by summarizing the values for each key. To be robust against some fraudulent values, we use summary functions that are not significantly skewed by a small fraction of outliers. For example, median is used for real-value attributes (e.g., throughput), plurality voting for multinomial attributes (e.g., port blocking), and average for probability attributes with

[5] A number of techniques and tools exist to estimate bandwidth [134] and response time [79]. These techniques are outside the scope of this dissertation, but the measurements we used can be implemented as an anonymous speed test.

$\{0, 1\}$ inputs (e.g., basic connectivity). In addition, a summary indicates the number of reports that it summarizes as an estimate of its robustness (i.e., a user will pay more heed to a summary of 10 different reports than a summary of just 1 report). A client may choose to ignore summaries with too few reports to mitigate the impact of erroneous reports by early adopters.

Before traveling, a user downloads and caches the summary statistics of all APs in the cities that he expects to visit. In practice, client software would update this cache whenever it has connectivity, similar to the iPass [82] client. To find a suitable hotspot, reports are shown to a user on a map. In order to facilitate this operation, reports must be search-able by geographic location. Unfortunately, we cannot rely on GPS because many wireless clients are not equipped with it and it is often does not work indoors. We describe existing techniques that we leverage to obtain coarse geographic coordinates in Section 4.4.1.

**Predict locally.** Finally, when a user sits down at a cafe, he typically wants to find the best AP that is visible. Although the client will have downloaded summaries for these APs earlier, the expected performance of each AP depends on the wireless channel conditions between the client and the AP. For example, conditions will vary based on the observed signal-to-noise ratio (SNR). Therefore, the client must apply a filter to the summaries to obtain an accurate prediction for the current conditions. We describe how a client can perform this filtering in Section 4.4.2.

## 4.3 Location Privacy

This section describes a novel report submission protocol that ensures *location privacy* and *limited influence*, properties that we define below. Define $U$ to be the set of all users that participate in Wifi-Reports, $S$ to be the current set of all APs, $u = \mathsf{submitter}(R)$ to be the user that submitted report $R$, and $s = \mathsf{target}(R)$ be the AP that $R$ reports on. Suppose $C \subset U$ is the largest set of colluding malicious users that try to violate any user's location privacy or to influence an AP's summary.

**Location privacy.** To preserve location privacy, we must satisfy three conditions. (1) No one, not even the account authority or report database, should be able to link any report to its submitter; i.e., no one should be able to guess $\mathsf{submitter}(R_i)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all reports $R_i$. (2) No one should be able link any two reports together unless they were submitted by the same user for the same AP; i.e., no one should be able to guess whether $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all $R_i, R_j$ where $\mathsf{submitter}(R_i) \neq \mathsf{submitter}(R_j)$ or $\mathsf{target}(R_i) \neq \mathsf{target}(R_j)$. (3) A user

should not have to reveal the target of a report in order to obtain the right to submit the report; i.e., after obtaining the right to submit $R_{k+1}$, the account authority should not be able to guess $\mathsf{target}(R_{k+1})$ with probability greater than $\frac{1}{|S|}$. In practice, achieving this third condition may be too expensive, so we later relax it by restricting $S$ to all APs in a city rather than all APs.

**Limited influence.** To limit the influence of dishonest users, exactly one report from each user who has submitted a report on AP $s$ should be used to compute the summary statistics for $s$. To ensure that this condition is satisfied, any two reports submitted by the same user for the same AP must be linked; i.e., for all $R_i, R_j$ where $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$ and $\mathsf{target}(R_i) = \mathsf{target}(R_j)$, anyone *should be able* to verify that $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$. When computing each summary, the database first summarizes each individual user's reports and then computes a summary over these summaries. This ensures that malicious users have at most $|C|$ votes on the final summary.

We may also want to limit the rate at which these users can submit reports on any AP. For example, we may want to prevent a malicious user from reporting on a large number of APs that he has never actually visited. We discuss how to achieve this additional property at the end of Section 4.3.3.

### 4.3.1 Threat Model

Users' location privacy should be protected from malicious users, the account authority, and report databases. To meet this goal, we don't assume any restrictions on the behavior of malicious users, but we make a few practical assumptions about the account authority and report databases.

**Account authority.** A challenge for recommendation systems is how to prevent malicious users from out-voting honest users, e.g., by using botnets or Sybil attacks to obtain many fake identities. Wifi-Reports, as with most existing recommendation systems, assumes that a central account authority can limit these large-scale attacks. For example, the authority can require a credential that is hard to forge, such as a token credit card payment or the reception of an SMS message on a real cell phone. These defenses are not perfect, but are enough of a deterrent that existing recommender systems work well in practice. These heuristics may also be supplemented by Sybil detection schemes (e.g., [175]). Thus, we assume that these mechanisms are sufficient to bound the number of malicious users to a small fraction of the total number of users. Section 4.5.3 shows that our system can limit the influence of this small number of malicious users. We assume that the account authority is honest but curious; that is, it may try to reveal information about users, but

it does not violate our protocol. We discuss how selfish violations can be detected in the next two sections. Since the account authority is a high profile entity, we believe that the legal implications of violations are sufficient deterrents to prevent them.

**Report databases.** Users have to trust the report database to summarize reports correctly. To distribute this trust, we assume that there are multiple databases and that most are honest (e.g., do not delete reports prematurely). Honest users submit reports to all the databases and download summary statistics from all databases, using the report on each AP that the majority of databases agree upon. We note that the existence of a single honest database can be used to audit all databases, because any valid report that exists should exist on all the databases, and reports are independently verifiable (see the protocol below). Independent verifiability also means that each database can periodically check the others to discover and obtain reports that it is missing. We assume that users learn about the list of report databases in an out-of-band manner; e.g., it may be distributed with the software.

A report database can link reports if they are submitted from the same IP address. Therefore, we assume that users submit reports through a mix network such as Tor [49] and that the mix achieves its goal, i.e., no one can infer the source IP address of the sender's messages.

### 4.3.2 Straw Man Protocols

**Anonymize reports.** One approach might be to have users simply submit reports to the databases via a mix network. This means that all reports are unlinkable, thus providing location privacy. However, this protocol does not provide limited influence because a database can not distinguish when one user submits many reports on an AP versus when many users submit one report each on the AP.

**Authenticate reports.** For this reason, nearly all existing recommender systems today rely on a trusted central authority that limits each real user to a single account. We can limit influence with an authority $A$ as follows: When a user $u_i$ wants to submit a report $R$ on AP $s_j$, it authenticates itself to $A$ (e.g., with a username/password) and then sends $R$ to $A$. $A$ checks if $u_i$ has previously submitted any reports on $s_j$ and, if so, deletes them from the report databases before adding the new one. $A$ explicitly remembers the user that submitted each report. If $A$ is the only one allowed to add and remove reports from the report databases, this protocol provides limited influence because each user is limited to one report. However, it fails to provide location privacy with respect to $A$. Indeed, $A$ *must* remember which reports each user submitted to prevent multiples.

### 4.3.3 Blind Signature Report Protocol

To achieve both location privacy and limited influence, Wifi-Reports uses a two phase protocol. We sketch this protocol here: First, when user $u_i$ joins Wifi-Reports, the account authority $A$ provides him with a distinct signed "token" $K_{ij}$ for each AP $s_j \in S$. By using a blind signature [21], no one, including $A$, can link $K_{ij}$ to the user or to any other $K_{ij'}$. This ensures location privacy. However, anyone can verify that $A$ signed $K_{ij}$ and that it can only be used for $s_j$. GENTOKEN describes this step in detail below. Second, to submit a report $R$ on AP $s_j$, $u_i$ uses the token $K_{ij}$ to sign $R$, which proves that it is a valid report for $s_j$. $u_i$ publishes $R$ to each report database anonymously via the mix network. Since $u_i$ only has one token for $s_j$, all valid reports that $u_i$ submits on $s_j$ will be linked by $K_{ij}$. This ensures limited influence. SUBMITREPORT describes this step in detail below.

**Preliminaries.** The RSA blind signature scheme [21] is a well known cryptographic primitive that we use in our protocol. Let $\mathsf{blind}(K, m, r)$ and $\mathsf{unblind}(K, m, r)$ be the RSA blinding and unblinding functions using RSA public key $K$, message $m$, and random blinding factor $r$ (we use 1024-bit keys and values). Let $\mathsf{sign}(K^{-1}, m)$ be the RSA signature function using RSA private key $K^{-1}$, and let $\mathsf{verify}(K, m, x)$ be the RSA verification function, which accepts the signature $x$ if and only if $x = \mathsf{sign}(K^{-1}, m)$. Let $H(m)$ be a public pseudorandom hash function (we use SHA-512). We leverage the following equivalence:

$$\mathsf{sign}(K^{-1}, m) = \mathsf{unblind}(K, \mathsf{sign}(K^{-1}, \mathsf{blind}(K, m, r)), r)$$

That is, blinding a message, signing it, and then unblinding it results in the signature of the original message.

Blind signatures have two important properties. (1) *Blindness*: without knowledge of $r$, $\bar{m} = \mathsf{blind}(K, m, r)$ does not reveal any information about $m$. (2) *Unforgeability*: suppose we are given valid signatures $(x_1, x_2, \ldots, x_k)$ for each of $(m_1, m_2, \ldots, m_k)$, respectively, where $m_i = H(\hat{m}_i)$. Without the secret key $K^{-1}$, it is infeasible to forge a new signature $x_{k+1} = \mathsf{sign}(K^{-1}, H(\hat{m}_{k+1}))$ for any $\hat{m}_{k+1} \neq \hat{m}_i$ for all $i$, under the assumption that the known-target or chosen-target RSA-inversion problems are hard [21]. However, anyone can check whether $\mathsf{verify}(K, H(\hat{m}_i), x_i)$ accepts.

**Protocol description.** Our protocol has two phases: GENTOKEN and SUBMITREPORT, described below. For now, assume that the set of APs $S$ is fixed and public knowledge. We describe later how APs enter and leave this set.

**GENTOKEN$(u_i, s_j)$.** The GENTOKEN phase is used by user $u_i$ to obtain a token to report on AP $s_j$ and $u_i$ only performs it once per $s_j$ in $u_i$'s lifetime. $s_j$ identifies an

AP by BSSID as well as a hash of $A$'s signing key for that AP (see below), i.e., $s_j = \{bssid_j, H(bssid_j|M_j)\}$. We assume that $u_i$ and $A$ mutually authenticate before engaging in the following protocol (e.g., with SSL and a secret passphrase).

$$
\begin{aligned}
A: &\quad \{M, M^{-1}\}, \{M_j, M_j^{-1}\}\ \forall s_j \in S, \\
&\quad msig_j \leftarrow \mathsf{sign}(M^{-1}, H(bssid_j|M_j))\ \forall s_j \in S \\
u_i: &\quad M, M_j, msig_j, \{K_{ij}, K_{ij}^{-1}\}, r \xleftarrow{R} \{0,1\}^{1024} \\
u_i: &\quad b \leftarrow \mathsf{blind}(M_j, H(K_{ij}), r) && (4.1) \\
u_i \rightarrow A: &\quad \texttt{"sig-request"}, s_j, b && (4.2) \\
A: &\quad sig'_{ij} \leftarrow \mathsf{sign}(M_j^{-1}, b) && (4.3) \\
A \rightarrow u_i: &\quad \texttt{"sig-reply"}, sig'_{ij} && (4.4) \\
u_i: &\quad sig_{ij} \leftarrow \mathsf{unblind}(M_j, sig'_{ij}, r) && (4.5)
\end{aligned}
$$

The lines before step 4.1 show items that are obtained before the protocol begins. $A$ has a single *master* RSA key pair $M, M^{-1}$ and has generated a different *signing* RSA key pair $M_j, M_j^{-1}$ for each $s_j$. $H(bssid_j|M_j)$ is signed by the authority's master key so that others can identify $M_j$ as a signing key for $bssid_j$. $M$, $M_j$, and $msig_j$ are publicly known (e.g., given to users and databases by $A$ when they join). $u_i$ generates a new *reporting* key pair $K_{ij}, K_{ij}^{-1}$ and a 1024-bit random value $r$. After step 4.2, $A$ checks whether it has already sent a `sig-reply` message to $u_i$ for $s_j$. If so, it aborts, otherwise it continues. After step 4.5, $u_i$ checks that $\mathsf{verify}(M_j, H(K_{ij}), sig_{ij})$ accepts. At completion, $u_i$ saves $K_{ij}$, $K_{ij}^{-1}$, and $sig_{ij}$ for future use.

This exchange can be described as follows: $A$ authorizes the reporting key $K_{ij}$ for use on reports for $s_j$ by blindly signing it with $s_j$'s signing key $M_j^{-1}$. By blindness, $A$ does not learn $K_{ij}$, only that the client now has a key for $s_j$. Thus, no one can link $K_{ij}$ to user $u_i$ or to any $K_{il}, l \neq j$. $\{K_{ij}, sig_{ij}\}$ is the token that $u_i$ attaches to reports on $s_j$. When a report is signed with $K_{ij}^{-1}$, this token proves that the report is signed with an authorized signing key. Since $A$ only allows each user to perform GENTOKEN once per AP, each user can only obtain one authorized reporting key for $s_j$. By unforgeability, even if multiple users collude, they cannot forge a new authorized reporting key.

**SUBMITREPORT**$(u_i, s_j, R)$**.** This phase is used by user $u_i$ to submit a report $R$ on AP $s_j$ after a token for $s_j$ is obtained. Let $\{D_1, \ldots, D_m\}$ be the $m$ independent databases. $R$ is submitted to each $D_k$ as follows.

$$
\begin{aligned}
D_k: &\quad M, M_j\ \forall s_j \in S \\
u_i: &\quad rsig \leftarrow \mathsf{sign}(K_{ij}^{-1}, H(R)) && (4.6) \\
u_i \rightarrow D_k: &\quad \texttt{"report"}, s_j, K_{ij}, sig_{ij}, R, rsig && (4.7)
\end{aligned}
$$

The message in step 4.7 is sent through a mix network so it does not explicitly reveal its sender. After step 4.7, $D_k$ checks that $\mathsf{verify}(M_j, H(K_{ij}), sig_{ij})$ and $\mathsf{verify}(K_{ij}, H(R), rsig)$ both accept. If any of these checks fail, the report is invalid and is discarded. In other words, $u_i$ anonymously publishes a report $R$ signed using $K_{ij}^{-1}$. By including $\{K_{ij}, sig_{ij}\}$, anyone can verify that the signature is generated using a key signed by $M_j^{-1}$, i.e., a key that $A$ authorized to report on $s_j$ during the GENTOKEN phase.

**Anonymizing GENTOKEN.** This protocol achieves limited influence and prevents each report from being linked to any user or any other report. However, if a user engages in GENTOKEN$(u_i, s_j)$ only when it reports on $s_j$, then it reveals to $A$ that it is reporting on $s_j$. In order to satisfy the third condition of our location privacy requirement, that $A$ cannot guess the AP with probability greater than $\frac{1}{|S|}$, $u_i$ would have to perform GENTOKEN on all $s \in S$ before submitting any reports so that $A$ cannot infer which tokens were used.

When performing GENTOKEN on all APs is too expensive, we relax this condition as follows. We allow $A$ to infer that the AP is in a smaller set $\hat{S} \subset S$. Determining an appropriate set $\hat{S}$ is a trade-off between more location privacy and less time spent performing GENTOKEN operations. We have users explicitly choose a region granularity they are willing to expose (e.g., a city). When reporting on an AP, they perform GENTOKEN on all APs in this region. We believe this small compromise in location privacy is acceptable since users already volunteer coarse-grained location information to online services (e.g., to get localized news) and IP addresses themselves reveal as much. In Section 4.5, we show that using the granularity of a city is practical.[6]

**Handling AP churn.** To support changes in the set of APs $S$, $A$ maintains $S$ as a dynamic list of APs. Any user can request that $A$ add an AP identified by BSSID and located via beacon fingerprint (see Section 4.4.1). $A$ generates a new signing key pair and its signature $\{M_j, M_j^{-1}\}, msig_j \leftarrow \mathsf{sign}(M^{-1}, H(bssid_j|M_j))$, and the new AP is identified by $s_j = \{bssid_j, H(bssid_j|M_j)\}$. $M_j$ and $msig_j$ are given to the user and he submits them along with the first report on $s_j$ to each report database. AP addition is not anonymous, as the user must reveal the AP to $A$, so Wifi-Reports will initially depend on existing hotspot and war driving databases and altruistic users to populate $S$. However, over time we believe that owners of well-performing APs will be incentivized to add themselves because otherwise they will not receive any reports. An AP is removed from $S$ if it is not reported on in 3 months (the report TTL, see below) and $A$ sends a revocation of their signing keys to each database. Users can thus obtain new signing public keys and

[6] An alternative solution is to perform GENTOKEN on a random subset of $n$ APs in addition to the target AP. However, since a user will likely submit reports on multiple correlated APs (e.g., APs in the same city), $A$ can exploit correlations to infer the APs actually reported on.

revocations from each database.

We take three steps to limit the impact of nonexistent or mislocated APs that malicious users may add. (1) When searching for APs on a map, the client report cache filters out APs that only have a small number of recent reports; these APs require more "locals" to report on them before distant users can find them. (2) After a sufficient number of reports are submitted, reported locations are only considered if a sufficient number are near each other, and the centroid of those locations is used. (3) $A$ rate limits the number of additions each user can make.

**Handling long-term changes.** AP performance can change over time due to back-haul and AP upgrades. However, these changes typically occur at timescales of months or more. Thus, reports have a time-to-live (TTL) of 3 months. Databases discard them afterward. Determining the most appropriate TTL is a trade-off between report density and staleness and is a subject of future work.

**Handling multiple reports.** Our protocol allows $u_i$ to submit multiple reports on $s_j$, which can be useful if they are from different vantage points or reflect changes over time; however, each report on $s_j$ will be linked by the key $K_{ij}$. To ensure limited influence, a database needs to summarize each user's reports on $s_j$ before computing a summary over these individual summaries. For simplicity, it computes an individual user's summary as just the most recent report from that user that was taken in the same channel conditions (see Section 4.4.2).[7] As a consequence, there is no need for an honest user to submit a new report on $s_j$ unless the last one it submitted expired or if $s_j$'s performance substantially changed. This approach also allows a client to mitigate timing side-channels (discussed below) by randomly dating his reports between now and the date in his last report on $s_j$ without changing $s_j$'s summary statistics.[8]

**Rate limiting reports.** As mentioned earlier, it may also be desirable to limit the rate at which an individual user can submit reports, say, to at most $t$ reports per week. This can be accomplished with a straight forward extension of the SUBMITREPORT stage of the protocol: $A$ keeps count of the number of reports that each user submits this week. Before submission of $report = \{s_j, K_{ij}, sig_{ij}, R, rsig\}$ (step 4.7), user $u_i$ sends $h =$

[7] A more sophisticated summarization algorithm might use the mean or median values of all a user's reports, weighted by report age. We leave the comparison of summary functions to future work as we do not yet know how many reports real users would submit on each AP.

[8] If the owner of $K_{ij}$ is ever exposed, then an adversary learns some approximate times when $u_i$ used $s_j$. If $u_i$ knows this, he can prevent any further disclosures by proving to $A$ that he revoked $K_{ij}$ and obtaining a new token for $s_j$ using GENTOKEN; i.e., $u_i$ can send $\{\texttt{"revoke"}, u_i, K_{ij}, ksig\}$ to $A$ and the databases, where $ksig \leftarrow \mathsf{sign}(K_{ij}^{-1}, H(\texttt{"revoke"}|u_i|K_{ij}))$, which proves that $u_i$ has $K_{ij}$'s secret key and that $K_{ij}$ (and all reports signed with it) is revoked.

$\mathsf{blind}(M, H(report), r)$ to $A$. If $u_i$ has not already exceeded $t$ reports this week, $A$ sends $lsig' = \mathsf{sign}(M^{-1}, h)$ back to $u_i$, and $u_i$ unblinds $lsig'$ to obtain $lsig = \mathsf{sign}(M^{-1}, H(report))$. $lsig$ is included in the report submitted to the report databases and is verified to be correct by recipients. The user would submit the report to the database at a random time after obtaining $lsig$, so $A$ would only be able to infer that it was requested by some user in the recent past, but not which one.

10-20 would be reasonable values for $t$; analysis of Wi-Fi probes shows most clients have not used more than 20 APs recently [63]. This approach only adds 4 ms of computational overhead on $A$ per report submitted (see Section 4.5.2).

### 4.3.4 Discussion

**BSSID spoofing.** One obvious concern is that some APs can change their BSSID identities. For example, a poorly performing AP might spoof the BSSID of a good AP to hijack its reputation. Ideally, each AP would have a public key pair to sign its beacons. APs could then be identified by the public key instead of BSSID to prevent spoofing. In 802.11, APs can offer this signature and its public key as part of a vendor-specific information element or as part of 802.1X authentication. Without public key identities, we can still mitigate spoofing with two techniques: First, if an AP masquerades as another AP that is geographically far away, then reports on each will be summarized separately as distinct APs and users will treat them as such. Second, if an AP attempts to spoof one that is nearby, the distribution of beacon SNRs that users receive will likely have two distinct modes. This at least enables users (and the original AP) to detect spoofing, though resolution requires action in the "real world" since the 802.11 protocol cannot distinguish the two APs. Finally, laws against device fraud (e.g., [57]) may be a sufficient deterrent in practice.

**Eclipse attacks.** If $A$ only reveals $s_j$ to a single user $u_i$, $A$ will know that any report for $s_j$ is submitted by $u_i$. Therefore, $u_i$'s view of the set of APs $S$ is obtained from the report databases rather than from $A$. Recall that the identity of $s_j = \{bssid_j, H(bssid_j|M_j)\}$ is added to each database when $s_j$ is added to $S$. Because a malicious database colluding with $A$ could tie $bssid$ to a different signing key $M_{j'}$, clients only consider AP identities that the majority of report databases agree upon.

**Side-channel attacks.** Side-channels exposed in reports may potentially link reports if the adversary has additional information. For example, if only one user visits an AP on a given day, the AP can infer that any report with a timestamp on that day is from that user. If a user submits many reports on APs at a time when most users rarely submit reports, the receiving database may infer from the submissions' timing that they are linked. Since

we add a small amount of noise to timestamps and submission times, we believe we can defeat most of these attacks in practice without significantly degrading accuracy.

## 4.4 Location Context

This section describes how Wifi-Reports obtains geographic coordinates for reports and how summary statistics are filtered by wireless channel condition.

### 4.4.1 Geographic Positioning

To obtain coarse geographic coordinates for APs, we leverage previous work on beacon "fingerprints." The set of Wi-Fi beacons and their signal strengths observed from a location can be used to obtain geographic coordinates with a median accuracy of 25 meters when paired with a sufficiently dense war driving database [101]. Existing war driving databases are sufficient to facilitate this task (e.g., Skyhook [145] is used to geolocate iPods). Thus, Wifi-Reports clients include estimated coordinates in reports. To generate the location estimate in summary statistics for each AP, the database uses the centroid of all reported positions that are close together (e.g., within two city blocks). Although these positions may be off by tens of meters, we believe that they are sufficiently accurate for locating areas of connectivity on a map. Network names can be correlated with business names to improve accuracy (e.g., from Google Maps), but doing this is outside the scope of this dissertation. We note that coordinates are only needed to allow clients to search for AP summary statistics by location.

### 4.4.2 Distinguishing Channel Conditions

Wireless performance differs based on channel conditions, which vary based on fine-grained location and environmental conditions. The loss rate of a wireless channel is roughly inversely proportional to the SNR, barring interference from other stations or multi-path interference [86]. The most obvious approach is to use summary statistics that only consider the $k$ reports with SNR values closest to the currently observed SNR. However, this approach has two problems. First, it requires users to download a different summary for each possible SNR value for each AP. Second, it may not be possible to choose an appropriate $k$: if $k$ is too large, summaries will consider many irrelevant reports; too small and summaries become vulnerable to outliers and fraud.

Fortunately, the continuum of SNR values can be partitioned into three ranges with respect to wireless loss: a range where clients experience near 100% loss, a range where clients experience intermediate loss, and a range where clients experience near 0% loss [86]. Therefore, Wifi-Reports categorizes reports based on these three channel conditions. In other words, clients measure the median SNR of beacons sent by their AP. Reports are annotated with this median SNR. When a client makes a local prediction about an AP, it considers only previous reports taken in the same SNR range. In practice, the database creates one summary for each of the three ranges for each AP, so the client does not need to download all the reports for an AP.

Since measured SNR depends on the AP's transmit power, these three SNR ranges may be different for each AP. We estimate these ranges as follows: Typical scenarios exhibit an intermediate loss range of 10 dB [86], so we exhaustively search for the "best" 10 dB range that satisfies the expected loss rates. Specifically, let $\overline{t_>}$ be the mean measured throughput of reports taken with SNR larger than the 10 dB range, $\overline{t_=}$ be the average throughput of reports with SNR in the 10 dB range, and $\overline{t_<}$ be the average throughput of reports with SNR smaller than the 10 dB range. We find the 10 dB range that maximizes $(\overline{t_>} - \overline{t_=}) + (\overline{t_=} - \overline{t_<})$, or the differences between the mean throughput in the three ranges.[9] We assume that reports of connectivity failures experienced 100% loss (i.e., have throughput of 0). Finally, if $\overline{t_<} < 0.75 \cdot \overline{t_=}$, we likely only have measurements in one of the 100% or 0% loss ranges, so we put all measurements in a single range.

Figure 4.4 shows the estimated ranges for several APs in our measurement study that were visible from multiple locations. We note that we do not need the distinguishing algorithm to work perfectly to obtain accurate predictions. There is already measurement noise within a single loss region due to TCP's sensitivity to loss. Thus, very inaccurate summaries typically only arise due to mixing reports in the 0% loss region with the 100% loss region so it usually suffices to estimate these regions within 10 dB. Clients could also directly measure wireless loss, either by observing other users' traffic [160] or by actively probing each AP.

### 4.4.3 Discussion

**Client calibration.** We use SNR to differentiate wireless channel conditions, but the reported SNR may have a bias due to manufacturing defects in Wi-Fi NICs. Therefore,

[9] When we have more than a few samples (i.e., $\geq 5$), we use the median rather than the mean because it is more robust to outliers. Since the distribution of noise is likely Gaussian, the median is likely to be close to the mean.

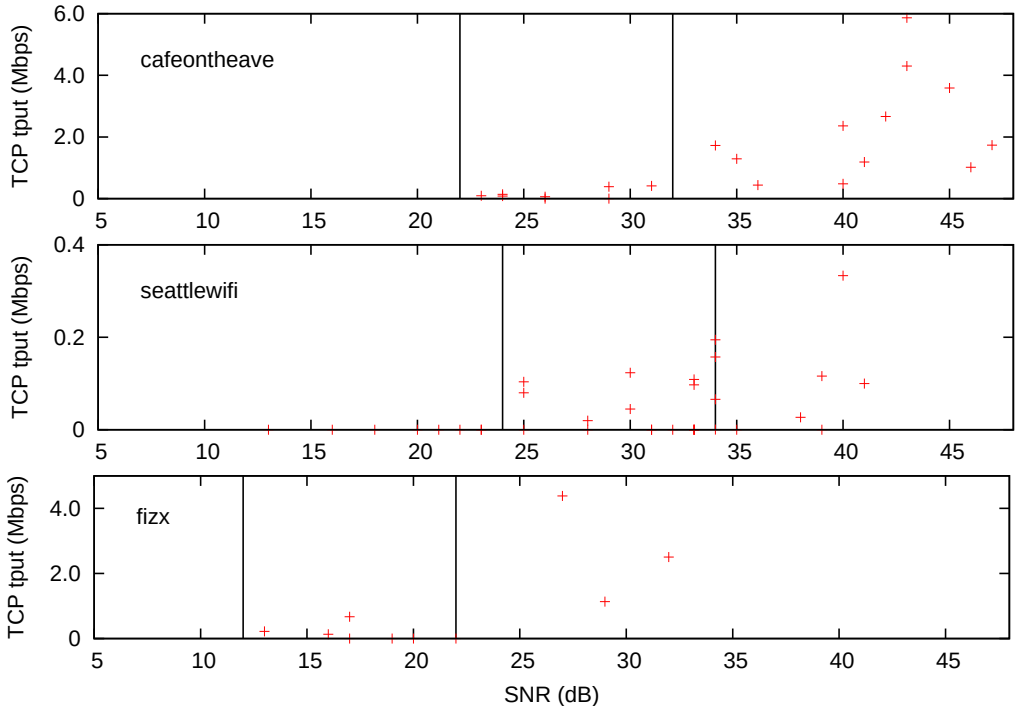


Figure 4.4: Estimated 100%, intermediate, and 0% loss regions for three APs in our measurement study.

different clients need to calibrate their reported SNR values. Previous work suggests that most of this error may be eliminated using a locally computed offset [86]. Reported SNR values for most cards after self-calibration may vary by 4 dB, a bias unlikely to affect our algorithm's accuracy significantly because the transitions between each SNR range are not sharply defined. To further improve accuracy, we can leverage existing self-calibration techniques that determine the biases of sensors (e.g., [14]). Implementing a distributed calibration algorithm is the subject of future work.

**Other environmental factors.** To improve prediction accuracy further, existing techniques can be used to measure and take into account other environmental factors that cause variation, such as multi-path interference and wireless contention [151, 160]. However, we found that contention is rare in our measurement study, so prediction accuracy is good even discounting these factors (see Section 4.5).

**User and AP mobility.** To localize reports, we currently assume that users and APs are stationary. If users are mobile, performance may change over time; we can detect user mobility by changing SNR values. Our current set of active measurements are short-

lived and can thus be associated with the SNR values observed when they are measured. Geolocating these mobile APs (e.g., those on a train) in a manner that makes sense is an area of future work.

## 4.5 Evaluation

We evaluate the utility and practicality of Wifi-Reports using our measurement study (see Section 4.1) and our implementation of the reporting protocol (see Section 4.3). This section presents our evaluation of three primary questions:

- Some APs' performance changes over time and at different locations. Are reports accurate enough to improve AP selection?
- Our reporting protocol provides location privacy at the cost of token generation overhead. Can Wifi-Reports provide users with a reasonable amount of location privacy with practical token generation overheads?
- A determined attacker may be able to trick the account authority into giving it a few accounts or collude with his friends to submit multiple fraudulent reports on an AP. How tolerant are summaries to such attacks?

### 4.5.1 AP Selection Performance

**Setup.** We use our measurement study to simulate two scenarios: First, we evaluate the scenario where a user chooses which hotspot to go to physically based upon the predicted performance of all hotspots nearby. In this scenario, a user is primarily interested in *prediction accuracy*; i.e., we want $\mathsf{predict}(s)/\mathsf{actual}(s)$ to be close to 1 for each AP $s$, where $\mathsf{predict}(s)$ is the predicted performance (e.g., throughput) of $s$ and $\mathsf{actual}(s)$ is the actual performance of $s$ when it is used. Second, we evaluate the scenario where the physical location is fixed (e.g., the user is already sitting down at a cafe) but the user wants to choose the AP that maximizes performance. This situation is comparable to the traditional AP selection problem [122, 151, 160]; i.e., given the set of visible APs $V = \{s_1, s_2, \ldots, s_n\}$, we want a selection algorithm $\mathsf{select}(\cdot)$ that maximizes $\mathsf{actual}(\mathsf{select}(V))$, where $s = \mathsf{select}(V)$ is the AP we choose. In this scenario, a user is primarily interested in relative *ranking accuracy*; e.g., for throughput, we would like to maximize $\mathsf{actual}(\mathsf{select}(V))/\max_{s \in V}(\mathsf{actual}(s))$. In Wifi-Reports $\mathsf{select}(V) = \text{argmax}_{s \in V}(\mathsf{predict}(s))$.

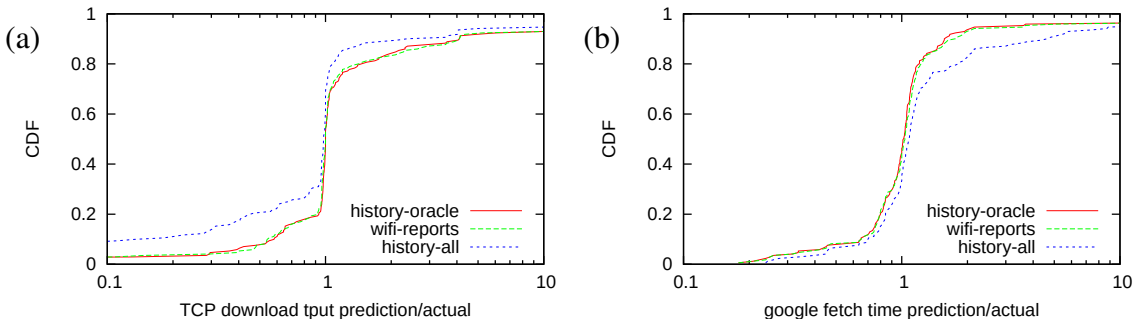


Figure 4.5: CDF prediction accuracy for (a) TCP download throughput and (b) Google fetch time over all trials on all official APs at their respective hotspots. Note the logarithmic scale on the x-axis.

We simulate these scenarios using our measurement study as ground truth. That is, we assume that after the user selects an AP $s$ to use, actual($s$) is equal to one of our measurements of $s$. We evaluate performance over all our measurement trials. To simulate the predict($s$) that would be generated by Wifi-Reports, we assume that all measurement trials except those for APs currently under consideration, are previously submitted reports. The reports for $s$ are summarized to generate predict($s$). This assumption implies that reports are generated by users that visit locations and select APs in a uniformly random manner. This is more likely to be the case when there are not yet enough reports in the system to generate any predictions. By counting devices associated with each AP in our measurement study, we observed that some users do currently use suboptimal APs. Thus, we believe that such reports would be obtained when bootstrapping new APs in Wifi-Reports.

**Prediction accuracy.** Figure 4.5 shows CDFs of prediction accuracy over all trials of official hotspot APs for TCP download throughput and Google response time. The x-axis in each graph shows the ratio of the predicted value over the actual achieved value. Values at 1 are predicted perfectly, values less than 1 are underestimates, and values more than 1 are overestimates. We compare three approaches for generating summary statistics. history-oracle shows the accuracy we would achieve if each summary summarizes only reports taken at the same hotspot location as the location under consideration; this requires an "oracle" because we would not automatically know the logical location where measurements are taken in practice. wifi-reports shows the accuracy when using Wifi-Reports' SNR filter before summarizing reports (see Section 4.4). history-all shows the accuracy when we summarize all reports to generate a prediction, regardless of the location where they were taken (e.g., even if the user is at Starbucks, the prediction includes reports of the same AP taken across the street).

In this graph, we focus on official APs, where we are sure to have some measurements in the 0% loss region, to better illustrate the impact of different channel conditions. Users in this scenario are more likely to desire a comparison of the 0% loss predictions rather than predictions in all three wireless channel conditions since they are choosing where to go. If an association or connection fails, we mark that trial as having 0 throughput and infinite response time. Recall that the summary function is median.

The graphs show that history-all underestimates TCP bandwidth and overestimates Google fetch time more often than history-oracle. This is because by including reports taken in the intermediate and near-100% loss regions, the median will generally be lower. In contrast, wifi-reports performs about as accurately as history-oracle, demonstrating that our SNR filter works well when we have some measurements in the 0% loss region. Furthermore, we note that at least 75% of predictions for both metrics are within a factor of 2 of the achieved value, while Figure 4.2 shows that the difference in the median throughputs and response times of official APs can be up to 50× and 10×, respectively. Therefore, most predictions are accurate enough to make correct relative comparisons.

**Ranking accuracy.** We now examine the scenario when a user is choosing between APs at a single location. Figure 4.6(a) and (b) show box-plots of achieved throughput and response time, respectively, when using one of several AP selection strategies to try to achieve the best performance at each location. best-open simulates Virgil [122], an algorithm that associates with and probes all open APs before selecting the best one. best-snr simulates the most common algorithm of picking the AP with the highest SNR value. This algorithm works well when wireless channel quality is the limiting factor. official simulates using the "official" AP of each location. We expect this algorithm to work well since we showed in Section 4.1 that the official AP is the best at most locations. Obviously this approach would not work at locations without an official AP. history-all simulates Wifi-Reports without the SNR filter. wifi-reports simulates Wifi-Reports. history-all and wifi-reports only generate a prediction for an AP if we have at least 2 reports to summarize; if no predictions for any AP are generated, they fall back to selecting the official AP. Finally, optimal shows the best performance achievable.

best-open performs the worst overall, failing to achieve any connections at tullys 1, starbucks 1, and cafeontheave since no open APs were visible. best-open performs better than all other algorithms only at yunnie, where most of the APs were open. We note that best-open is qualitatively different than the other selection algorithms because it cannot select any closed AP; we include it only to demonstrate that restricting the choice of APs to open ones often results in substantially suboptimal performance. Furthermore, best-open also has more overhead (linear in the number of open APs visible) than the others because it must actively test each AP.

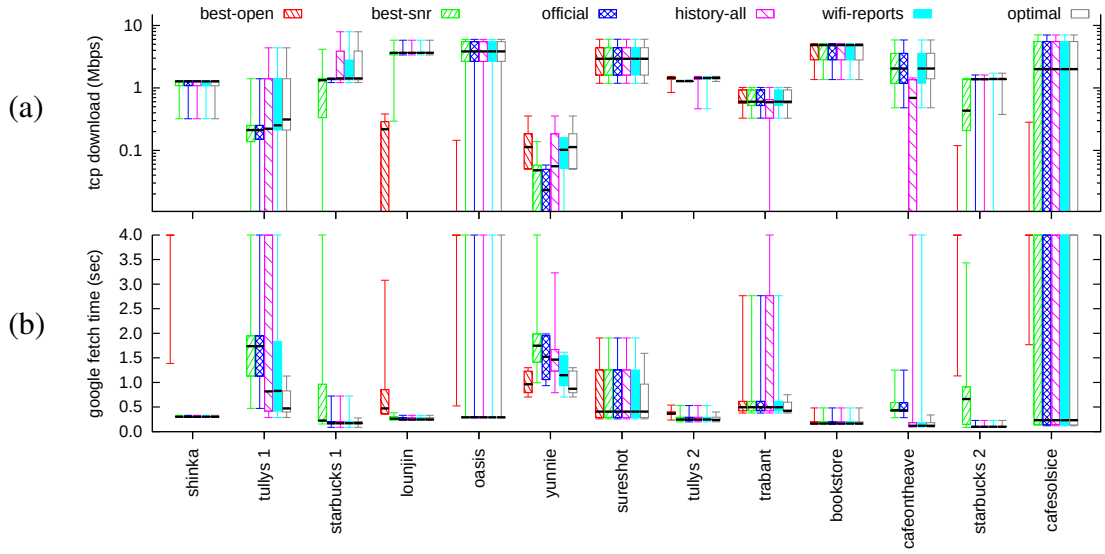


Figure 4.6: (a) Box-plot of achieved TCP download throughput when using each of five AP selection algorithms at each location. Note the logarithmic scale. Missing boxes for the best-open algorithm are at 0. (b) Box-plot of the achieved response time of `http://www.google.com` using each of five AP selection algorithms at each location. The whiskers that extend to the top of the graph actually extend to infinity (i.e., the fetch failed). missing boxes for the best-open algorithm are also at infinity. Each group of boxes are ordered in the same order as the key at the top.

history-all again demonstrates the need for the SNR filter. Without the SNR filter, Wifi-Reports would achieve poorer performance than official or best-snr at least 25% of the time at tullys 1, trabant, and cafeontheave.

In contrast, wifi-reports achieves performance closest to optimal for both metrics in all cases except for two. It achieves worse TCP throughput than best-open once at yunnie and worse response time than best-snr or official once at cafeontheave. In each of these cases, the AP chosen by wifi-reports experienced an association or DHCP failure. However, a real client would quickly fall back to the second best AP chosen by wifi-reports, which was the optimal one. Furthermore, wifi-reports is able to achieve higher bandwidth more of the time than all other algorithms at yunnie and starbucks 1 and better response time more of the time than all other algorithms at tullys 1 and cafeontheave. Thus, it performs strictly better in more locations when compared with each of the other approaches individually.

Finally, we note that unlike all other approaches, Wifi-Reports enables users to rank

| | mean | min | max | std dev | description |
|---|---|---|---|---|---|
| Server | 58.918 | 33.18 | 421.26 | 59.056 | generate key |
| Server | 3.979 | 3.87 | 6.29 | 0.222 | sign |
| Client | 95.517 | 18.00 | 560.45 | 47.364 | generate key |
| Client | 0.150 | 0.14 | 22.21 | 0.222 | verify |
| Client | 0.058 | 0.03 | 1.43 | 0.134 | unblind |
| Client | 0.006 | 0.00 | 1.88 | 0.027 | hash |
| Client | 0.003 | 0.00 | 1.88 | 0.019 | blind |

Table 4.1: Microbenchmarks of cryptographic processing times. All keys are 1024 bit RSA keys and SHA-512 is used as the hash function. All values in milliseconds with a resolution of 10 microseconds. 1000 trials were executed.

APs that are nearby but not visible. This is useful when users are willing to move to obtain better connectivity.

### 4.5.2 Report Protocol Performance

We implemented our reporting protocol (Section 4.3) in software to evaluate its practicality. We present measurements of its processing time, total token fetch time, and message volume using workloads derived from actual AP lists. We focus on the token generation phase (GENTOKEN) since, given a desired level of location privacy, its performance depends on actual densities of APs. The report submission phase (SUBMITREPORT) runs in constant time per report and uses standard fast RSA primitives.

**Setup.** We emulate a client that obtains the right to report on APs while at home (e.g., before or after traveling). Our client has a 2.0 GHz Pentium M and our account authority server used one 3.4GHz Xeon processor (the software is single threaded). Both run Linux and all cryptography operations used openssl 0.9.8. The bottleneck link between the client and server is the client's cable Internet connection (6 Mbps down, 768 kbps up). The round trip time from client to server is 144 ms.

**Processing time.** Table 4.1 presents microbenchmarks of each step of the protocol. All times are in milliseconds. The most heavyweight steps are the generation of 1024 bit RSA keys by both the client ($K_{ij}$) and server ($M_j$).[10] However, both keys can be generated anytime beforehand so these operations need not be executed inline in the GENTOKEN

[10]The standard deviation for key generation is high because the algorithm has a random number of iterations.

protocol. The remaining steps must happen inline, but have very low processing times. A server can sign a blinded message in under 4 ms, so it can process about 250 tokens per second, while a client can perform the verification and unblinding steps in roughly 0.2 ms, or 5000 times per second.

**Token fetch time.** A user who wants to obscure his locations within a region must perform GENTOKEN on all APs in that region. Figure 4.7 shows the end-to-end time to fetch tokens for all APs in each of the ten cities that JiWire [85] reports to have the most APs (as of November 15, 2008). JiWire lists commercial APs that service providers or users have manually added, which parallels how most APs are added to Wifi-Reports. Nonetheless, some commercial hotspots may not be listed by JiWire, so this graph serves to establish a lower bound for cities with many APs. Since a user can fetch these tokens at any time before submitting a report, even the longest delay, 5.5 seconds for all of New York, is completely practical. Even obtaining tokens for several cities at once is practical since each client only does this once in its lifetime.

WiGLE [167] is a database of all APs that war drivers have overheard, including both commercial and private APs. Figure 4.8, presents fetch times for all WiGLE APs in a 32 km square centered at each city. Since most APs listed are not intended to be used by the public (e.g., home APs) and WiGLE does not filter out erroneous or stale measurements, this graph serves as a loose upper bound on fetch times. Even so, the worst fetch time (Seattle) is 20 minutes. Since a client can batch `sig-request` messages for multiple APs, a reasonable approach would be to request all tokens and then retrieve them at a later time. In addition, by choosing a region granularity of less than a city, a client can achieve much better delay and still mask his locations to a reasonable extent. Figure 4.9 shows the CDF of number of WiGLE APs in $1\text{km}^2$ areas in each of the cities. Most cells in all cities have fewer than 188 APs, which only takes about 1 second to fetch, and no cell has more than 7400, which only takes about 30 seconds to fetch. Since commercial areas in most cities are not spread out, most will be covered by a small number of cells. Finally, we note that the server can parallelize the generation of each token to improve performance.

**Message volume.** A request for tokens transmits 173 bytes per token, while the response transmits 529 bytes per token. Therefore, our protocol is CPU-bound on the server even for a client on a cable modem. For example, it takes our client 8.7 minutes to send all requests for Seattle APs on WiGLE and 3.4 minutes to receive the replies (these latencies are included in the token fetch times reported above).

**Admission rate and server cost.** We next estimate the rate at which users can join given limited server resources. To simulate "average" American users joining the system, we assume that each user requests all tokens from one of the cities shown in Figure 4.7, chosen

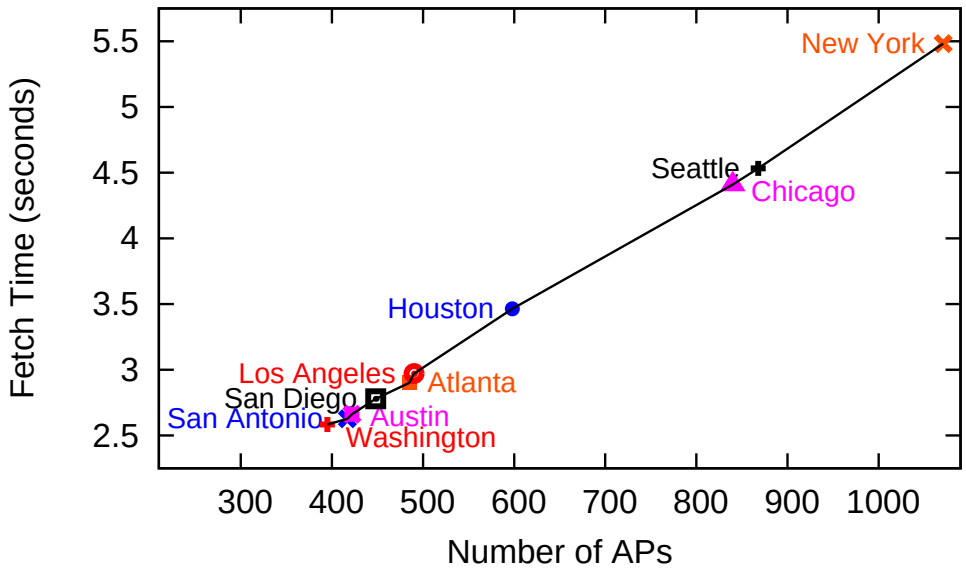


Figure 4.7: Time to acquire the right to report on all APs listed by JiWire in ten cities. The cities presented are the ten with the most APs, according to JiWire as of November 15, 2008.

at random weighted by each city's population (according to 2007 U.S. census data [158]). While a user may request more, the authority rate limits each user to prevent denial-of-service attacks.

Suppose the authority has $x$ CPUs. For JiWire APs, it can admit 27,455$x$ new users per day. For example, if the authority has 100 CPUs, it can admit the entire population of these cities in 5.6 days. How much would this overhead cost over a system that stores reports without privacy? If deployed on Amazon's EC2 [9], this would only cost about 0.02 cents per user for CPU and bandwidth resources. For all WiGLE APs, the authority can admit 165$x$ new users per day and the overhead cost would be about 2.6 cents per user. This one-time cost is a very small fraction of the $5+ each user would have to spend to use most commercial APs just for one day. There are also recurring costs incurred for computing tokens for new APs that are added and, if enabled, signing reports for rate limiting (see the end of Section 4.3.3). However, these costs are also trivial. For example, even if 10 new hotspots appear in each city every week and every user submits 10 new reports per week, the recurring cost would only be about 0.02 cents per user per year.

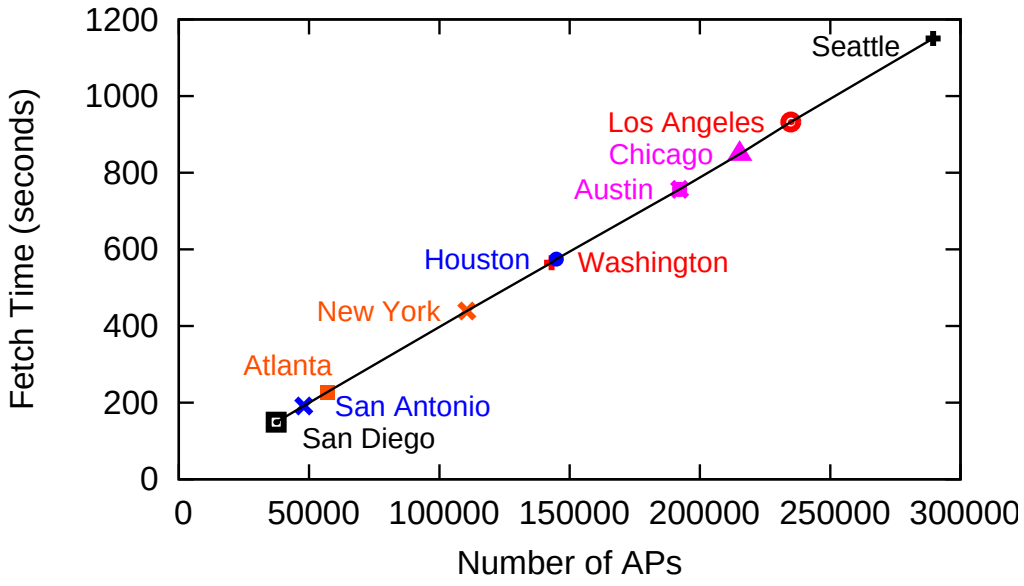


Figure 4.8: Time to acquire the right to report on all APs listed by WiGLE in 32 km x 32 km squares centered on each of ten cities.

### 4.5.3 Resistance to Fraud

Summary values are robust to fraudulent reports that try to boost or degrade an AP's value because we use summary functions that are resilient to outliers. However, since there is variability in honest reports as well, a small number fraudulent reports may still be able to degrade prediction accuracy, e.g., by shifting the median higher or lower.

**Setup.** We consider the same scenario as in Section 4.5.1. To evaluate the extent that fraudulent reporting can degrade accuracy, we simulate an adversary that tries to boost the predicted TCP download throughput of an AP by submitting reports that claim the AP achieves 54 Mbps, the maximum theoretically possible in 802.11g. In this evaluation users only consider each AP's 0%-loss summary, so we assume that each adversarial user submits one report with SNR in the middle of this range. Although he could submit more, they would not change the summary since only one report per user is used. We vary the power of the adversary by varying the number of users that collude to submit these fraudulent reports. A typical AP would also have many honest reports. Therefore, we simulate each AP with 100 reports total: $x$ are the fraudulent reports described above and $100 - x$ are honest reports that are randomly sampled (with replacement) from our $\sim$10

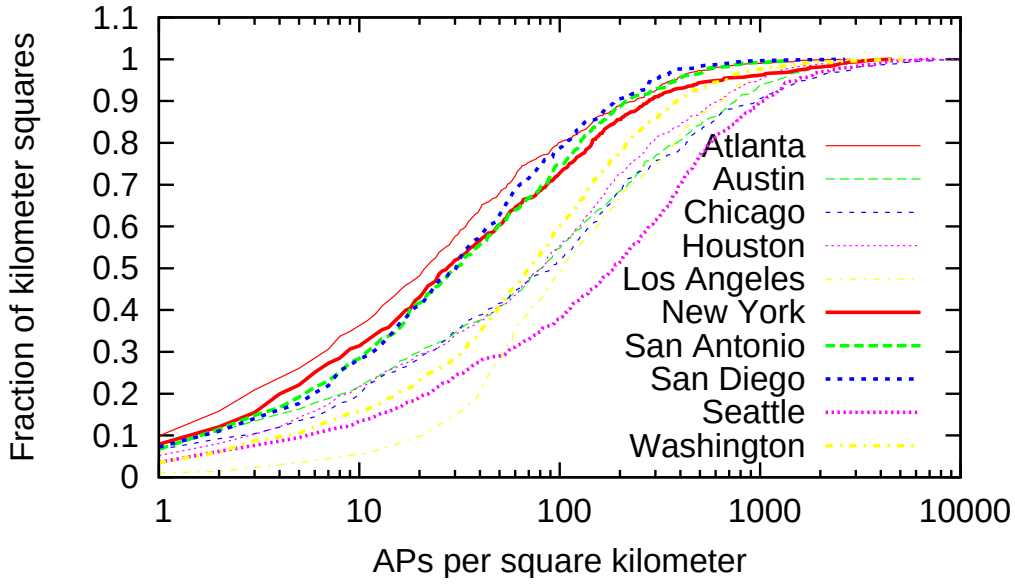


Figure 4.9: CDF of the number of APs listed by WiGLE in each 1 km$^2$ region of a 32 km x 32 km grid centered on each of ten cities.

actual measurements per AP. Note that even if the total number of reports is different, our results still hold on expectation if the ratio of fraudulent to total reports remains the same. The remainder of our simulation setup is identical to Section 4.5.1. For comparison to Figure 4.5(a), we again focus on official APs.

**Accuracy.** Figure 4.10 shows Wifi-Reports' prediction accuracy on official APs as we vary the percentage of fraudulent reports. Negligible degradation of accuracy is observed when up to 10% of reports are fraudulent. Even with 30% of fraudulent reports, most predictions are still correct within a factor of 2. However, when 50% of reports are fraudulent, most predictions are gross overestimates. This result is expected since the median function is not robust to 50% or more outliers larger than the actual median.

**Discussion.** We note that even if an adversary is successful in luring honest clients to a poor AP, those clients will submit reports that correct the summary statistics. Successful fraud attacks that degrade a good AP's reputation (or contract its 0%-loss SNR range) are harder to correct because honest users may be dissuaded from using that AP. However, since cost, venue, and other external factors will influence selections in practice, we believe some honest users will eventually report on these APs and correct their summary statistics.

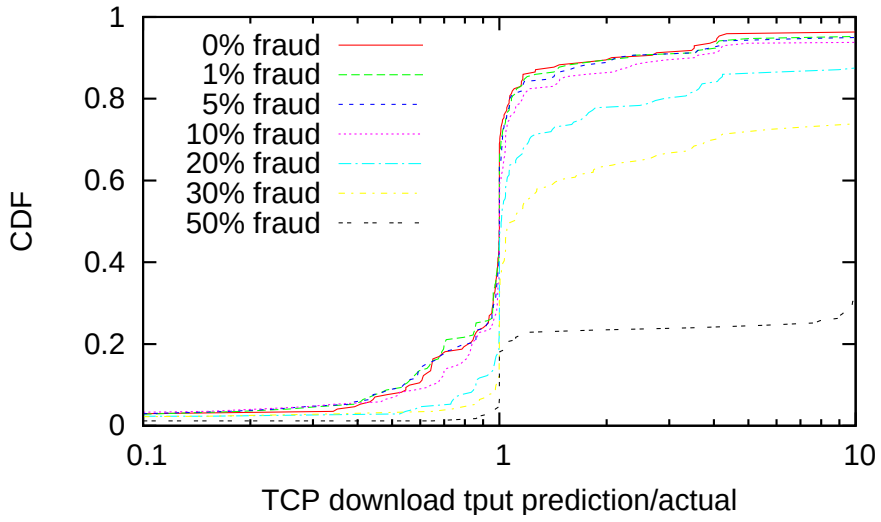


Figure 4.10: CDF of prediction accuracy for TCP download throughput of all official APs at their respective hotspots. We vary the percentage of fraudulent reports that claim throughput is 54Mbps. Note the logarithmic scale on the x-axis.

## 4.6 Related Work

Wifi-Reports is related to four areas of previous work: recommender systems, collaborative sensing, AP selection, and electronic cash/secure voting protocols.

**Recommender systems.** Previous proposals for reputation-based AP location systems, such as Salem et al. [141], do not protect contributors' location privacy. Moreover, Salem et al.'s protocol assumes APs can predict their performance and it does not address varying wireless channel conditions. Of course, having users report on items or services to ascertain their value is a well known idea (e.g., for a survey see [5]). For example, Broadband reports [37] rates ISPs using user-reported speed tests (e.g., [149]) that measure their back-haul capacities. However, Broadband reports takes few measures to prevent fraud. This may be because, unlike the identity of an AP, it is difficult to forge the IP address that identifies the ISP in a speed test. Furthermore, it is relatively easy to limit sybil attacks [52] because a user is identified by an IP address, which is hard to spoof while maintaining the necessary TCP connection for reporting.

Some recommendation systems use collaborative filtering (CF) (e.g., [162, 176]) to identify users that submit many bad reports. However, these techniques require that all reports from the same user are linked and thus do not protect privacy, which is important

when location information is at stake. Some proposed CF techniques can limit the exposure of this information by using secure multi-party voting [38, 39]. However, these techniques require all users to be simultaneously online to update summary statistics, and thus are impractical for services that have many users and continuous submission of reports.

**Collaborative sensing.** A number of recent proposals use mobile devices as collaborative sensor networks (e.g., [96, 4]), but they do not address the unique challenges of AP measurement and reporting. Anonysense [45] is one such platform that ensures that reports are anonymous by using a mix network. However, Anonysense relies on a trusted computing base (TCB) to prevent fraudulent reports and cannot prevent non-software based tampering (e.g., disconnecting a radio antenna) nor fraudulent reports from clients that collude with APs without a TCB. These attacks are simple to carry out to modify wireless measurements. Nonetheless, the Wifi-Reports measurement client could also leverage a TCB to mitigate fraud even more.

[122, 151, 160] argue for metrics other than signal strength for ranking access points, but only consider metrics that can be instantaneously measured by a single client. We showed in Section 4.5 that leveraging historical information out-performs direct measurement [122] because it isn't always possible to test an AP before use. In addition, Wifi-Reports is the only system that enables users to evaluate APs that are not in range, such as when searching for an AP in a hotspot database. Nonetheless, our work is complementary to [151] and [160], which can better estimate the quality of the wireless channel when it is the performance bottleneck.

**Electronic cash and secure voting.** Wifi-Reports uses blind signatures in a manner similar to well-known electronic cash [42, 41] (e-cash) and secure voting [60] (e-voting) protocols. However, unlike traditional e-cash protocols where a user has multiple tokens that can be spent on any service, a user of our reporting protocol has a single token per service that can only be used for that service. Traditional e-voting protocols typically assume that all users vote (e.g., report) on all candidates (e.g., APs) before tallying the votes, whereas reports are continuously tallied in Wifi-Reports but a precise count is not necessary. As a consequence, our reporting protocol is simpler than traditional e-cash and e-voting protocols, but, like these protocols, it relies on an account authority and distributed talliers (e.g., report databases) to prevent attacks.

## 4.7 Summary and Discussion

This chapter presented Wifi-Reports, a service that improves AP selection by leveraging historical information about APs contributed by users. A system like Wifi-Reports is important because our measurement study, the first of commercial APs, showed that there is substantial diversity in AP performance, even among those close by. Wifi-Reports can handle reports submitted at different locations, protects users' location privacy, and is resilient to a small fraction of fraudulent reports. The central contribution of Wifi-Reports' design is to demonstrate how crowd-sourced LBSes can make user-submitted reports unlinkable, yet also ensure that each user has limited influence in summarized results.

### 4.7.1 Discussion

Although we presented our reporting protocol in the context of a hotspot directory service, it is applicable to crowd-sourced recommender systems more generally. The protocol can be applied to other collaborative reporting services by replacing the set of APs, $S$, with the set of items being reported on. Nonetheless, we note that there are two important limitations. First, the protocol's practicality is dependent on our ability to group items into subsets that are reasonably small and not revealing. Second, the protocol can not be applied when collaborative filtering is required.

**Reasonably sized subsets.** Users in Wifi-Reports can fetch tokens in a practical amount of time because we presumed that they are willing to reveal a coarse grain region that they have visited (e.g., a city) and that each of these regions does not contain more than a million APs. If the set of items that a user has to fetch to obtain sufficient privacy is larger than several million, then the cost of generating tokens becomes prohibitive. Therefore, an important challenge when applying this protocol to other crowd-sourced recommender systems is how to subdivide the set of items into subsets of reasonable size and that have appropriate privacy semantics.

**Collaborative filtering.** In contrast to plain query-based LBSes and crowd-sourced LBSes, collaborative filtering (CF) services help users by finding users with similar interests. For example, a dating site might try to match users that have been to similar places. CF services that match users based on location history actually *need* to link a user's location samples together to function using existing collaborative filtering techniques. For example, techniques to prevent sybil attacks and fraud in such systems, such as DSybil [176], actually rely on tracking the history of each user's votes or reports. Although some proposed CF techniques for peer-to-peer systems can limit the exposure of user histories by

using dimensionality reduction and secure multi-party voting [38, 39], these techniques are difficult to apply when clients which are not always online. Thus, designing appropriate privacy models and mechanisms for CF services is a final important area of future work.

# Chapter 5

# Conclusions and Future Work

Remember that the primary threat to location privacy comes from the collection of location traces, i.e., a set of location samples known to be from the same device or user. Eavesdroppers and crowd-sourced LBSes can collect these traces when we use existing wireless protocols because they can link a user's *identity* to the *locations* that he visits. We conclude this dissertation with a summary of how our work limits the ability of these parties to link identities to locations. We then discuss some open problems that remain.

## 5.1 Summary

This dissertation made the following thesis: *Existing protocols and techniques that wireless devices use to discover and communicate with each other pose risks to users' location privacy. It is, however, possible to redesign these protocols and techniques to substantially mitigate location privacy threats without degrading their functionality or practicality.*

Recall that the usage model of wireless protocols such as 802.11 typically has four phases: select, rendezvous, communicate, and report. Our work shows that existing link layer protocols reveal implicit identifiers during the rendezvous and communication phases even when pseudonym schemes to mask unique identifiers in those protocols (e.g., MAC addresses in 802.11) are employed. In addition, we showed that crowd-sourced Wi-Fi directory services are highly useful, but currently do not protect the anonymity of user reports because they need to limit the number of fraudulent reports that malicious users can submit. Thus, these services can currently link our identities to our locations during the reporting phase. Although previous work showed how using coarse grain or noisy location queries can be used to conceal our identity during the selection phase, these

techniques can not be applied to crowd-sourced LBSes.

In light of these shortcomings, we showed that we can build link layer protocols that conceal all transmitted bits with comparable efficiency and functionality as existing protocols. For example, SlyFi can replace 802.11 with very little degradation in performance if both ends of a communication support it. Concealing all transmitted bits eliminates all explicit identifiers and makes it substantially more difficult for eavesdroppers to detect implicit identifiers during the rendezvous and communication phases. Furthermore, we showed that we can build location-based recommender systems, such as Wi-Fi directory services, that preserve the anonymity of user-submitted reports and limit the number of fraudulent reports a single user can submit, thereby concealing our identity during the reporting phase. Wifi-Reports shows that a privacy-preserving hotspot directory service would be practical today.

## 5.2 Contributions

This dissertation made contributions in answering three principle questions:

| | |
|---|---|
| Are simple pseudonym protocols sufficient to prevent location tracking by eavesdroppers? | *No* |
| Can we build complete and practical link-layer protocols that conceal all identifiers? | *Yes* |
| Can we build practical crowd-sourced LBSes for wireless service location in a manner than preserves both privacy and accountability? | *Yes* |

### 5.2.1 Implicit Identifier Tracking Threats in Practice

**Central Insight.** To our knowledge, we are the first to show with empirical evidence that design considerations beyond eliminating explicit identifiers, such as unique MAC addresses, must be addressed to protect anonymity in wireless networks. This is because, even without a unique address, other characteristics of users' 802.11 traffic can identify them implicitly and track them with high accuracy. An example of such an *implicit identifier* is the SSID (i.e., network name) of access points that his laptop tries to discover. In a population of several hundred users, the set of these SSIDs might be unique to one individual; thus, the mere observation of this IP address would indicate the presence of that user.

**Key Results.** We identified four previously unrecognized implicit identifiers that remain even when previously proposed MAC address pseudonym schemes are applied: network destinations, network names advertised in 802.11 probes, differing configurations of 802.11 options, and sizes of broadcast packets that hint at their contents. The later three of these implicit identifiers remain even when link-layer encryption is employed. Furthermore, we developed an automated procedure to identify users. This procedure allows us to quantify how much information implicit identifiers, both alone and in combination, reveal about several hundred users in three empirical 802.11 traces. Our evaluation using this technique shows that many of these users emit highly discriminating implicit identifiers, and, thus, even a small sample of network traffic can identify them more than half (56%) of the time in public networks, on average. Moreover, we will almost never mistake them as the source of other network traffic (1% of the time). Since adversaries will obtain multiple traffic samples from a user over time, this high accuracy in traffic classification enables them to track many users with even higher accuracy in common wireless networks. For example, an adversary can identify 64% of users with 90% accuracy when they spend a day at a busy hot spot that serves 25 concurrent users each hour. Even if the hot spot employed link-layer encryption, 31% of users can be still identified with 90% accuracy.

### 5.2.2 Practical Identifier-free Link Layer Protocols

**Central Insight.** To address the implicit identifier problem, we developed SlyFi, a link layer protocol that reveals *no* transmitted bits to eavesdroppers. The obvious difficulty with simply removing all explicit fields is that they play key roles in the efficient operation of existing protocols. For example, a connection identifier allows a device to decide whether it is the destination of a message by using a simple compare operation. Our central contribution is a set of two efficient primitives for constructing service discovery protocols and subsequent data transfer protocols that conceal all transmitted bits. These primitives leverage assumptions about the rendezvous and communication phases of the wireless protocol to synchronize cryptographically secure pseudorandom sequences. As a consequence, in both cases a device can determine whether it is the recipient of a message with lightweight table look-ups. We demonstrate that all wireless protocol features that rely on identifiers—service discovery, packet filtering, and address binding—can be supported without exposing them.

**Key Results.** We have implemented SlyFi on commodity 802.11 NICs and our experiments show that SlyFi's performance impact is modest, as it uses only cryptographic operations already used by WPA. In particular, we showed that a SlyFi client can discover and associate with services even faster than 802.11 with WPA using PSK authentication.

SlyFi's overhead results in a throughput degradation that is only slightly greater than that of WPA with CCMP encryption (10% vs. 3%).

### 5.2.3 Privacy-preserving Crowd-sourced Location-based Systems

**Central Insight.** SlyFi only protects communication sent between devices that already trust each other. To support improved discovery of new public wireless services, such as 802.11 APs, we need hotspot directories that accept user submitted reports and summarize them (e.g., a crowd-sourced LBS). For example, if users report the bandwidth of APs that they use, the hotspot directory can list the median bandwidth of all APs to help users select among them. However, in order to preserve location privacy, a user should not have to reveal that he used an AP to report on it. Otherwise he would implicitly reveal a location that he visits. At the same time, however, a few users should not be able to significantly skew an AP's summary statistics because some may have an incentive to submit fraudulent reports, e.g., to promote APs that they own. Our central contribution is a report submission protocol that tolerates a few misbehaving users and does not require the disclosure of location related information to anyone, including the LBS. Our protocol leverages blind signatures to ensure that the service can regulate the number of reports that each user submits, but cannot distinguish one user's reports from another's. To demonstrate the practicality of this protocol we presented the design, implementation, and evaluation of Wifi-Reports, a hotspot directory service that implements this protocol. Although we presented this reporting protocol in the context of a hotspot directory service, it is also applicable to some other location-based collaborative recommender systems.

**Key Results.** We conduct the first study that examine the attributes of commercial encrypted and "pay-for-access" APs in the wild. Our results suggest that hotspot recommender systems would have enormous utility because there is a large range in AP performance. For example, different APs' median throughputs and response times differ by up to $50\times$ and $10\times$, respectively. We show that Wifi-Reports can predict AP throughput and response time to within a factor of 2 at least 75% of the time. Moreover, we show that Wifi-Reports generates accurate summary statistics for each AP even if 10% of that AP's users collude to promote it.

## 5.3 Future Work

A number of important questions about location privacy in the context of wireless protocols and services remain. In this section, we outline several open problems and suggest some initial directions for solutions where possible.

### 5.3.1 Private Key Distribution for Device Discovery

In both existing secure wireless protocols and in SlyFi, an important challenge is how to bootstrap the symmetric or public keys that are necessary before two devices are able to rendezvous and authenticate each other for the first time. This bootstrapping phase is typically done out-of-band between the selection and rendezvous phases of the wireless usage model. The two traditional classes of key establishment mechanisms, *pairing* and *certificates*, can be used in some scenarios, but may also be insufficient to establish keys in many useful service discovery settings. In this section, we outline the challenges and some initial solutions for *key establishment* protocols that conceal device identities [129].

**Challenges.** Pairing [152] refers to the techniques used to establish keys on two personal devices that a user wants to connect together (e.g., Bluetooth peripherals). Pairing mechanisms usually involve the physical exchange of some secret, such as a verbally exchanged passkey, or the use of an out-of-band channel, such as direct physical contact. These mechanisms assume that users of the devices can identify them physically, which is often not the case (e.g., when trying to find an 802.11 AP). Moreover, all these mechanisms assume that a client *already knows* the specific service it wants to discover. Ironically, this assumption means that to use pairing, *users* have to discover services before their *devices* can discover them, which defeats the purpose of "automatic" service discovery. Service discovery is often useful because it enables users to find services they do not yet know about (two such scenarios are described below).

Public wireless services, such as infrastructure APs, can advertise certificates signed by trusted authorities (e.g., VeriSign) to prove their authenticity to clients. This is the trust model we use for public hotspots in Wifi-Reports. However, private services can not offer certificates to unknown clients without violating their own privacy. Similarly, clients can not privately offer proof of identity before authenticating a service. Even predistribution of certificates via out-of-band channels is difficult because mobile devices are often disconnected.

**Initial solutions.** Devices in two mutually trusted domains can often assume bilateral trust. For example, if Alice and Bob are friends, they may allow all their current and

future devices to discover each other. They might naïvely try to achieve this either by sharing a single private key among all devices in one domain, or by exchanging all their device keys. However, the former approach compromises all the devices if even one is stolen, and the later approach does not enable the discovery of new devices that Alice and Bob obtain after the key exchange.

To establish keys automatically in this scenario, we can leverage *anonymous identity based encryption* [3] (AIBE), a public key encryption scheme primarily used for confidential email. Using AIBE, Alice can assign a different private key to each device and Bob can encrypt a discovery message to that device using a human-readable string as the encryption key and a publicly known encryption algorithm. This string is chosen based on an agreed upon naming convention (e.g., Alice.iPhone can be the encryption key for Alice's iPhone) but a message encrypted with it hides both the key and the recipient. A trusted authority provides Alice with the private decryption key, ensuring that only she can decrypt messages encrypted with keys beginning with Alice (i.e., a unique user name she uses with this authority). The authority also publicly publishes the encryption algorithm, which is the same for all its users (i.e., the same algorithm is used whether the key is Alice.iPhone, or Alice.Zune, or Charlie.iPhone, etc.).

For example, suppose Bob and Alice each purchase a new iPhone, each preloaded with AIBE private keys. Bob configures his iPhone to trust (the string) Alice and Alice configures hers to trust Bob (e.g., by adding each string to their respective address books). If Bob and Alice are nearby, their iPhones could discover each other and use 802.11 to connect their calls, without first needing to exchange keys (indeed, Bob need not even know that Alice has an iPhone). To do this, Bob's iPhone simply sends a discovery message encrypted using the string Alice.iPhone and only Alice's iPhone can receive it.

Two mutually trusted devices may also be willing to trust each other's relations. For example, Bob's iPhone may permit all 802.11 APs that Alice's iPhone uses to discover it. Similarly, some of these APs may permit Alice's friends to discover them. Bob might naïvely attempt to bootstrap keys for these APs by having Alice give him all their public keys. However, this approach does not work for new APs that Alice uses after this key exchange, and it forces Alice to reveal all her AP relations to Bob.

Instead, we can leverage a private social proximity test [58] to automatically establish keys in this scenario. This test enables Bob's iPhone to send a message that can only be read by devices that Alice has allowed to trust her friends, without having to explicitly tell Bob about any of them. Note that Bob will still learn about Alice's relationship with an AP when he discovers it. However, Alice, who must agree to use this mechanism, may be willing to permit this revelation.

**Discussion.** Although neither of these mechanisms provide absolute confidentiality as they require additional trust assumptions, they are only used when devices attempt to discover others with which they have not established keys. A symmetric key can then be exchanged for subsequent rendezvous attempts using SlyFi. Thus, an important research challenge is to determine how to limit the use of these mechanisms to those circumstances for which they are truly needed. For example, in the case of 802.11, clients generally prefer known APs to unknown ones. Thus, these mechanisms are not needed when known APs are present.

Finally, the rapid evolution of service discovery scenarios means that we do not yet have a clear picture of all use cases. Emerging scenarios may require access based on capabilities that these mechanisms do not support. For example, a wireless network may want to be visible only to clients in a specific physical area. Private key establishment mechanisms for these scenarios are an important area of future work.

### 5.3.2 Concealing Higher-layer Explicit Identifiers

In this dissertation, we focused on concealing link-layer identifiers and those exposed to LBSes. However, identifiers that remain in IP and application layer protocols may still be observed by some entities during the communication phase of the wireless usage model. The most prominent example of these identifiers are application-level service identifiers used in application-level service discovery. Windows, Mac OS X, and Linux operating systems use service discovery protocols such as NetBIOS, UPnP SSDP, SLP, and Multicast DNS (mDNS) to discover other devices and services on the same local network. User-level applications such as iTunes and iChat use service discovery to find other instances of the same application on the local network. These protocols function in the same way as link-layer rendezvous protocols (i.e., using either probes or announcements). Although eavesdroppers that are not on an encrypted local network can not observe such identifiers, users may not always trust all devices on the local network either (e.g., at a hotspot). Fortunately, the same primitive we used to conceal identifiers in link-layer rendezvous (Tryst) can be used to conceal identifiers in higher-layer discovery protocols. Doing so is mostly an engineering challenge. However, there is also a research challenge in bootstrapping the symmetric keys necessary to use Tryst. This is a more important challenge for application layer discovery protocols because they are often used to support "zero-configuration" networking, and requiring manual key exchange would hinder ease-of-use. We proposed some initial solutions to the key exchange problem above.

### 5.3.3 Concealing Physical Layer Implicit Identifiers

In addition to higher-layer explicit identifiers, lower-layer implicit identifiers at the physical layer that are exposed during the rendezvous and communication phases of the wireless usage model may be detected by adversaries with special equipment [16, 36, 70, 143]. We believe that such equipment, such as signal analyzers that cost tens of thousands of dollars, are out of reach of casual eavesdroppers and do not expect large scale surveillance networks constructed using them soon. However, the cost of such equipment will invariably come down as it is commoditized. Therefore, it is also important to begin to develop countermeasures to conceal physical layer fingerprints as well. We believe that such concealment is not technically difficult with hardware modifications (e.g., Brik *et al.* [36] used fixed calibration errors to fingerprint wireless cards, so inserting randomized noise to these errors would mask the fingerprints). However, doing so would likely add to the cost of wireless devices, so a thorough understanding of the economic trade-offs is crucial to determining whether deploying such countermeasures at scale is practical.

# Bibliography


[1] Martín Abadi and Cédric . Private authentication. *Theor. Comput. Sci.*, 322(3):427–476, 2004. 2.7.1, 3.1, 3.3.1

[2] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *TCS '00: Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, pages 3–22, London, UK, 2000. Springer-Verlag. 3.5.1, 3.5.2

[3] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malonelee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *CRYPTO*, 2005. 5.3.1

[4] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *Pervasive Computing, IEEE*, 6(2):20–29, 2007. 4.6

[5] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005. 4.2, 4.6

[6] Advisory group asks eu to consider pentium iii ban. CNN, `http://www.cnn.com/TECH/computing/9911/29/eu.p3.ban.idg/index.html`, November 1999. 1

[7] William Aiello, Steven M. Bellovin, Matt Blaze, John Ioannidis, Omer Reingold, Ran Canetti, and Angelos D. Keromytis. Efficient, DoS-resistant, secure key exchange for internet protocols. In *CCS*, 2002. 3.4.1

[8] Aditya Akella, Glenn Judd, Srinivasan Seshan, and Peter Steenkiste. Self-management in chaotic wireless deployments. In *MobiCom*, pages 185–199, New York, NY, USA, 2005. ACM Press. 2.3

[9] Amazon elastic compute cloud. `http://aws.amazon.com/ec2/`. Accessed on 03/26/2009. 4.5.2

[10] Jari Arkko, Pekka Nikander, and Mats Näslund. Enhancing privacy with shared pseudo random sequences. In *13rd International Workshop on Security Protocols*, April 2005. 3.1

[11] Frederik Armknecht, Joao Girão, Alfredo Matos, and Rui L. Aguiar. Who said that? privacy at link layer. In *INFOCOM*. IEEE, 2007. 3.1, 3.7.1

[12] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000. 2.2, 3.6.4

[13] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM (2)*, pages 775–784, 2000. 2.3

[14] Laura Balzano and Robert Nowak. Blind calibration of sensor networks. In *IPSN*, pages 79–88, 2007. 4.4.3

[15] M. Barbaro, T. Z. Jr., and S. Hansell. A face is exposed for AOL searcher no. 4417749, August 2006. 1

[16] M. Barbeau, J. Hall, and E. Kranakis. Detection of rogue devices in bluetooth networks using radio frequency fingerprinting. In *Proceedings of the 3rd IASTED International Conference on Communications and Computer Networks, CCN 2006*, October 2006. 2.2, 3.6.4, 5.3.3

[17] A. Barth, D. Boneh, and B. Waters. Private encrypted content distribution using private broadcast encryption. In *Financial Crypto*, 2006. 3.8.1

[18] Kevin Bauer, Damon McCoy, Ben Greenstein, Dirk Grunwald, and Douglas Sicker. Using wireless physical layer information to construct implicit identifiers. In *1st Hot Topics in Privacy Enhancing Technologies*, July 2008. 2.2, 3.6, 3.6.4, 3.8.1

[19] Kevin Bauer, Damon McCoy, Ben Greenstein, Dirk Grunwald, and Douglas Sicker. Physical layer attacks on unlinkability in wireless lans. In *Proceedings of the 9th Privacy Enhancing Technologies Symposium*, August 2009. 2.2, 3.6, 3.6.4, 3.8.1

[20] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 394–403, 1997. 3.5.1

[21] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003. 4.3.3

[22] M. Bellare and B. Yee. Forward-security in private-key cryptography. *Topics in Cryptology - CT-RSA'03, LNCS*, 2612, 2003. 3.4.1

[23] Mihir Bellare. Practice-oriented provable-security. In *ISW '97: Proceedings of the First International Workshop on Information Security*, pages 221–231, London, UK, 1998. Springer-Verlag. 3.5.1

[24] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, 2001. 3.3.1

[25] Mihir Bellare and Phillip Rogaway. Introduction on modern cryptography, 2005. `http://www-cse.ucsd.edu/users/mihir/cse207/classnotes.html`. 3.5.1

[26] Benetton to tag 15 million items. RFID Journal, `http://www.rfidjournal.com/article/articleview/344/1/1/`, March 2003. 1

[27] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003. 1.1, 1.3, 4.2.1

[28] C. Bettini, X.S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *2nd VLDB workshop on secure data management*, pages 185–199, 2005. 1.3

[29] Big boss is watching. CNET News.com, September 2004. http://news.com.com/Big+boss+is+watching/2100-1036_3-5379953.html. 1

[30] A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of a5/1 on a pc. In *FSE '00: Proc. 7th International Workshop on Fast Software Encryption*, pages 1–18, 2001. 1

[31] George Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *Proc. Privacy Enhancing Technologies Workshop (PET 2005)*, 2005. 2.2

[32] John Black and Phillip Rogaway. Cbc macs for arbitrary-length messages: The three-key constructions. *J. Cryptol.*, 18(2):111–131, 2005. 3.5.1, 3.5.4, 3.5.5, 1

[33] Bluetoothtracking.org. http://www.bluetoothtracking.org/. 1, 1.1.1

[34] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proc. MobiCom*, pages 180–189, 2001. 1

[35] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999. 2.5.1

[36] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 116–127, New York, NY, USA, 2008. ACM. 2.2, 3.6.4, 5.3.3

[37] Broadband reports. http://www.dslreports.com/. 4.6

[38] J. Canny. Collaborative filtering with privacy. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 45–57, 2002. 4.6, 4.7.1

[39] John Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR*, pages 238–245, New York, NY, USA, 2002. 4.6, 4.7.1

[40] Ranveer Chandra, Ratul Mahajan, Venkat Padmanabhan, and Ming Zhang. CRAWDAD data set microsoft/osdi2006 (v. 2007-05-23). crawdad.cs.dartmouth.edu/meta.php?name=microsoft/osdi2006. (document), 3.1

[41] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, pages 319–327, 1990. 4.6

[42] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology*, pages 199–203. Springer-Verlag, 1982. 4.6

[43] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. *SIGCOMM Comput. Commun. Rev.*, 36(4):39–50, 2006. (document), 2.4, 3.1

[44] Computer Research Association. Four grand challenges in trustworthy computing, November 2003. 1

[45] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys*, pages 211–224, 2008. 4.6

[46] Landon P. Cox, Angela Dalton, and Varun Marupadi. SmokeScreen: Flexible privacy controls for presence-sharing. In *MobiSys*, 2007. 3.1, 3.8.1

[47] Crawdad. A community resource for archiving wireless data at Dartmouth. 2.1, 2.4

[48] Tassos Dimitriou. A lightweight rfid protocol to protect against traceability and cloning attacks. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 59–66, Washington, DC, USA, 2005. IEEE Computer Society. 3.1

[49] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security*, pages 303–320, 2004. 3, 4.3.1

[50] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004. 3.2.1

[51] Cory Doctorow. Why hotel wifi sucks. `http://www.boingboing.net/2005/10/12/why-hotel-wifi-sucks.html`, October 2005. 4

[52] John R. Douceur. The sybil attack. In *IPTPS*, pages 251–260, 2002. 4.6

[53] M. Duckham and L. Kulik. Location privacy and location-aware computing. In et al. J. Drummond, editor, *Dynamic and Mobile GIS: Investigating Change in Space and Time*, pages 34–51. CRC Press, 2006. 1.3

[54] Federal Trade Commission. Federal trade commission - privacy initiatives. http://www.ftc.gov/privacy/. 1

[55] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, 1993. 3.8.1

[56] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jami Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proc. 15th USENIX Security Symposium*, pages 167–178, Vancouver, Canada, jul-aug 2006. 2.2

[57] Fraud and related activity in connection with access devices. Homeland Security Act (18 U.S.C. §1029), 2002. 4.3.4

[58] Michael J. Freedman and Antonio Nicolosi. Efficient private techniques for verifying social proximity. In *IPTPS*, 2007. 5.3.1

[59] J. Froehlich and J. Krumm. Route prediction from trip observations. In *Society of Automotive Engineers (SAE) 2008 World Congress*, 2008. 1.1

[60] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT*, pages 244–251, 1993. 4.6

[61] Fyodor. Nmap network security scanner. http://insecure.org/nmap/. 2.2

[62] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive*, 2009. 1.1

[63] Ben Greenstein, Damon McCoy, Jeffrey Pang, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *MobiSys '08: 6th International Conference on Mobile Systems, Applications, and Services*, June 2008. 2, 3.4.3, 4.3.3

[64] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Second international conference on security in pervasive computing*, pages 179–192, 2005. 1.1, 1.3, 4.2.1

[65] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proc. 1st international conference on Mobile systems, applications and services*, pages 31–42, New York, NY, USA, 2003. ACM Press. 1.3

[66] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: A quantitative analysis. *ACM Mobile Networks and Applications (MONET)*, 10(3):315–325, 2005. 2, 2.3

[67] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: A quantitative analysis. *ACM MONET*, 10, 2005. 3.1, 3.2.3

[68] Yong Guan, Xinwen Fu, Dong Xuan, Prashanth U. Shenoy, Riccardo Bettati, and Wei Zhao. Netcamo: Camouflaging network traffic for qos-guaranteed mission critical applications. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(4), July 2001. 3

[69] Ramakrishna Gummadi, David Wetherall, Ben Greenstein, and Srinivasan Seshan. Understanding and mitigating the impact of rf interference on 802.11 networks. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 385–396, New York, NY, USA, 2007. ACM. 3.6.2

[70] J. Hall, M. Barbeau, and E. Kranakis. Detection of transient in radio frequency fingerprinting using phase characteristics of signals. In *Proceedings of the 3rd IASTED International Conference on Wireless and Optical Communications (WOC 2003)*, pages 13–18, July 2003. 2.2, 3.6.4, 5.3.3

[71] Dongsu Han, Aditiya Agarwala, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, and Srinivasan Seshan. Mark-and-sweep: getting the "inside" scoop on neighborhood networks. In *IMC*, pages 99–104, 2008. 4, 4.1.1

[72] T. Hashem and L. Kulik. Safeguarding location privacy in wireless ad-hoc networks. In *9th International Conference on Ubiquitous Computing (UbiComp 2007)*, pages 372–390, 2007. 1.3

[73] H. Haverinen and J. Salowey. Extensible authentication protocol method for global system for mobile communications (gsm) subscriber identity modules (eap-sim), January 2006. IETF RFC 4186. http://www.ietf.org/rfc/rfc4186.txt. 3.1

[74] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SECURECOMM 2005)*, pages 194–205, 2005. 1.1, 1.3

[75] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006. 1.1, 1.3

[76] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 161–171, New York, NY, USA, 2007. ACM. 1.3

[77] Hostap driver. http://hostap.epitest.fi/. 3.7.1

[78] Hotspotr. http://hotspotr.com/wifi. 1.1.2, 1.2

[79] HTTP Analyzer. http://www.ieinspector.com/httpanalyzer/. 5

[80] Ieee 802.11i-2004 amendment to ieee std 802.11, 2004. http://standards.ieee.org/getieee802/download/802.11i-2004.pdf. 2, 3.3.5

[81] IETF geographic location/privacy working group charter. http://www.ietf.org/html.charters/geopriv-charter.html. 1

[82] iPass. http://www.ipass.com. 4.2.2

[83] Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for omac, tmac, and xcbc. In *Progress in Cryptology - INDOCRYPT 2003*, pages 133–158, 2003. 3.5, 3.5.2, 4

[84] Tao Jiang, Helen J. Wang, and Yih-Chun Hu. Preserving location privacy in wireless lans. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 246–257, New York, NY, USA, 2007. ACM. 2, 3.1, 3.2.3

[85] JiWire. http://www.jiwire.com. 1.1.2, 1.2, 4, 4.5.2

[86] Glenn Judd and Peter Steenkiste. Using emulation to understand and improve wireless networks and applications. In *NSDI*, pages 203–216, 2005. 4.4.2, 4.4.3

[87] A. Juels. Minimalist cryptography for rfid tags. In *Security of Communication Networks (SCN)*, pages 149–164, 2004. 3.1

[88] Ari Juels. RFID security and privacy: A research survey. *IEEE Journal on Selected Areas in Communication*, 24(2), February 2006. 1.1.1

[89] Ari Juels and Stephen A. Weis. Defining strong privacy for rfid. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 342–347, Washington, DC, USA, 2007. IEEE Computer Society. 3.1

[90] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of p2p traffic. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 121–134, New York, NY, USA, 2004. ACM Press. 2.2

[91] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35(4):229–240, 2005. 2.2

[92] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards ip geolocation using delay and topology measurements. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84, New York, NY, USA, 2006. ACM. 1.1.1

[93] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *IEEE International Conference on Pervasive Services (ICPS2005)*, pages 88–97, 2005. 1.3

[94] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000. 3.4.3

[95] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. Remote physical device fingerprinting. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 211–225, Washington, DC, USA, 2005. IEEE Computer Society. 2.2

[96] Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Toward community sensing. In *IPSN*, pages 481–492, 2008. 4.6

[97] J. Krumm. Inference attacks on location tracks. In *Fifth International Conference on Pervasive Computing (Pervasive 2007)*, pages 127–143, 2007. 1.1, 1.3

[98] J. Krumm. A markov model for driver turn prediction. In *Society of Automotive Engineers (SAE) 2008 World Congress*, 2008. 1.1

[99] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Ubiquitous Computing*, pages 243–260, 2006. 1.1

[100] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 2008. 1.3

[101] Anthony Lamarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Tim Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *Pervasive*, Munich, Germany, 2005. 4.4.1

[102] Paul Lewis. Bluetooth is watching: secret study gives bath a flavour of big brother. The Guardian, `http://www.guardian.co.uk/uk/2008/jul/21/civilliberties.privacy`, July 2008. 1, 1.1.1

[103] John Leyden. Cisco preps wi-fi tracking kit. The Register, `http://www.theregister.co.uk/2005/05/05/cisco_wi-fi_tracking_kit/`, May 2005. 1, 1.1.1

[104] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *MobiCom '01: Proc. 7th annual international conference on Mobile computing and networking*, pages 61–69, New York, NY, USA, 2001. ACM Press. 1

[105] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263, New York, NY, USA, 2006. ACM. 2.2, 3.6.4

[106] libgcrypt. http://directory.fsf.org/project/libgcrypt/. 3.4.3

[107] Janne Lindqvist, Tuomas Aura, George Danezis, Teemu Koponen, Annu Myllyniemi, Jussi Mäki, and Michael Roe. Privacy-preserving 802.11 access-point discovery. In *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, pages 123–130, New York, NY, USA, 2009. ACM. 3.1

[108] Janne Lindqvist and Laura Takkinen. Privacy management for secure mobility. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 63–66, New York, NY, USA, 2006. ACM. 3.1

[109] Location privacy protection act of 2001. U.S. Senate bill, `http://www.theorator.com/bills107/s1164.html`, 2001. 1

[110] Madwifi driver. http://madwifi.org/. 3.4.2, 3.7.1

[111] S. Mascetti and C. Bettini. A comparison of spatial generalization algorithms for lbs privacy preservation. In *International Workshop on Privacy-Aware Location-based Mobile Services (PALMS 2007)*, 2007. 1.3

[112] Yutaka Matsuo, Naoaki Okazaki, Kiyoshi Izumi, Yoshiyuki Nakamura, Takuichi Nishimura, and Kiti Hasida. Inferring long-term user property based on users' location history. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007. 1.1

[113] Microsoft. Wireless client update for windows xp with service pack 2. `http://support.microsoft.com/kb/917021`. 2

[114] M.F. Mokbel, C.-Y. Chow, and W.G. Aref. The new casper: Query processing for location services without compromising privacy. In *International Conference on Very Large Data Bases (VLDB 2006)*, pages 763–774, 2006. 1.3

[115] Fabian Monrose and Aviel Rubin. Authentication via keystroke dynamics. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 48–56, New York, NY, USA, 1997. ACM Press. 2.2

[116] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press. 2.2, 2.5.2

[117] Muniwireless. MuniWireless is the portal for news and information about citywide wireless broadband projects around the world. 2.3

[118] Ellen Nakashima. Judge limits searches using cellphone data. Washington Post, `http://www.washingtonpost.com/wp-dyn/content/article/2008/09/11/AR2008091103292.html`, September 2008. 1.1.2

[119] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, 2001. 3.8.1

[120] W.M. Newman, M.A. Eldridge, and M.G. Lamming. Pepys: Generating autobiographies by automatic tracking. In *Second European Conference on Computer Supported Cooperative Work (ECSCW 1991)*, pages 175–188, 1991. 1.1

[121] R.E. Newman-Wolfe and B.R. Venkatraman. Performance analysis of a method for high level prevention oftraffic analysis. In *Computer Security Applications Conference*, December 1992. 3

[122] Anthony J. Nicholson, Yatin Chawathe, Mike Y. Chen, Brian D. Noble, and David Wetherall. Improved access point selection. In *MobiSys*, pages 233–245, 2006. 4, 4.1.1, 4.5.1, 4.5.1, 4.6

[123] Helen F. Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1), 2004. 1

[124] Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In *International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy: Current Status and Future Directions*, Nottingham, England, September 2004. 3.1

[125] p0f. http://freshmeat.net/projects/p0f/. 2.2

[126] Balaji Padmanabhan and Yinghui Yang. Clickprints on the web: Are there signatures in web browsing data?, 2006. 2.2

[127] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *MobiCom*, September 2007. 1, 3.2.3

[128] Jeffrey Pang, Ben Greenstein, Michael Kaminsky, Damon McCoy, and Srinivasan Seshan. Wifi-reports: Improving wireless network selection with collaboration. In *MobiSys '09: 7th International Conference on Mobile Systems, Applications, and Services*, June 2009. 3

[129] Jeffrey Pang, Ben Greenstein, Damon Mccoy, Srinivasan Seshan, and David Wetherall. Tryst: The case for confidential service discovery. In *HotNets*, 2007. 2, 3.2.1, 4.2.1, 5.3.1

[130] Donald Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In *UbiComp 2003: Ubiquitous Computing*, pages 73–89, 2003. 1.1

[131] Neal Patwari and Sneha K. Kasera. Robust location distinction using temporal link signatures. In *MobiCom*, 2007. 2.2, 3.2.1

[132] A. Pfitzmann and M. Kohntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, 2000. 1.3

[133] David Pogue. Wi-fi to go, no cafe needed. New York Times, `http://www.nytimes.com/2009/05/07/technology/personaltech/07pogue.html`, May 2009. 1.1.1

[134] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17:27–35, 2003. 5

[135] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0. 2.5.2

[136] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. *SIGCOMM CCR*, 36(4), 2006. 3.4.2

[137] Robert Morris' webpage. http://pdos.csail.mit.edu/∼rtm/. 2.1

[138] Maya Rodrig, Charles Reis, Ratul Mahajan, David Wetherall, John Zahorjan, and Ed Lazowska. CRAWDAD data set uw/sigcomm2004 (v. 2006-10-17). `crawdad.cs.dartmouth.edu/meta.php?name=uw/sigcomm2004`. (document), 3.1

[139] Phillip Rogaway. Nonce-based symmetric encryption. In *Proc. FSE 2004*, pages 348–359. Springer, 2004. 3.3.3, 3.5.1, 3.5.2, 3.5.2, 5

[140] Roofnet. http://pdos.csail.mit.edu/roofnet/doku.php. 2.1, 3.4.3

[141] Naouel B. Salem, Jean-Pierre Hubaux, and Markus Jakobsson. Reputation-based wi-fi deployment protocols and security analysis. In *WMASH*, pages 29–40, 2004. 4.6

[142] Scott T. Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *USENIX Security*, 2007. 3, 3.6.3

[143] Bartlomiej Sieka. Using radio device fingerprinting for the detection of impersonation and sybil attacks in wireless networks. *Security and Privacy in Ad-Hoc and Sensor Networks*, 4357:179–192, 2006. 2.2, 3.6.4, 5.3.3

[144] Dave Singelée and Bart Preneel. Location privacy in wireless personal area networks. In *WiSe*, 2006. 3.1

[145] Skyhook wireless. http://www.skyhookwireless.com/. 4.4.1

[146] Soekris engineering. http://www.soekris.com/net4801.htm. 3.7.2

[147] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *10th USENIX Security Symposium*, 2001. 2.2, 3

[148] JH. Song, R. Poovendran, J. Lee, and T. Iwata. The AES-CMAC algorithm. RFC 4493, June 2006. 3.3.3

[149] Speedtest.net. http://www.speedtest.net/. 4.6

[150] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 19, Washington, DC, USA, 2002. IEEE Computer Society. 2.2, 2.7.1, 3

[151] Karthikeyan Sundaresan and Konstantina Papagiannaki. The need for cross-layer information in access point selection algorithms. In *IMC*, pages 257–262, 2006. 4.4.3, 4.5.1, 4.6

[152] Jani Suomalainen, Jukka Valkonen, and N. Asokan. Security associations in personal networks: A comparative analysis. Technical Report NRC-TR-2007-004, Nokia Research Center, January 2007. 3.2.1, 3.8.1, 5.3.1

[153] P. Tao, A. Rudys, A. Ladd, and D. Wallach. Wireless LAN location sensing for security application. In *WISE*, 2003. 2.2, 3.2.1, 3.6.4

[154] Ping Tao, Algis Rudys, Andrew M. Ladd, and Dan S. Wallach. Wireless lan location-sensing for security applications. In *WiSE '03: Proceedings of the 4rd ACM workshop on Wireless security*, September 2003. 2.3

[155] Linda Tauscher and Saul Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. *Int. J. Hum.-Comput. Stud.*, 47(1), 1997. 2.5.1

[156] tcpdump. http://www.tcpdump.org/. 2.3

[157] Telecommunications act of 1996. Section 222, `http://www.fcc.gov/telecom.html`, 1996. 1.1.2

[158] United States Census Bureau. Table 1: Annual estimates of the population for incorporated places over 100,000. 2007. `http://www.census.gov/popest/cities/tables/SUB-EST2007-01.csv`. 4.5.2

[159] C.J. van Rijsbergen. *Information Retrieval. 2nd ed.* Butterworths, 1979. 2.5.2

[160] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley. Facilitating access point selection in ieee 802.11 wireless networks. In *IMC*, pages 1–6, 2005. 4.4.2, 4.4.3, 4.5.1, 4.6

[161] B.R. Venkatraman and R.E. Newman-Wolfe. Performance analysis of a method for high level prevention oftraffic analysis using measurements from a campus network. In *Computer Security Applications Conference*, December 1994. 3

[162] Kevin Walsh and Emin G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI*, 2006. 4.2.1, 4.6

[163] Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212, 2004. 3.1

[164] A. Westin. *Privacy and Freedom*. Atheneum, 1967. 1

[165] Wi-fi hacking, with a handheld pda. ZDNet, February 2007. http://blogs.zdnet.com/security/?p=19. 2.3

[166] D. Wilson and C. Atkeson. Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors. In *Third International Conference on Pervasive Computing (Pervasive 2005)*, pages 62–79, 2005. 1.3

[167] Wireless geographic logging engine. http://www.wigle.net/. 1.1.2, 1.2, 2.1, 2.3, 4.5.2

[168] Wireless location tracking draws privacy questions. CNET News.com, May 2006. `http://news.com.com/Wireless+location+tracking+draws+privacy+questions/2100-1028_3-6072992.html`. 1, 4

[169] Wireless privacy protection act of 2005. U.S. House bill, `http://www.theorator.com/bills109/hr83.html`, 2005. 1

[170] Ford-Long Wong and Frank Stajano. Location privacy in bluetooth. In *ESAS*, pages 176–188, 2005. 3.1

[171] Charles Wright, Lucas Ballard, Fabian Monrose, and Gerald Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *USENIX Security*, August 2007. 3, 3.6.3

[172] C.V. Wright, F. Monrose, and G.M. Masson. On inferring application protocol behaviors in encrypted network traffic. *Journal of Machine Learning Research*, August 2006. 2.1, 2.2, 2.5.2, 3, 3.6.3

[173] T.-H. You, W.-C. Peng, and W.-C. Lee. Protecting moving trajectories with dummies. In *International Workshop on Privacy-Aware Location-based Mobile Services (PALMS 2007)*, 2007. 1.3

[174] You can't send secure email from starbuck's (at least not easily). http://staff.washington.edu/oren/blog/2004/04/you-cant-send-s.html, April 2004. 4

[175] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Security and Privacy*, pages 3–17, 2008. 4.3.1

[176] Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B. Gibbons, and Feng Xiao. DSybil: Optimal sybil-resistance for recommendation systems. In *IEEE Security and Privacy*, 2009. 4.2.1, 4.6, 4.7.1

[177] Sebastian Zander, Thuy Nguyen, and Grenville Armitage. Automated traffic classification and application identification using machine learning. In *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, pages 250–257, Washington, DC, USA, 2005. IEEE Computer Society. 2.2, 2.5.2